F-Tile Avalon® Streaming IP for PCI Express* Design Example User Guide

Updated for Quartus® Prime Design Suite: **25.3**

IP Version: **12.4.0**

Online Version
Send Feedback

683372
UG-20332
2026.02.10

Contents

1. Acronyms....3

2. Design Example Description....4

2.1. Programmed Input/Output Design Example....4

2.1.1. Programmed Input/Output Design Example Functional Description....7

2.1.2. Programmed Input/Output Design Example Limitation....12

2.1.3. Programmed Input/Output Design Example Simulation Testbench....12

2.2. Single Root I/O Virtualization (SR-IOV) Design Example....14

2.2.1. Single Root I/O Virtualization (SR-IOV) Design Example Functional Description....15

2.2.2. SR-IOV Design Example Limitation....16

2.2.3. Single Root I/O Virtualization (SR-IOV) Design Example Simulation Testbench..17

2.3. Performance Design Example....17

2.3.1. Performance Design Example Functional Description....18

2.3.2. Performance Design Example Simulation Testbench....20

3. Quick Start Guide....22

3.1. Directory Structure....23

3.2. Generating the Design Example....24

3.3. Simulating the Design Example....27

3.3.1. Steps to Run Simulation....28

3.3.2. Testbench....31

3.4. Compiling the Design Example....37

3.5. Hardware and Software Requirements....38

3.6. Program the FPGA....38

3.7. Installing the Linux Kernel Driver....40

3.8. Running the Design Example....41

3.8.1. Running the PIO Design Example....42

3.8.2. Running the SR-IOV Design Example....44

3.8.3. Running the Performance Design Example....47

4. F-Tile Avalon Streaming IP for PCI Express Design Example User Guide Archives....56

5. Revision History for the F-Tile Avalon Streaming IP for PCI Express Design Example User Guide....57

1. Acronyms

Table 1. Terms and Definitions

Term	Definition
AVMM	Avalon Memory Mapped
AVST	Avalon Streaming
BAM	Burst Avalon Master
CplD	Completion with Data
DUT	Design Under Test
DW	Double Word
ED	Example Design
FBE	First Byte Enable
FIFO	First In First Out
Gen3	PCIe* 3.0
Gen4	PCIe 4.0
PIO	Programmed Input/Output
LBE	Last Byte Enable
MPS	Maximum Payload Size
MRd	Memory Read
MWr	Memory Write
RX	Receiver
HIP	Hard IP
TLP	Transaction Layer packet
TX	Transmit
SR-IOV	Single Root Input/Output Virtualization

2. Design Example Description

The F-Tile Avalon-ST IP for PCI Express Design Example is a simple design to demonstrate the establishment of PCIe connectivity of F-Tile FPGA in Quartus® Prime. The design performs write and read sequences from the host processor to the target device through PCIe Quartus Prime Hard IP. The Programmed Input/Output (PIO) application block is needed to handle the translation from PCIe TLP to AVMM protocol.

Table 2. F-Tile Avalon-ST IP Design Examples

Design Example	Hard IP Mode	Simulation	Hardware
PIO	Gen4 x16 512-bit Endpoint	Supports VCS*, VCS MX, Questa*, Xcelium* and Riviera-PRO* simulators.	Agilex™ 7 FPGA F-Series Development Kit (Production 1 2x F-Tile) (F-Tile B0 OPN: AGFB027R24C2E2VR2)
	Gen4 x8x8 256-bit Endpoint		
	Gen4 x8 256-bit Endpoint		
	Gen3 x16 512-bit Endpoint		
	Gen3 x8 256-bit Endpoint		
	Gen3 x8x8 256-bit Endpoint		
	Gen4 x8x8 512-bit Endpoint		
	Gen3 x16 256-bit Endpoint		
SR-IOV	Gen4 x16 512-bit Endpoint		
	Gen3 x16 512-bit Endpoint		
	Gen4 x8x8 256-bit Endpoint		
	Gen3 x8x8 256-bit Endpoint		
Performance	Gen4 x16 512-bit Endpoint		

Note: Design examples only support the default settings in the Parameter Editor of the F-Tile Avalon Streaming IP for PCI Express.

Note: Design examples do not support 10-bit tag completer feature. Running the design example on host machine enforce 10-bit tag at PCIe Gen4, can cause completion timeout or system crashed.

2.1. Programmed Input/Output Design Example

The PCIe F-Tile Design Example is designed to highlight the application of PCIe Gen4 in F-Tile. The PCIe IP run up to 500 MHz at user interface with the maximum data width of 512 bits for Gen4 x16 and 256 bits for each of the two Gen4 x8 ports.

The clock frequency supplied from the `coreclkout_hip` is limited to 500 MHz.

The Design Example consists of 3 main components to display the intended design

- F-Tile Avalon-ST IP for PCI Express Hard IP (DUT)
- Programmable I/O (PIO) Design Example for PCI Express G4
- On-Chip Memory (MEM)

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 required files to simulate and compile in the Quartus Prime software. The design example covers a wide range of parameters. However, it does not cover all possible parameterizations of the F-Tile Hard IP for PCIe.

Figure 1. PCIe Gen3/Gen4 x16 Design Example Variant Block Diagram

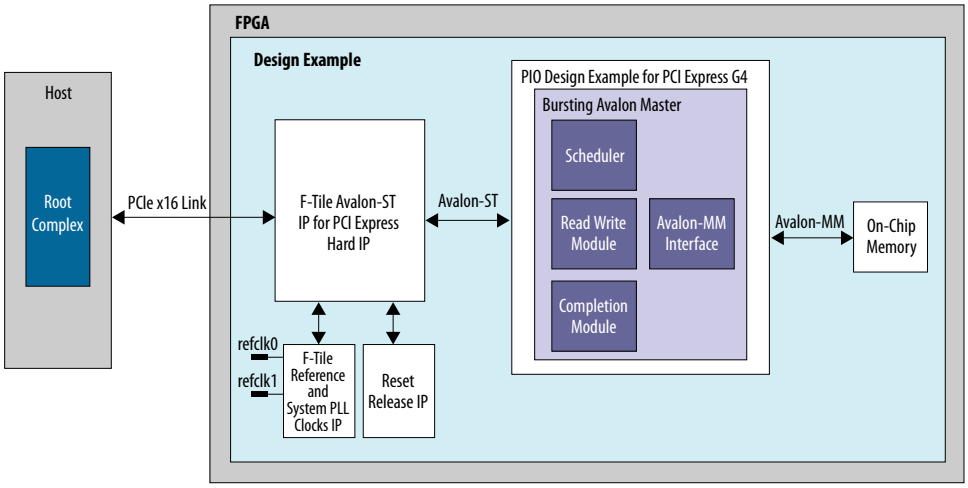


Figure 2. PCIe Gen3/Gen4 x8x8 Design Example Variant

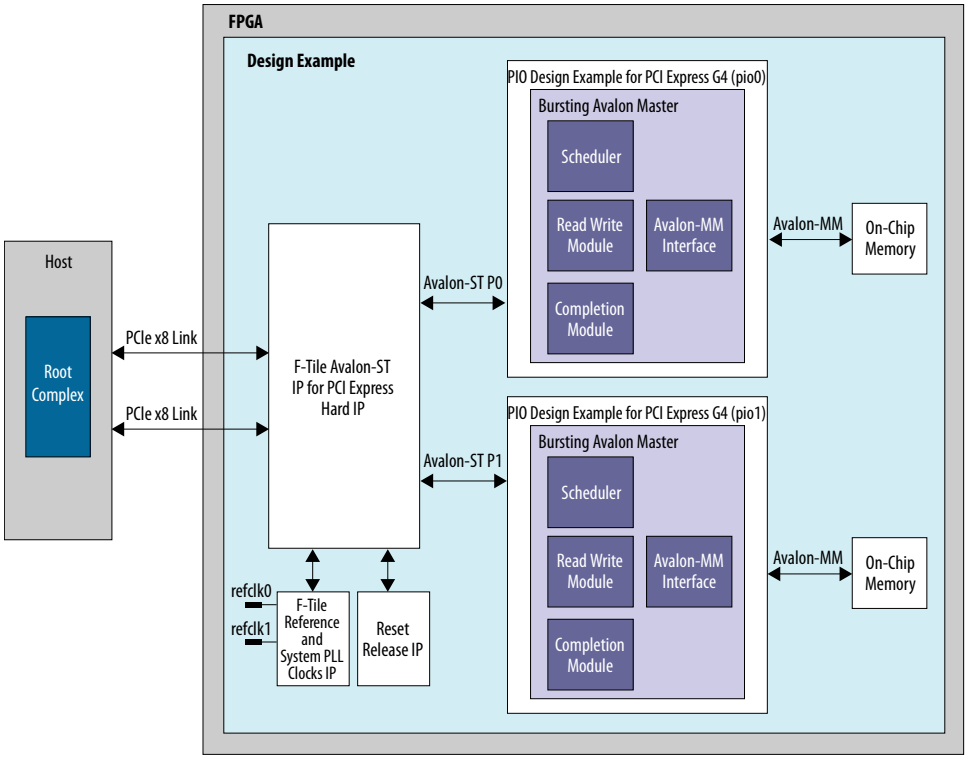
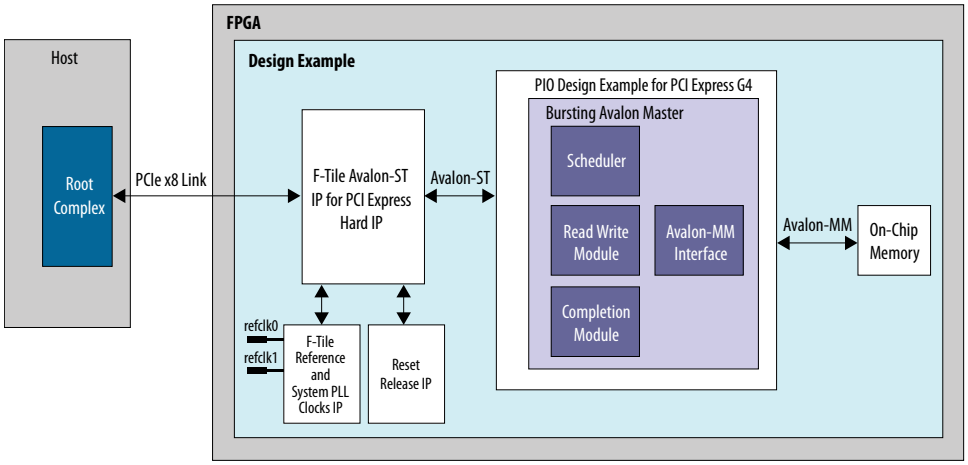


Figure 3. PCIe Gen3/Gen4 x8 Design Example Variant



2.1.1. Programmed Input/Output Design Example Functional Description

Figure 4. Platform Designer System Contents for F-Tile Avalon®-ST IP for PCI Express PIO Design Example [Gen4 x16 variant]

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

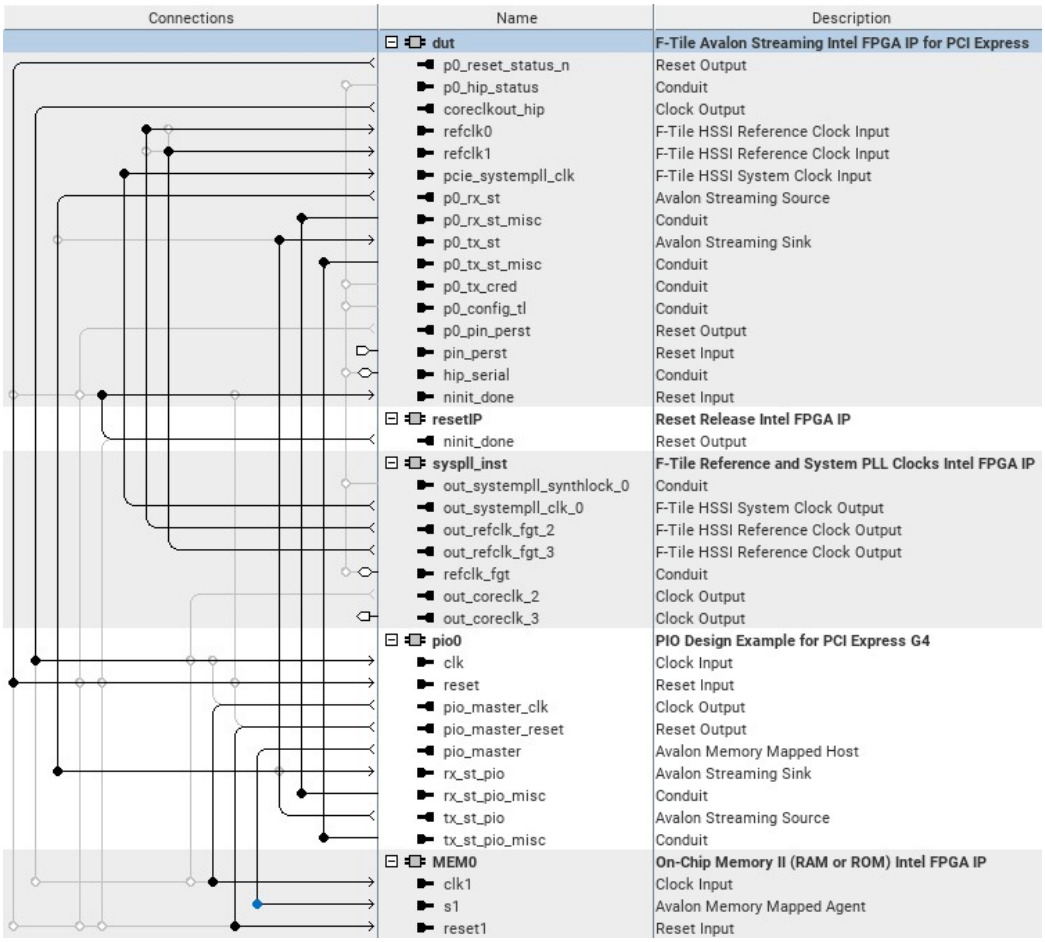


Figure 5. Platform Designer System Contents for F-Tile Avalon®-ST IP for PCI Express PIO Design Example [Gen4 x8x8 variant]

The Platform Designer generates this design for up to Gen4 x8x8 variants.

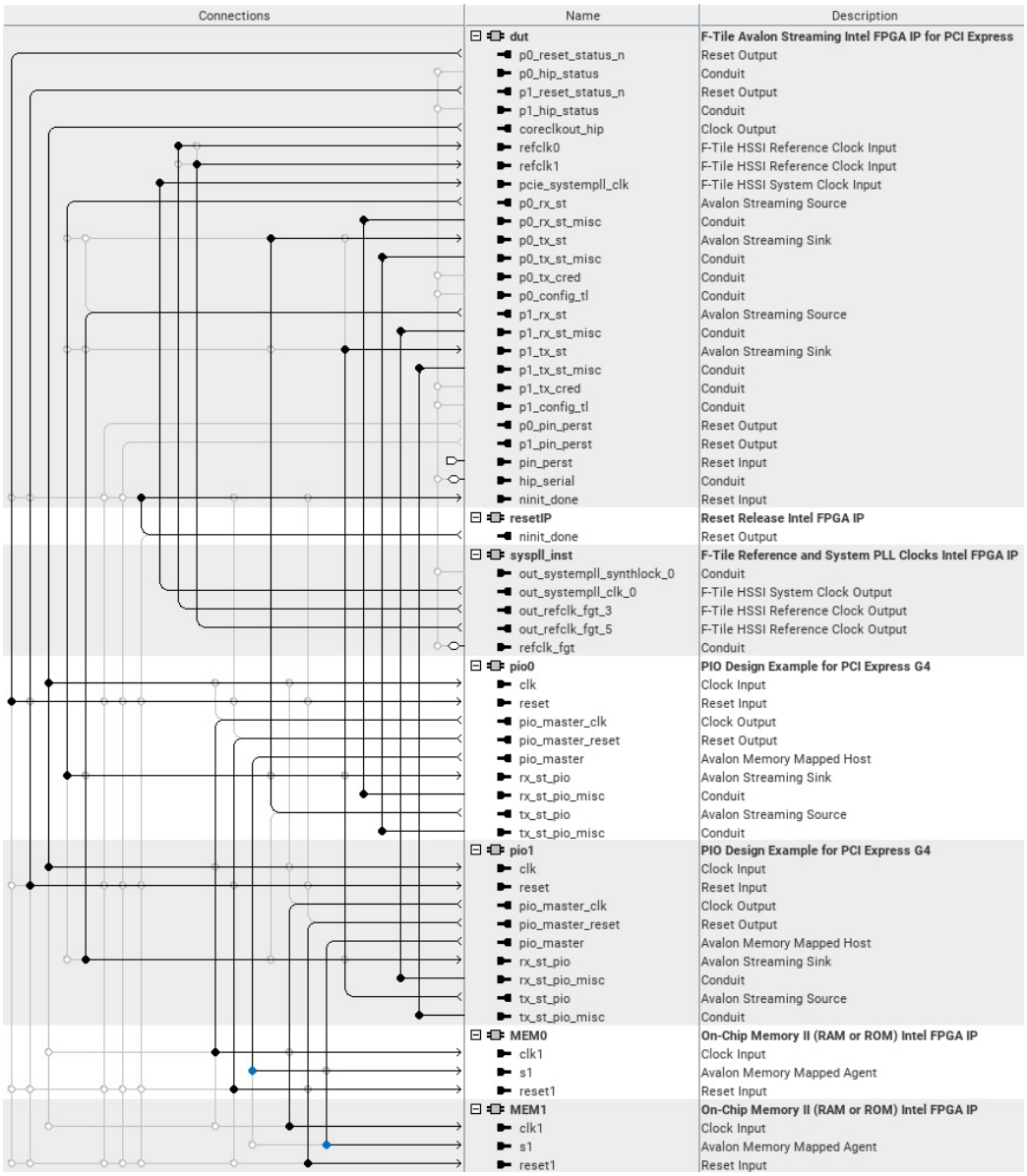
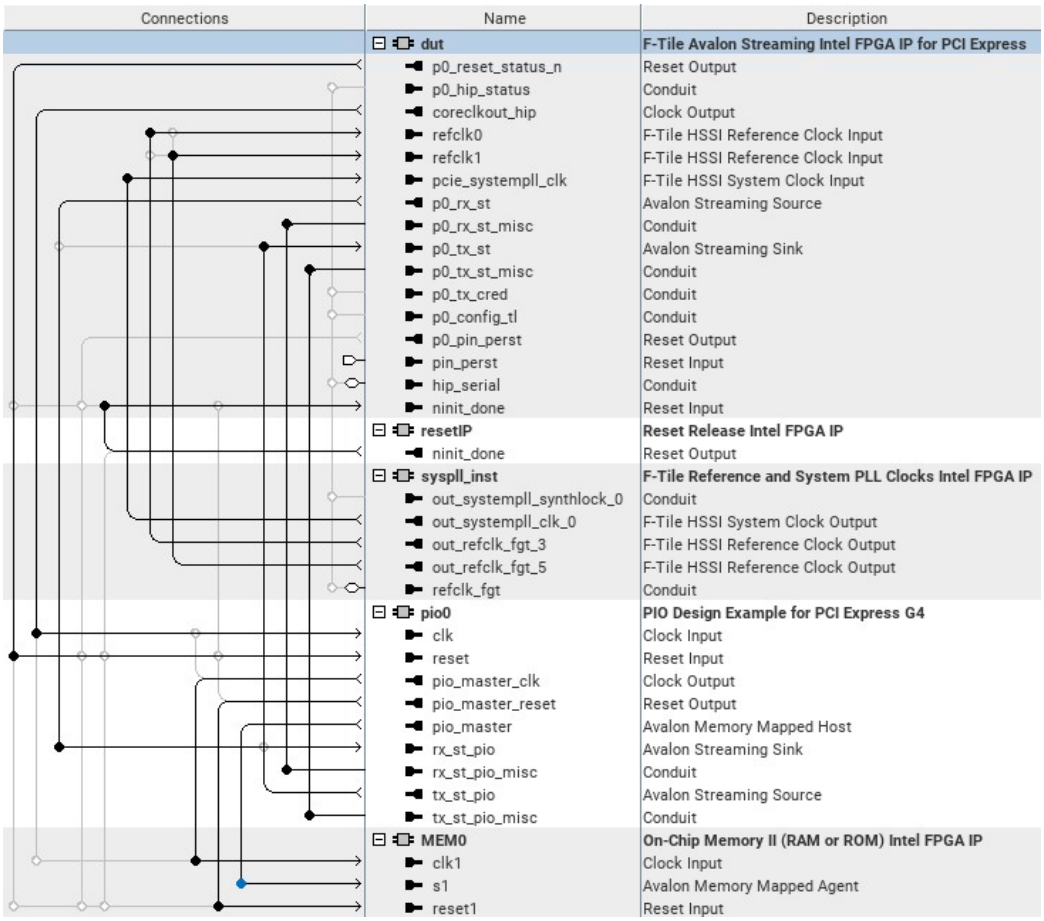


Figure 6. Platform Designer System Contents for F-Tile Avalon®-ST IP for PCI Express PIO Design Example [Gen4 x8 variant]

The Platform Designer generates this design for up to Gen4 x8 variants.



This design example includes the following components

2.1.1.1. F-Tile Avalon-ST IP for PCI Express Hard IP (DUT)

This is the F-Tile Avalon-ST IP for PCI Express Hard IP (DUT) with the parameters you specified in the IP parameter editor. This component drives TLP data received to the PIO Design Example for PCI Express G4. The DUT component is the F-Tile Avalon-ST IP for PCI Express Hard IP configured as Endpoint interacting with the root complex/ switch at the other end. The DUT component translates the PCIe serial link transfer interface to Avalon-ST interface.

2.1.1.2. PIO Design Example for PCI Express G4

This component performs the necessary translation between the PCI Express TLPs and simple Avalon®-MM writes and reads to the onchip memory. The PIO component interfaces between the Avalon-ST and Avalon-MM. It decodes the TLP headers/data and converts it into Avalon-MM compatible instructions. For a write operation, a single write data TLP is converted into a single AVMM write instruction. While for the read operation, it could be multiple data read back depending on the max payload size

boundary. It reads and writes in 512 bits and support contiguous byte enables. These operations are done in the Bursting Avalon Master module in the PIO component which consists of 4 sub-modules, namely the Scheduler, Read Write Module, Avalon-MM Interface and Completion Module.

2.1.1.2.1. Scheduler

The data stream from DUT is buffered in the data FIFO within the scheduler block. The header will be pre-processed to and calculate some prerequisite values. It will pass the command to Read Write Module after precalculated the write burst count and write First DWORD Byte Enable and Last DWORD Byte Enable from the header information.

- Write sequence: The data in the data FIFO is forwarded to Read Write Module.
- Read sequence: Read TLP attributes to generate the completion TLP and additional processing to determine the Type1 read and Type2 read. When the TLP DW length is less than Max Payload Size boundary, Type1 read is initiated. When TLP DW length is larger than Max Payload Size boundary, the Type2 read is initiated. Each Type1 and Type2 read could only hold up to the size of its MPS. No data in the data FIFO is forwarded to the Read Write Module since this is a read sequence. Only the pre-processed commands are forwarded to the pre-processed command FIFO for Avalon-MM generation in the Read Write Module.

2.1.1.2.2. Read Write Module

Prepare the Avalon-MM write or read sequence for the Avalon-MM Interface.

- Write sequence: Data is aligned by the Barrel Shifter and passed to aligned data FIFO. The write state machine extracts the address, burst count, FBE/LBE and generate Avalon-MM write command. The command is then stored to the Avalon-MM command FIFO and eventually passed to the Avalon-MM Interface. Maximum write burst count allowed is 8 as decided from the aligned data FIFO depth.
- Read sequence: The read state machine decodes the combination of Type1 read and Type2 read based on the PREPROC CMD. Next it generates the Avalon-MM command accordingly for each Type1 or Type2 read. Concurrently, the CPL command is generated for each AVMM read command and stored to the CPL CMD FIFO. In the event of waitrequest by the MEM device, Avalon-MM command FIFO can hold up to 16 data cycles.

2.1.1.2.3. Avalon-MM Interface

When Avalon-MM command from the Read Write Module is available, Avalon-MM Master State Machine checks the Completion Buffer watermark to ensure it is not full and the Avalon-MM Master, the driver to the On-Chip memory is not in waitrequest. When the conditions are met, the Avalon-MM Master State Machine gets Avalon-MM command or data and issues the write/read to the Avalon-MM Master. Concurrently the Avalon-MM data is directly flushed to the Avalon-MM Master. For read sequence, the returned data from the Avalon-MM Master is stored at Completion Buffer and ready to move to Completion module.

2.1.1.2.4. Completion Module

When both Completion command from Read Write Module and read data from Avalon-MM interface are available, Completion State Machine captures the information. The Completion command will be stored into the Completion command FIFO. The read

data will be stored into Aligned Completion Data Buffer after shifted by the Barrel Shifter. The stored Completion command and read data will be shifted out to the PCIe upstream through TX Completion.

2.1.1.2.5. Width Adapter

Width Adapter converts the Avalon-ST signals from 256 bit @ 500 MHz to 512 bit @ 250 MHz. This adaptation is to maintain data bandwidth to reuse the existing BAM architecture and to interface between the two clock domains. Features like TX and RX Credit Interface, Error Interface, FLR Interface, CII Interface and Interrupt Interface are not used in the design example. This is applicable for PCIe 1x8 and 2x8 design example variants only.

2.1.1.3. On-Chip Memory (MEM)

An on-chip memory (MEM) component stores and reads data depending on the instructions from PIO. The size of the on-chip memory is configured to 16 KB for both PCIe Gen3/4 x16 and PCIe Gen3/4 x8x8 design example variants.

2.1.1.4. F-Tile Reference and System PLL Clocks IP

This IP is required for F-Tile PCIe interface implementation to configure the reference clock for the FGT PMA and System PLL. The clock from this IP is a logical connection. It is physically inside the F-Tile Avalon Streaming IP for PCI Express Hard IP. The main clock of the PIO design example originates from `coreclkout_hip` of the F-Tile Avalon Streaming IP for PCI Express Hard IP running at 500 MHz. The clock originates from the System PLL.

The reference clock to the System PLL must adhere to the following requirements:

- If compliance to PCIE link training timing specifications is required, the reference clock to the System PLL must be available and stable before device configuration begins. You must set the **Refclk is available at power-on** parameter in the System PLL IP to **On**. You must derive the reference clock from an independent and free-running clock source. Alternately, if the reference clock from the PCIe link is guaranteed available before device configuration begins, you can use it to drive the System PLL. Once the PCIe link `refclk` is alive, it can never be allowed to go down.
- If compliance to PCIE link training timing specifications is not required and the reference clock to the System PLL may not be available before device configuration begins, you must set the **Refclk is available at power-on** parameter in the System PLL IP to **Off**. In this case, you may use the reference clock from the PCIE link to drive the System PLL. The System PLL does not lock to the reference until you perform the Global Avalon memory-mapped interface write operations signaling that the reference clock is available.

  Note: Refer to the *Example Flow to Indicate All System PLL Reference Clocks are Ready* section in the F-Tile Architecture and PMA and FEC Direct PHY IP User Guide to trigger the System PLL to lock to the reference clock.

Once the reference clock for the System PLL is up, it must be stable and present throughout the device operation and must not go down. If you are not able to adhere to this requirement, you must reconfigure the device.

2.1.1.5. Reset Release IP

This IP holds a 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 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.

Note: For more information on Reset Release IP, refer to *Agilex 7 Configuration User Guide*.

Note: For more information about the F-Tile Reference and System PLL Clocks IP, refer to *F-Tile Architecture and PMA and FEC Direct PHY IP User Guide*

Related Information

- Agilex 7 Configuration User Guide
- F-Tile Architecture and PMA and FEC Direct PHY IP User Guide

2.1.2. Programmed Input/Output Design Example Limitation

F-Tile Avalon Streaming IP for PCI Express Design Example fails to write back-to-back transactions correctly.

For the PIO design example, there is no support for the back-to-back TLP packets from the host processor.

The design example is intended to handle simple read-write instructions based on the TLP command. TLP transaction of memory write request (MWr) and write the data to the MEM device. As for the TLP transaction of memory read request (MRd), the design will read the data from the MEM device and return completion with data (CplD).

Note: This design example does not include the full feature of the F-Tile Avalon Streaming IP for PCI Express. Hence, it is not suitable for customer design reference.

2.1.3. Programmed Input/Output Design Example Simulation Testbench

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

Figure 7. Block Diagram for the PCIe x16 PIO Design Example Simulation Testbench

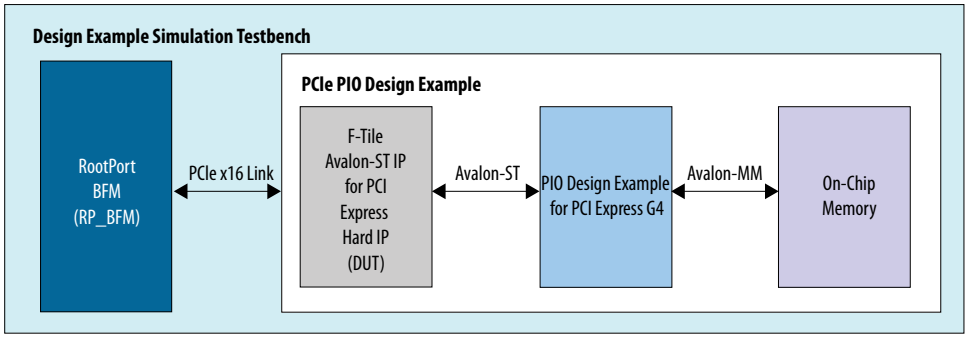
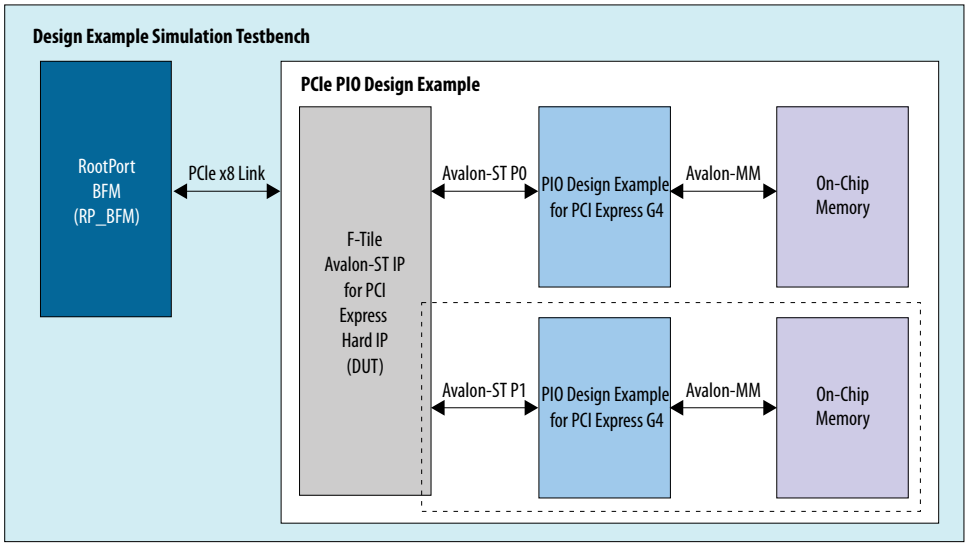
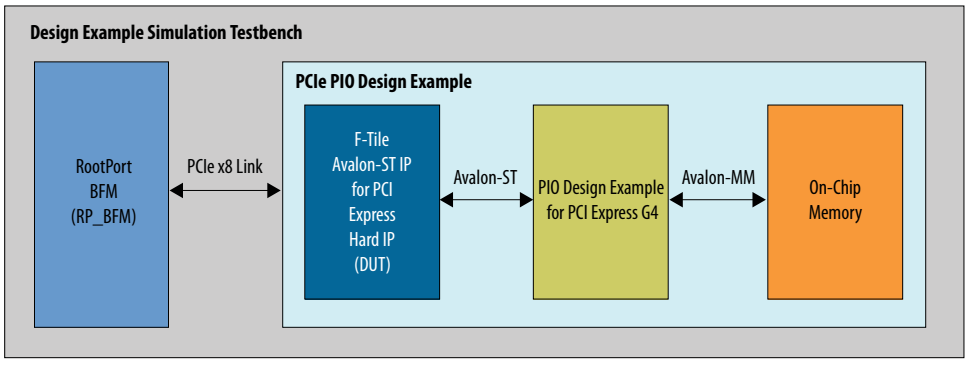


Figure 8. Block Diagram for the PCIe x8x8 PIO Design Example Simulation Testbench



Note: The simulation testbench for PCIe x8x8 PIO Design Example is configured for a single PCIe x8 link although the actual design implements two PCIe x8 links.

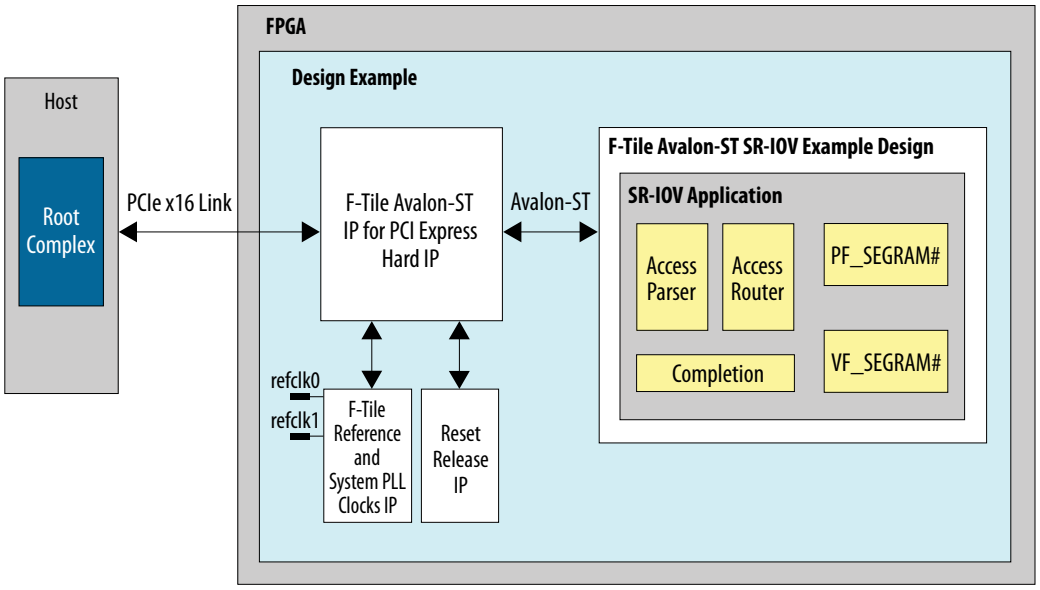
Figure 9. Block Diagram for the PCIe x8 PIO Design Example Simulation Testbench



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.

2.2. Single Root I/O Virtualization (SR-IOV) Design Example

Figure 10. PCIe Gen 4 x16 Single Root I/O Virtualization (SR-IOV) Design Example Block Diagram



The SR-IOV design example performs memory transfers from a host processor to a target Agilex 7 device configured as Gen4 x16 Endpoint. The design example runs up to 500 MHz at user interface with the maximum data width interface of 512 bits. It demonstrates the SR-IOV capability of 2 PF and up to 32 VF per PF. With two PFs and 32 VFs per PF, there are 66 memory locations that the design example can access. The two PFs can access two memory locations, while the 64 VFs (2 x 32) can access 64 memory locations.

The design example is intended to handle simple read/write instructions based on TLP command. TLP transaction of memory write request (MWr) writes the data to the designated RAM memory space. As for the TLP transaction of memory read request (MRd), the design reads the data from RAM memory space and returns completion with data (CplD). The data and address requested to access the F-Tile Avalon-ST SR-IOV Example Design must be double word aligned.

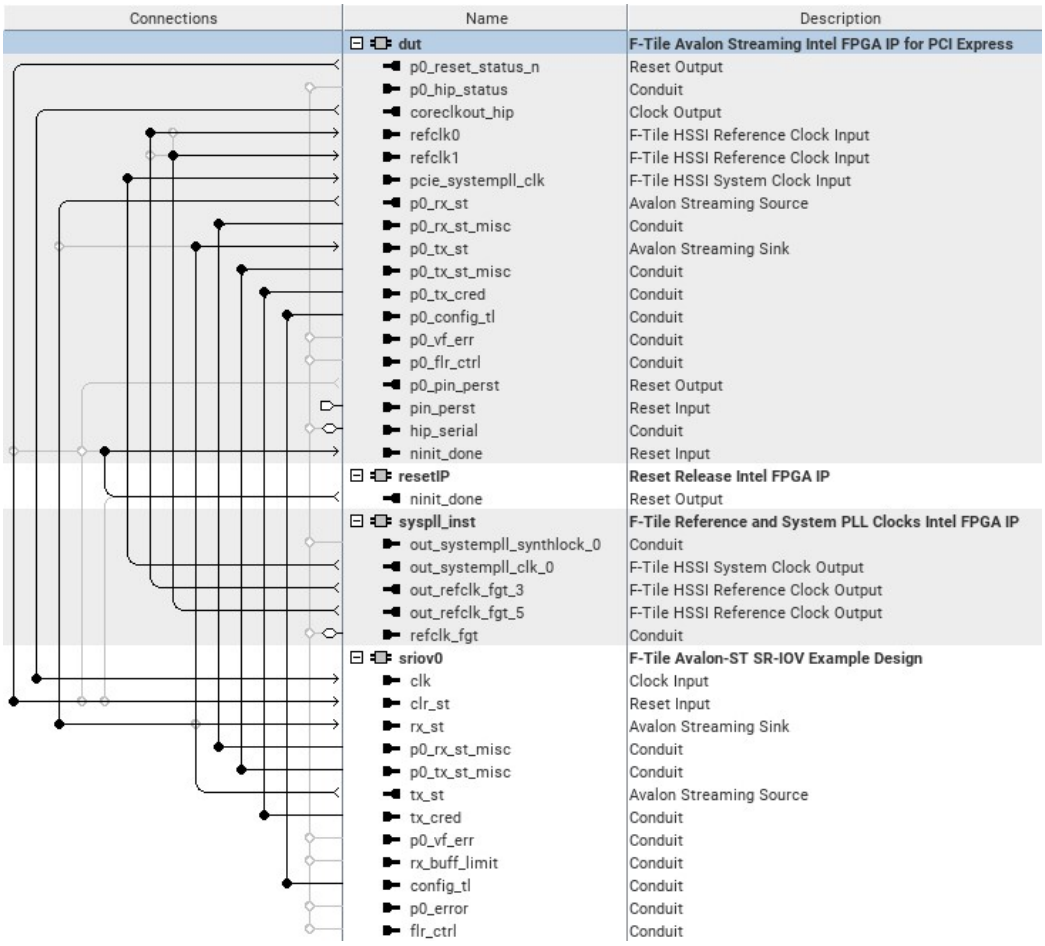
The SR-IOV design example automatically creates the files necessary to simulate and compile in the Quartus Prime software. The design example covers a wide range of parameters. However, it does not cover all possible parameterizations of the F-Tile Hard IP for PCIe.

This design example consists of 2 main components:

- F-Tile Avalon-ST IP for PCI Express Hard IP Endpoint variant (DUT)
- F-Tile Avalon-ST SR-IOV Example Design

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

Figure 11. Platform Designer System Contents for F-Tile Avalon-ST IP for PCI Express SR-IOV Design Example [Gen4 x16 variant]



2.2.1.1. F-Tile Avalon-ST IP for PCI Express Hard IP (DUT)

The DUT component is the F-Tile Avalon-ST IP for PCI Express Hard IP configured as Endpoint interacting with the root complex/switch on one end, and drives the received TLP data to the SR-IOV application at the other end. The DUT component translates the PCIe serial link transfer interface to Avalon-ST interface.

2.2.1.2. F-Tile Avalon-ST SR-IOV Example Design

The F-Tile Avalon-ST SR-IOV Example Design interfaces with the DUT through the AVST interface. It initiates a series of RAMs to store the incoming data from the DUT. The number of RAM is determined by PF and VF from the user. During operation modes, the F-Tile Avalon-ST SR-IOV Example Design decodes the TLP headers/data

into MWr or MRd instruction. If the instructions is valid, MWr or MRd is performed to the PF and VF Segram according to the PF and VF information from the AVST interface with DUT.

2.2.1.2.1. Access Parser

Access Parser is a submodule to decode and validate the incoming instruction whether it is valid and good to proceed to the next phase of processing. It ensures the target address and data is double word aligned. If the received MWr instruction is not fulfilling the requirement, it is silently dropped. While for MRd instruction which does not fulfill the requirement, it completely aborts.

2.2.1.2.2. Access Router

Access Router is a network of connections that connects all the instantiated Segrams. It reroutes the data from the source to the target destination. Based from the decoded PF and VF information from the Access Parser, the targeted Segram activates and does further processing.

2.2.1.2.3. Completion

The Completion submodule stores the necessary header information from the incoming MRd instruction and repackages the acquired information into a CplD header. The CplD header is released together with the fetched data from targeted Segram.

2.2.1.2.4. PF Segram

The PF Segram is a RAM which is instantiated from the ED based on the PF number. Each RAM instantiated consists of 64-bit data width and 64 depth which taking 4 KB.

2.2.1.2.5. VF Segram

The VF Segram is a RAM which is instantiated from the example design based on the PF and VF number. Each RAM instantiated consists of 64-bit data width and 64 depth which takes 4 KB.

2.2.1.2.6. F-Tile Reference and System PLL Clocks IP

Note: Refer to F-Tile Reference and System PLL Clocks IP on page 11 for additional information.

2.2.1.2.7. Reset Release IP

Note: Refer to Reset Release IP on page 12 for additional information.

2.2.2. SR-IOV Design Example Limitation

F-Tile Avalon Streaming IP for PCI Express Design Example fails to write back-to-back transactions correctly.

For the SR-IOV design example, there is no support for the back-to-back TLP packets from the host processor.

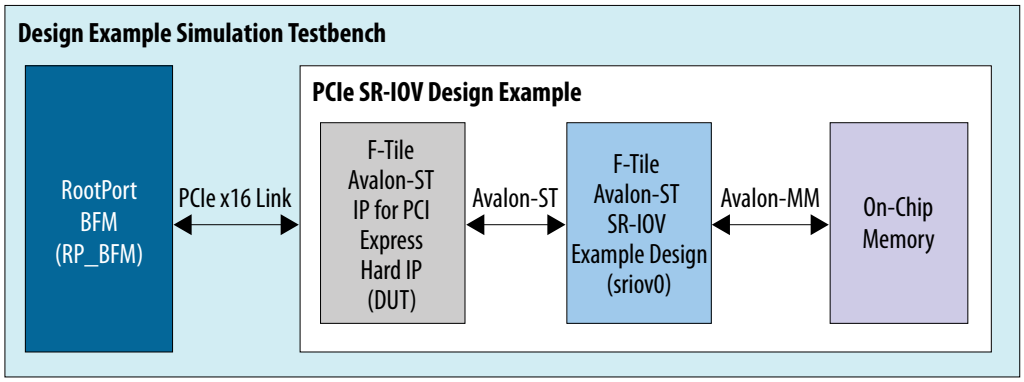
The design is intended to handle simple read-write instructions based on the TLP command. TLP transaction of memory write request (MWr) and write the data to the designated RAM memory space. As for the TLP transaction of memory read request (MRd), the design will read the data from RAM memory space and return completion with data (CplD).

No upstream request from the F-Tile Avalon-ST SR-IOV Example Design. The data and address requested to access the F-Tile Avalon-ST SR-IOV Example Design must be DW-aligned. The maximum data transfer is 128 bits.

2.2.3. Single Root I/O Virtualization (SR-IOV) Design Example Simulation Testbench

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

Figure 12. Block diagram for the PCIe x16 SR-IOV Design Example Simulation Testbench



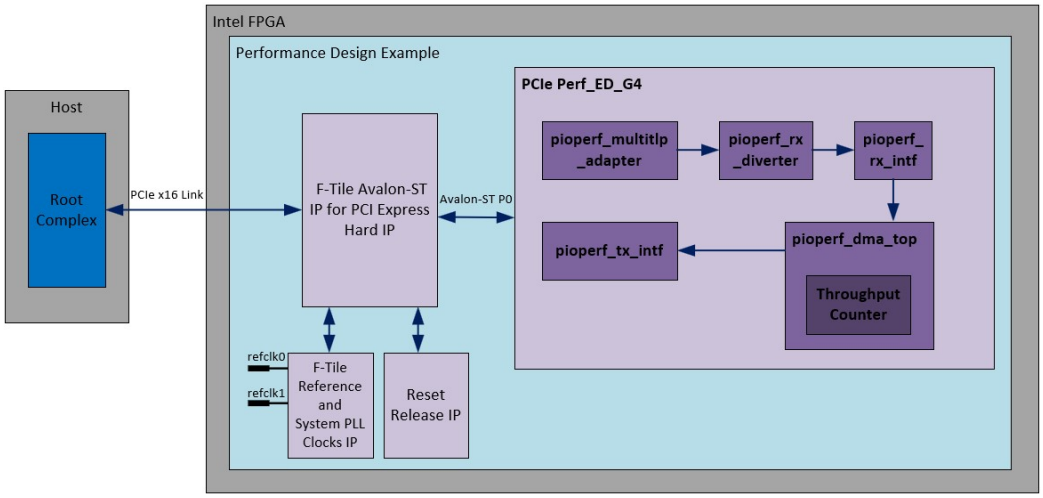
The test program writes to and reads back data from the same location in the on-chip memory across 2 PFs and 32 VFs per PF. It compares the data read to the expected result. The test reports, "Simulation stopped due to successful completion" if no errors occur. The SR-IOV design example supports Gen4 x16 Endpoint.

2.3. Performance Design Example

The Performance Design Example is designed to showcase the performance of the F-Tile Avalon-ST for PCI Express Hard IP. It can be configured to send memory write only TLPs, memory read-only TLPs or both memory write and memory read TLPs for throughput measurement. The throughput counter is implemented in the FPGA application logic to minimize the software overhead. For throughput measurement, the software application running at the host side issues a memory read TLP and acquires the throughput counter value from the control register and then prints the throughput figure at the system terminal. The software application is required to issue a memory write to the control register to stop the traffic at the end of the test.

The Performance Design Example automatically creates the files necessary to simulate and compile in the Quartus Prime software. It supports the Gen4 x16, 512-bit interface Hard IP mode. For the Agilex 7 device family, this design example supports only a 500 MHz PLD clock frequency.

Figure 13. Block Diagram for PCIe x16 Performance Design Example



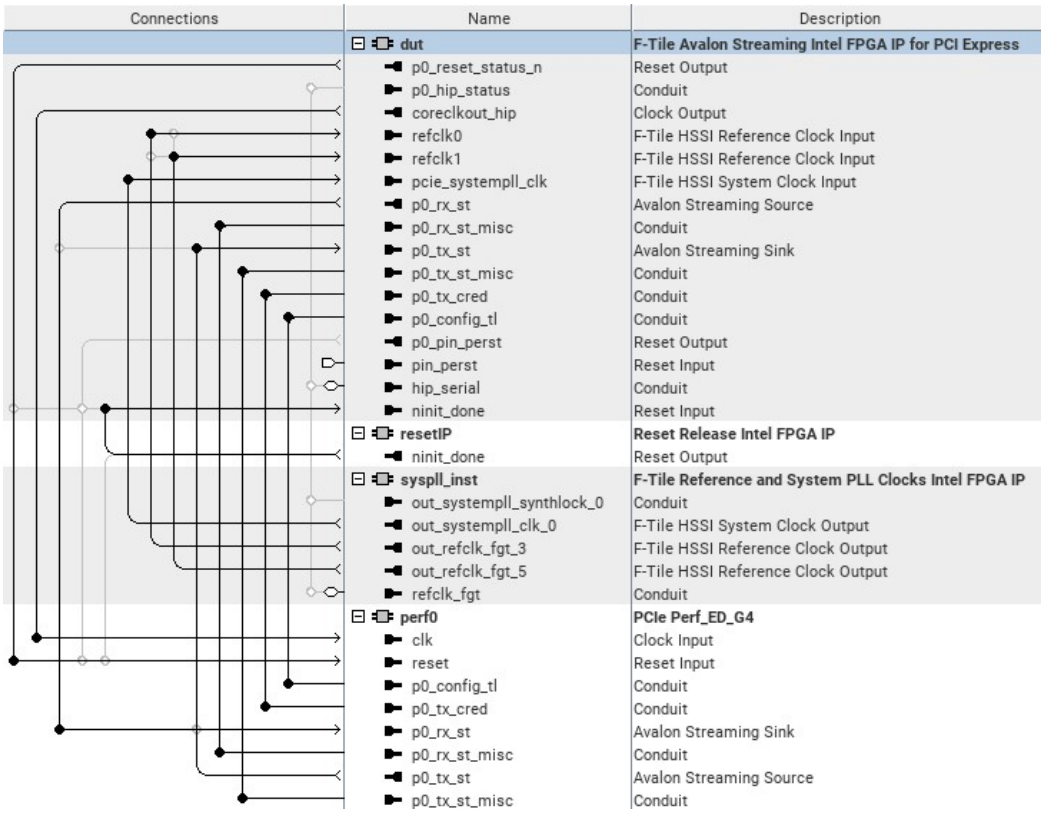
2.3.1. Performance Design Example Functional Description

1. The generated F-Tile Avalon Streaming Hard IP Endpoint variant (DUT) with the parameters you specified. This component interacts with the root complex/switch at the other end of the PCIe link and translates the data from the PCIe link into Avalon-ST data format, and vice versa.
2. The PCIe Perf_ED_G4 (perf0) component generates the requested data traffic for throughput measurement. It consists of the following sub-modules:
   a. The **pioperf_multitlp_adapter** (Avalon-ST Interface Adapter) module converts the 2-segment data of the Avalon-ST interface into two single segment streams of Avalon-ST data.
   b. The **pioperf_rx_diverter** module diverts memory write or memory read TLPs from the host and completion TLPs to their respective destinations for further processing.
   c. The **pioperf_rx_intf** (RX Interface) module decodes the TLP headers and data from the pioperf_rx_diverter module. It also extracts the information needed to construct the TLP header of the completion data such as the requester ID, tag, attribute, tc and byte count and then pass this to pioperf_tx_intf for further processing.
   d. The **pioperf_tx_intf** (TX Interface) module converts the requests from the pioperf_rx_intf and pioperf_dma_top modules into TLPs and send them to the F-Tile Avalon Streaming Hard IP. It transmits TLPs in a simplified weighted round-robin manner with this priority scheme: completion TLP > memory read TLP > memory write TLP.
   e. The **pioperf_dma_top** (DMA_TOP) module generates memory write and memory read requests and passes them to the pioperf_tx_intf module based on the information in the control register. Every memory read request will be tagged to expecting completion data before timeout to ensure data

completeness. The release of memory write and memory read TLPs will build up the traffic at the TX and RX interfaces of the PCIe link. A throughput counter is included to analyze the overall throughput of the system.

3. F-Tile Reference and System PLL Clocks IP: This IP is required for F-Tile PCIe interface implementation to configure the reference clock for the FGT PMA and System PLL. The clock from this IP is a logical connection. It is physically inside the F-Tile Avalon-ST IP for PCI Express Hard IP. There is no clock gating requirement at the design example level. The main clock of the PIO design example originates from coreclkout_hip of F-Tile Avalon-ST IP for PCI Express Hard IP running at 500 MHz. The clock originates from System PLL. This IP is required for F-Tile PCIe interface implementation to configure the reference clock for the FGT PMA and System PLL.
4. 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:
   a. To gate an external or internal reset.
   b. To gate the reset input to the transceiver and I/O PLLs.
   c. To gate the write enable of design blocks such as embedded memory blocks, state machine, and shift registers.
   d. To synchronously drive register reset input ports in your design.

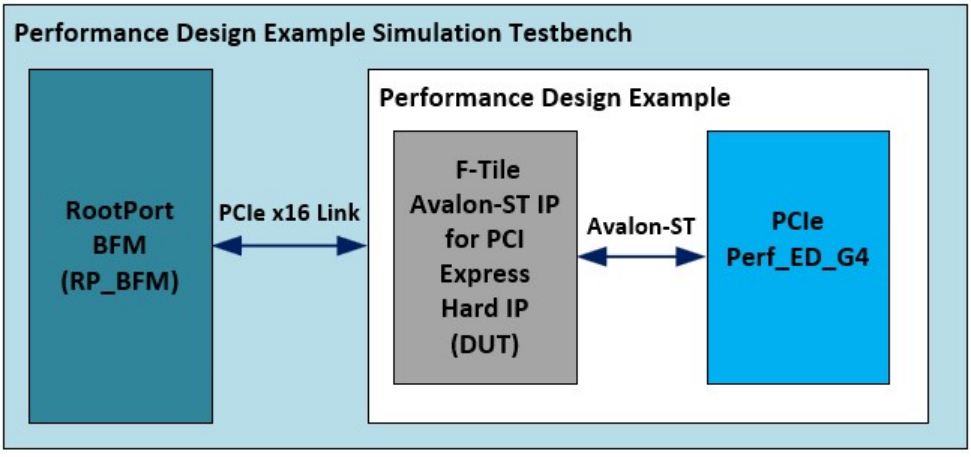
Figure 14. Platform Designer System Contents for F-Tile Avalon-ST for PCI Express x16 Performance Design Example



2.3.2. Performance Design Example Simulation Testbench

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

Figure 15. Performance Design Example Simulation Testbench

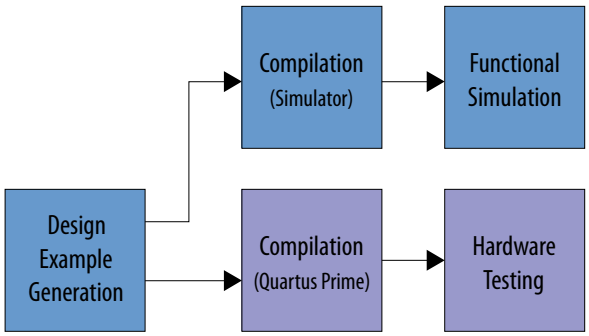


The Performance design example simulation testbench is similar to the PIO design example simulation testbench. It issues Memory Write cycles to set up and trigger the Performance design example to issue 10 Memory Writes followed by 10 Memory Reads for functional check. No throughput figure is generated from the simulation.

3. Quick Start Guide

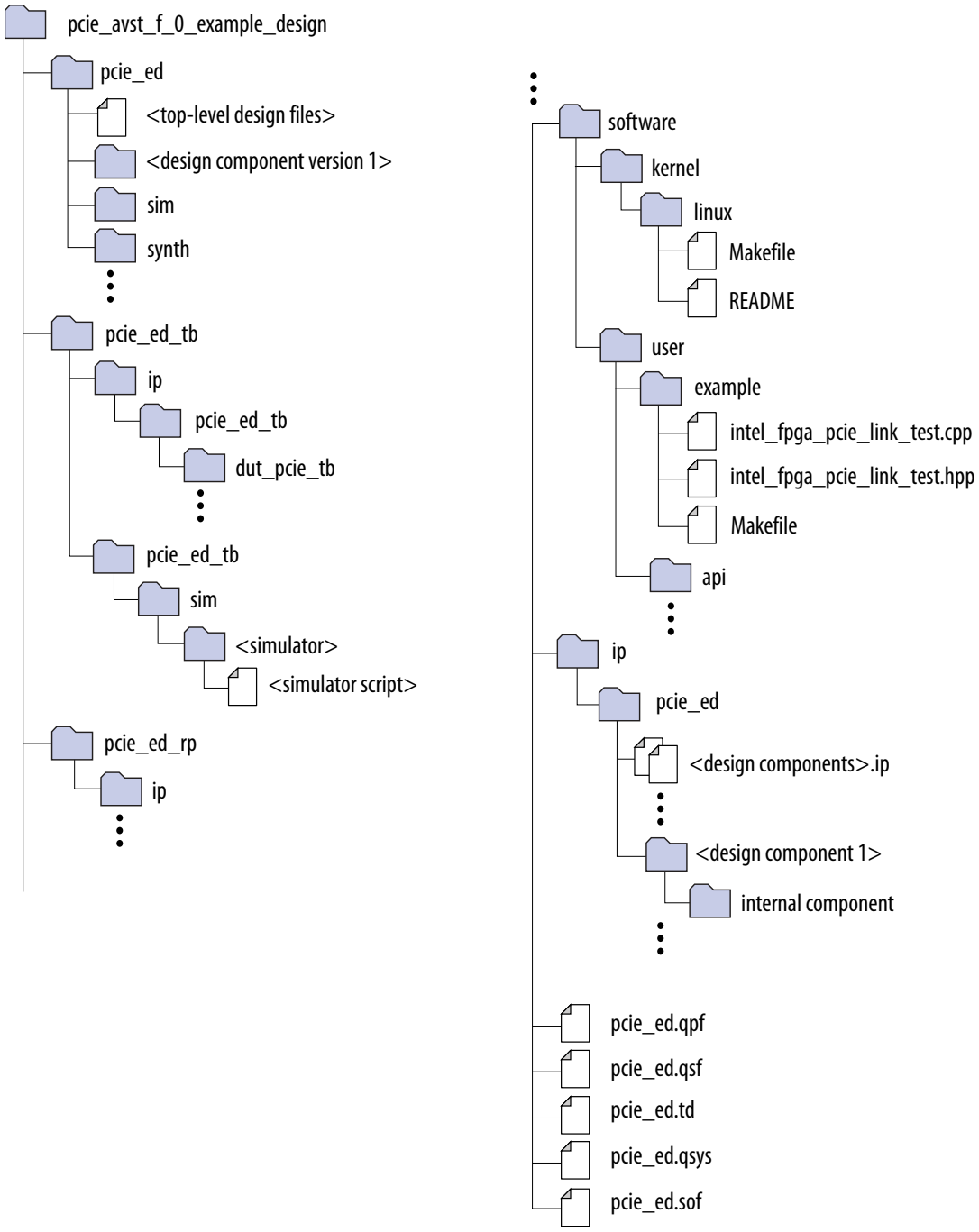
Using the Quartus Prime software, you can generate a Programmed I/O (PIO) design example or a Single Root Input/Output Virtualization (SR-IOV) design example for the F-Tile Avalon-ST Hard IP for PCI Express* IP core. The generated design example reflects the parameters that you specify. The PIO example transfers data from a host processor to a target device. This design example automatically creates the files necessary to simulate and compile in the Quartus Prime software. You can download the compiled design to your FPGA Development Board. To download to custom hardware, update the Quartus Prime Settings File (`.qsf`) with the correct pin assignments .

Figure 16. Development Steps for the Design Example



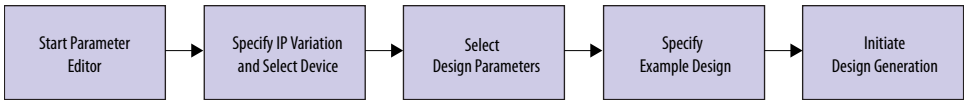
3.1. Directory Structure

Figure 17. Directory Structure for the Generated Design Example

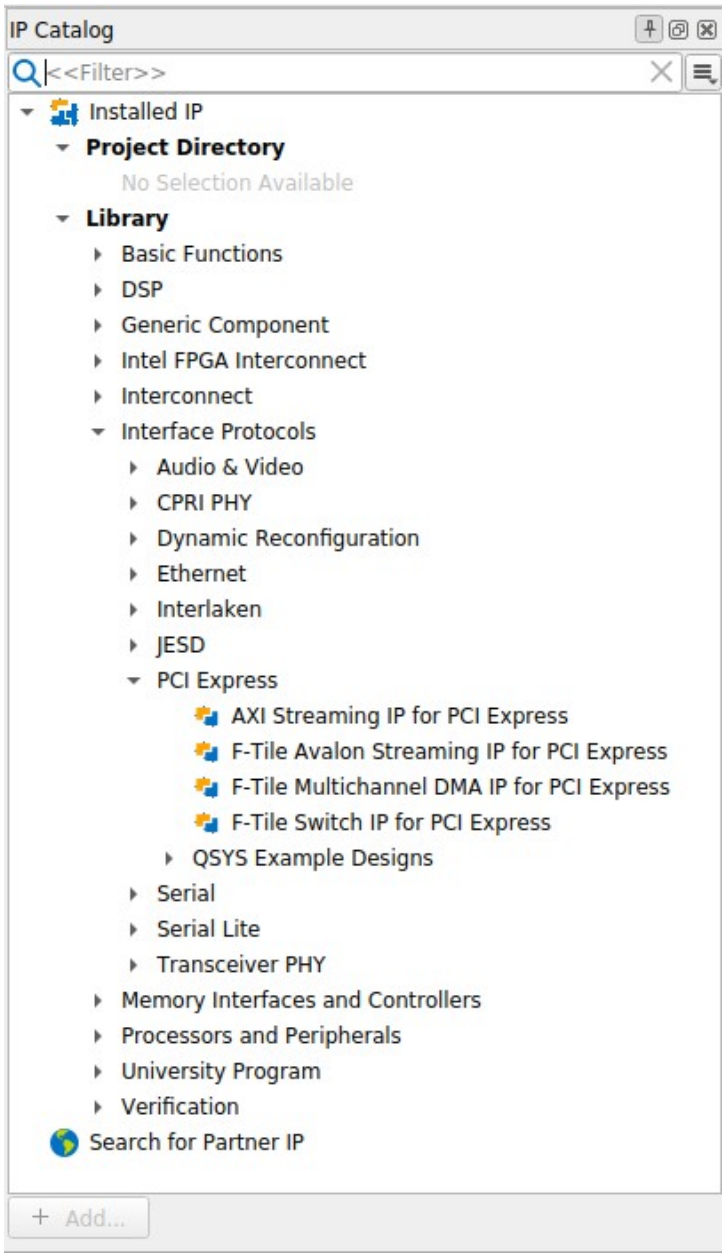


3.2. Generating the Design Example

Figure 18. Procedure



1. In the Quartus Prime Pro Edition software, create a new project (**File ➤ New Project Wizard**). Click **Next**.
2. Select the **Empty Project** type and specify the **Directory, Name**, and **Top-Level Entity**. Click **Next**.
3. For **Family, Device & Board Settings** under **Family**, select **Agilex 7** and then select **Target device** for your design. After that, click **Finish** to proceed.
4. Select **Tools → IP Catalog** to open the IP Catalog. Select the **F-Tile Avalon Streaming IP for PCI Express** (**Library → Interface Protocols → PCI Express → F-Tile Avalon Streaming IP for PCI Express**), then click **Add**.



5. In the **New IP Variant** dialogue box, specify a top-level name for your new custom IP variant and the directory for it. The IP Parameter Editor saves the IP variant settings in a file named `<your_ip>.ip`.
6. Click **Create**. The IP Parameter Editor appears as shown in the figure below. The F-Tile Avalon Streaming IP for PCI Express has **Top-Level Settings**, **PCIe0 Settings**, **Example Designs**, and **Analog Parameters** tabs to allow you to quickly configure your custom IP variant.
7. On the **Top-Level Settings** tabs, specify the parameters for your IP variant.

Note: This design example only supports the default settings in the Parameter Editor of the F-tile Avalon Streaming IP for PCIe.

If you are using the SR-IOV design example, run the following steps to enable SR-IOV:

a. At the **PCIe Settings** → **PCIe PCI Express / PCI Capabilities** → **PCIe Device** → **PCIe Multifunction and SR-IOV System Settings** tab, check the box **Enable multiple physical functions** and the **Enable SR-IOV support** option.

b. At the **PCIe MSI-X** tab under the **PCIe PCI Express / PCI Capabilities** tab, enable the **MSI-X** feature as required.

c. At the **PCIe Base Address Registers** tab under the **PCIe Settings** tab, enable **BAR0** for both PF and VF.

d. Other parameter settings are not supported for this design example.

8. 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 Agilex 7 FPGA F-Series Development Kit (Production 1 2x F-Tile)**, or **NONE** to target on the device selected for the current Quartus Prime project. If you select the development kit, the VID-related settings including the pin assignments 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.

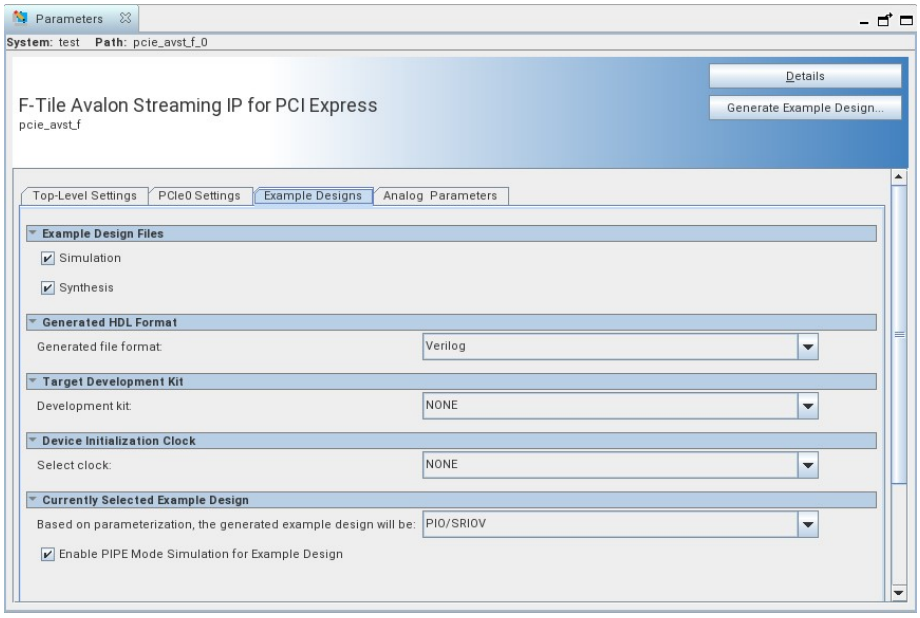
   d. For **Device Initialization Clock**, select OSC_CLK_1_125MHz when the **Agilex 7 FPGA F-Series Development Kit (Production 1 2x F-Tile)** is selected. Otherwise, the selected frequency must match the frequency you have provided for the device's OSC_CLK_1 pin of your board.

      Note: Starting with the Quartus Prime Pro Edition software version 23.4, the software enforces a check for the appropriate .qsf assignment required to constrain the device's OSC_CLK_1 pin for projects which contain transceivers in the design. A failure to provide this .qsf assignment causes the compilation to fail.

   e. For **Currently Selected Example Design**, select **PIO/SRIOV** or **PERFORMANCE_DESIGN**.

9. Click on **Generate Example Design** to generate the design example variant. When the prompt asks you to specify the directory for your design example, you can accept the default directory, `/pcie_avst_f_0_example_design`, or choose another directory. Then, click **OK** to kick off the design example generation.

Figure 19. Example Designs Tab



10. Click **Finish**. You may save your `.ip` file when prompted, but it is not required to be able to use the design example.

3.3. Simulating the Design Example

Generating tile files

The tile files are generated during the design example generation. Running Support-Logic Generation manually is not required before simulating the design example.

Tile files generation is a required step before simulation if you created your design with the F-Tile Avalon-ST IP for PCI Express from scratch. You can run **Analysis & Elaboration** on the Processing menu in the Quartus Prime Pro Edition software to generate the F-Tile specific tiles file for your design. The **Support-Logic Generation** command runs automatically as part of the process.

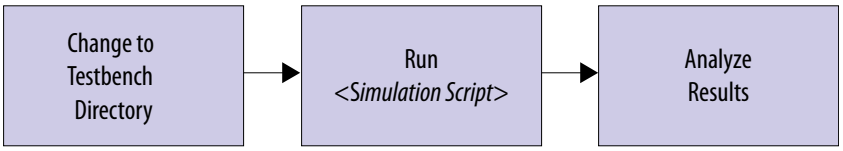
A successful tile file generation results in the **<IP_instance_name>__tiles.x** files where **x** represents the necessary file extensions. The generated files are located in your project directory and contain the full netlist for simulation and synthesis.

The following options are available for simulation time optimization:

- FASTSIM simulation mode: This mode is supported in the design examples simulation. The FASTSIM mode uses a simplified PMA abstract model and some simplified parameters driving the design to improve the overall simulation time for the F-Tile Avalon-ST IP for PCI Express. The PMA model has compile time switches **IP7581SERDES_UX_SIMSPEED** and **GDR_FASTSIM_AIB_BYPASS** to use a simplified PMA abstract model. If these switches are not defined by the compile environment, then a detailed or existing model is used.
- PIPE simulation mode: In this mode, the simulation speed is further enhanced by excluding the transceiver component where the PIPE interface of the F-Tile Avalon-ST IP for PCI Express connects to the PIPE interface of the link partner. At the IP core boundary, PIPE interface signals are produced for access to the external device when you choose the **Enable PIPE Mode Simulation** option in the F-Tile Avalon-ST IP for PCI Express Parameter Editor. Additionally, this feature provides the necessary hooks to use third-party PCI Express* VIPs/BFMs instead of the Root Port model provided with the design example. For optimal simulation performance, it is advisable to activate both FASTSIM and PIPE simulation modes. The PIPE simulation includes a compile-time switch, **gdrb_GDR_PCIE_SS_DV**, which must be defined in the compile environment when you choose the PIPE Mode Simulation option during the generation of the design example.

Note: The **Enable PIPE Mode Simulation for Example Design** option in the **Example Design** tab of the IP Parameter Editor is enabled by default for FASTSIM + PIPE mode when you generate the design example. If this option is turned off, the `+define +gdrb_GDR_PCIE_SS_DV` simulation option needs to be removed manually from the simulation command in the `run_<simulator>*` script before running it for Quartus Prime Pro Edition v25.1 and earlier versions.

Figure 20. Procedure



1. Run the simulation script under `<example_design>/pcie_ed_sim_tb/pcie_ed_sim_tb/sim/<simulator>` directory for the simulator of your choice. Refer to the table below.
2. Analyze the results.

3.3.1. Steps to Run Simulation

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

The same procedure is applicable for PCIe Gen3/4 x16, PCIe Gen3/4 x8x8 and PCIe Gen3/4 x8 design example variants.

3.3.1.1. Steps to Run Simulation : VCS

Working Directory

`<example_design>/pcie_ed_sim_tb/pcie_ed_sim_tb/sim/synopsys/vcs/`

Instructions

1. Run the following command:

```
sh run_vcs.sh
```

2. A successful simulation ends with the following message,

```
"Simulation stopped due to successful completion!"
```

in the simulation.log file that was generated.

Note: To run a simulation in interactive mode, use the following steps: (if you already generated a simv executable in noninteractive mode, delete the simv and simv.daidir)

1. Open the **vcs_setup.sh** file and add a debug option to the VCS command:

```
vcs -debug_access+all
```

2. Add the `SKIP_SIM=1` option to the command in the `run_vcsmx.sh` script before running it. The command should look like below:

```
sh vcs_setup.sh USER_DEFINED_COMPILE_OPTIONS="" USER_DEFINED_ELAB_OPTIONS="-
ignore\ initializer_driver_checks\ +vcs+lic+wait\ -full64\ -hsopt=gates\ -
debug_pp\ +define+gdrb_GDR_PCIE_SS_DV\ +define+GDR_FASTSIM_AIB_BYPASS\
+define+RTLSIM\ +define+IP7581SERDES_UX_SIMSPEED\ +define+SSM_SEQUENCE"
USER_DEFINED_SIM_OPTIONS="" SKIP_SIM=1 DEVICES_SIM_LIB_DIR=
$QUARTUS_ROOTDIR/../devices/sim_lib2   QUARTUS_SIM_LIB_DIR=
$QUARTUS_ROOTDIR/eda/sim_lib2
```

Note: The command above is a single-line command.

3. Start the simulation in interactive mode:

```
simv -gui &
```

4. Pull the signals of interest into the waveform list and click **Start** to run the simulation.
5. A successful simulation ends with the following message in the log console:

```
"Simulation stopped due to successful completion!"
```

3.3.1.2. Steps to Run Simulation : VCS MX

Working Directory

`<example_design>/pcie_ed_sim_tb/pcie_ed_sim_tb/sim/synopsys/vcsmx/`

Instructions

1. Run the following commands:

```
sh run_vcsmx.sh
```

2. A successful simulation ends with the following message,

```
"Simulation stopped due to successful completion!"
```

in the simulation.log file that was generated.

Note: To run a simulation in interactive mode, use the following steps: (if you already generated a simv executable in noninteractive mode, delete the simv and simv.daidir)

1. Open the **vcsmx_setup.sh** file and add a debug option to the VCS command:

```
vcs -debug_access+all
```

2. Add the `SKIP_SIM=1` option to the command in the `run_vcsmx.sh` script before running it. The command should look like below:

```
sh vcsmx_setup.sh USER_DEFINED_COMPILE_OPTIONS="-kdb\ +define
+gdrb_GDR_PCIE_SS_DV\ +define+GDR_FASTSIM_AIB_BYPASS\  +define+RTLSIM\
+define+SSM_SEQUENCE\ -ignore\ initializer_driver_checks\ -sverilog\ +define
+IP7581SERDES_UX_SIMSPEED" USER_DEFINED_ELAB_OPTIONS="+vcs+lic+wait\ -
full64\ -ignore\ initializer_driver_checks\ -hsopt=gates\ -debug_pp\ -
debug_access+all" USER_DEFINED_SIM_OPTIONS="" SKIP_SIM=1 DEVICES_SIM_LIB_DIR=
$QUARTUS_ROOTDIR/../devices/sim_lib2   QUARTUS_SIM_LIB_DIR=
$QUARTUS_ROOTDIR/eda/sim_lib2
```

Note: The command above is a single-line command.

3. Start the simulation in interactive mode:

```
simv -gui &
```

4. Pull the signals of interest into the waveform list and click **Start** to run the simulation.
5. A successful simulation ends with the following message in the log console:

```
"Simulation stopped due to successful completion!"
```

3.3.1.3. Steps to Run Simulation : Questa*

Working Directory

`<example_design>/ pcie_ed_sim_tb/pcie_ed_sim_tb/sim/mentor/`

Instructions

1. Run the following command from the working directory: `vsim -do run_msim.tcl`
2. A successful simulation ends with the following message:

```
"Simulation stopped due to successful completion!"
```

Note: Due to a problem in the Quartus Prime Pro Edition Software version 25.3, you are required to add "`-suppress 2732`" to USER_DEFINED_COMPILE_OPTIONS and "`-suppress 10000 `" to USER_DEFINED_ELAB_OPTIONS in the `run_msim.tcl` simulation script manually for a successful simulation with the Questa simulator.

3.3.1.4. Steps to Run Simulation : Xcelium

Working Directory

`<example_design>/ pcie_ed_sim_tb/pcie_ed_sim_tb/sim/xcelium/`

Instructions

1. Run the following command from the working directory: `sh run_xcelium.sh`.
2. A successful simulation ends with the following message in the simulation.log file that was generated.

```
"Simulation stopped due to successful completion!"
```

3.3.1.5. Steps to Run Simulation : Riviera-PRO

Simulation Flow starting with Quartus Prime 22.4 version

1. Generate the example design
2. Follow the steps below:
   a.
   ```
   cd <my_design>/pcie_ed_sim_tb/pcie_ed_sim_tb/sim/aldec
   ```
   b. Run the following command from the working directory:
   ```
   vsim -do run_riviera.tcl
   ```
   c. A successful simulation ends with the following message:
   ```
   "Simulation stopped due to successful completion!"
   ```

Note: Due to a problem in the Quartus Prime Pro Edition Software version 25.3, a design example simulation with Riviera-Pro may run into a fatal error. This problem is planned to be fixed in a future release.

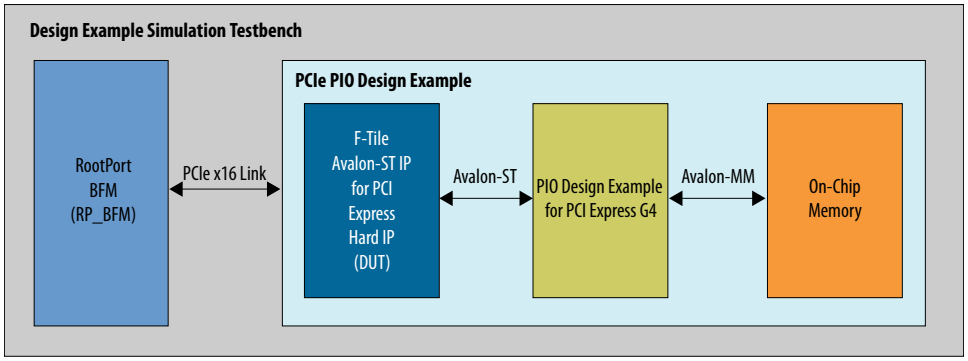
3.3.2. Testbench

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 F-Tile IP for PCIe. The testbench uses the parameters that you specify in the Parameter Editor in Quartus Prime.

This testbench simulates up to a x16 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 21. 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>/pcie_avst_f_0_example_design/pcie_ed_sim_tb/ip/
pcie_ed_sim_tb/dut_pcie_tb_ip/intel_pcie_ftile_tbed_<ver>/sim
```

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

```
//Directory path
<project_dir>/pcie_avst_f_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>/pcie_avst_f_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>/pcie_avst_f_0_example_design/ip/pcie_ed
```

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 *F-Tile Avalon Streaming IP for PCI Express User Guide*.

3.3.2.1. Test Driver Module

The test driver module, `intel_pcie_ftile_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.

3.3.2.2. 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 22. PIO Design Example Simulation Design Hierarchy

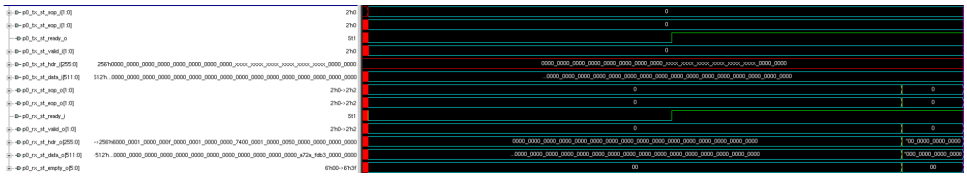
Hierarchy	Module
pcie_ed_tb	pcie_ed_tb
dut_pcie_tb	dut_pcie_tb_ip
dut_pcie_tb	intel_pcie_ftile_tbed_hwtcl
tile	pcie_ed_tiles
g_bfm	
p_dut_ep	
altpcietb_bfm_top_rp	altpcietb_bfm_top_rp
bfm_log_common	altpcietb_bfm_log_common
bfm_req_intf_common	altpcietb_bfm_req_intf_common
bfm_shmem_common	altpcietb_bfm_shmem_common
ltssm_mon	altpcietb_ltssm_mon
rst_clk_gen	altpcietb_rst_clk
g_bfm	
g_ser_pipen	
pcie_ed_inst	pcie_ed
dut	pcie_ed_dut
mem0	pcie_ed_MEM0
mm_interconnect_0	pcie_ed_altera_mm_interconnect_1920_ljfibnq
pio0	pcie_ed_pio0
resetip	pcie_ed_resetIP
rst_controller	altera_reset_controller
syspll_inst	pcie_ed_syspll_inst
pcie_ed_inst_ninit_done_bfm	pcie_ed_inst_ninit_done_bfm_ip
pcie_ed_inst_p0_config_tl_bfm	pcie_ed_inst_p0_config_tl_bfm_ip
pcie_ed_inst_p0_pin_perst_bfm	pcie_ed_inst_p0_pin_perst_bfm_ip
pcie_ed_inst_p0_tx_cred_bfm	pcie_ed_inst_p0_tx_cred_bfm_ip

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.

Trace the transition of signal `p0_rx_st_sop_o[0]` (for example h'0 to h'1) for the first memory write. It is followed by a memory read at the Avalon-ST RX interface of the F-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 23. Simulation Waveforms for the PIO Design Example for the F-Tile Avalon-ST IP for PCIe

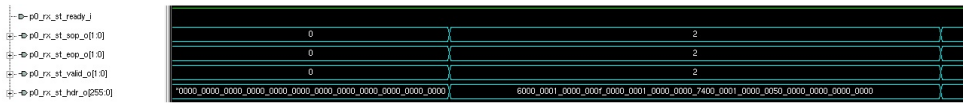


3.3.2.2.1. Configure the Testbench to Perform a Dword Address-Aligned Read and Write Operation

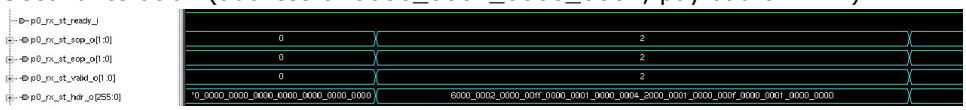
The simulation testbench from the Programmed Input/Output (PIO) design example can be modified to demonstrate Double Word (DW) address alignment through 16 iterations of Memory Write (MWr) and Memory Read (MRd) operations. The initial address is set to 0x00, with a payload size of 1 DW. In each iteration, both the Transaction Layer Packet (TLP) payload and the request address increment by 1 DW. The data retrieved from MRd operations are validated against the corresponding MWr data. Any mismatches identified during the comparison are logged as errors in the simulation logfile.

The waveform shows the increment of the address and payload size.

First iteration (address of 0000_0001_0000_0000, payload of 1DW)



Second iteration (address of 0000_0001_0000_0004, payload of 2DW)



1. Locate the file `altpcietb_bfm_rp_gen4_x16.sv` in the `<design example directory>/pcie_ed_sim_tb/ip/pcie_ed_sim_tb/dut_pcie_tb_ip/intel_pcie_ftile_tbed_100/sim/` directory.
2. Modify the following localparam in the `altpcietb_bfm_rp_gen4_x16.sv` file as shown below:

   Line 6037: RCSLAVE_MAXLEN=16;

   Note: The RCSLAVE_MAXLEN is to control the loop count of the MWr/MRd instruction routines.
3. Task downstream_read is responsible for the MWr/MRd instruction routines. It initiates the MWr and MRd based on the variable passed in.

a. Insert "Iepmem_address = epmem_address;" after line 7407.
b. Comment out "Ibyte_length=4; at line 7411.
c. Rename lines 7414 and 7421 from "epmem_address;" to "Iepmem_address,".
d. Uncomment line 7424 to enable a delay of #1000000.
e. Insert "Iepmem_address = Ibyte_length;" after line 7429.

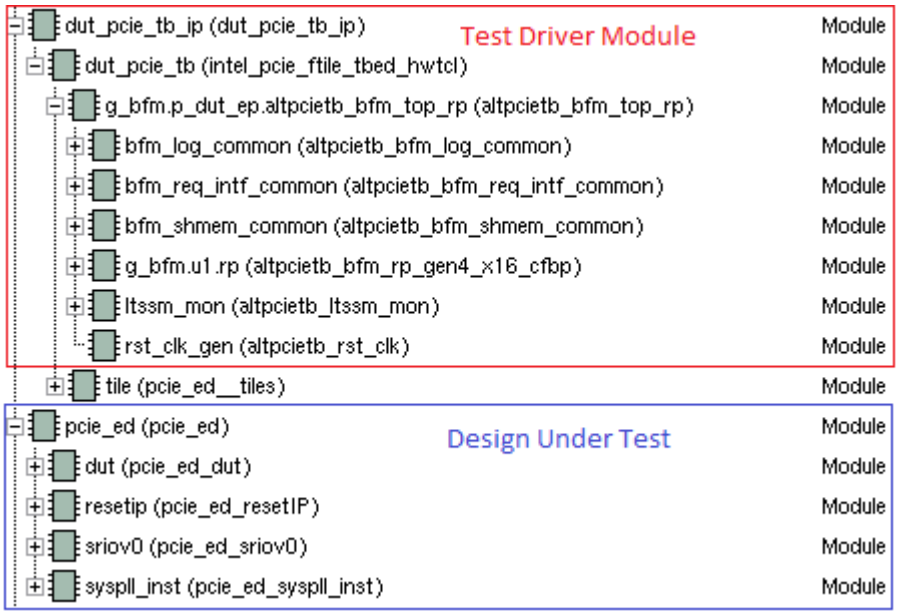
```
7407    Iepmem_address = epmem_address;
7408        for (i=0;i<loop;i=i+1) begin
7409            //Comment out next line to open up TLP length
7410        //TODO extend to more than 1 DW
7411            //Ibyte_length=4;
7412            downstream_write( bar_table,
7413                              setup_bar,
7414                              Iepmem_address,
7415                              Istart_val,
7416                              Ibyte_length);
7417        // HSD: 1509833345 - larger the gap between memrd and memwr to solve 2x8 gen 3 hung issue
7418            #10000
7419            downstream_read ( bar_table,
7420                              setup_bar,
7421                              Iepmem_address,
7422                              Ibyte_length);
7423            //testing to add 1 clock cycle of time before the memory check because there is racing condition
7424            #1000000
7425            $display ("KLAI4_top_scr_mem_comp Ibyte_length= %0d", Ibyte_length);
7426            scr_memory_compare(Ibyte_length,
7427                               SCR_MEM_DOWNSTREAM_WR,
7428                               SCR_MEM_DOWNSTREAM_RD);
7429            Istart_val    = Istart_val+cfg_maxpload_byte;
7430            Iepmem_address = Ibyte_length;
```

4. The successful run should have 16 iterations of MWr and MRd. The logfile should contain the message "successful completion!"

3.3.2.3. 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 24. SR-IOV Design Example Simulation Design Hierarchy



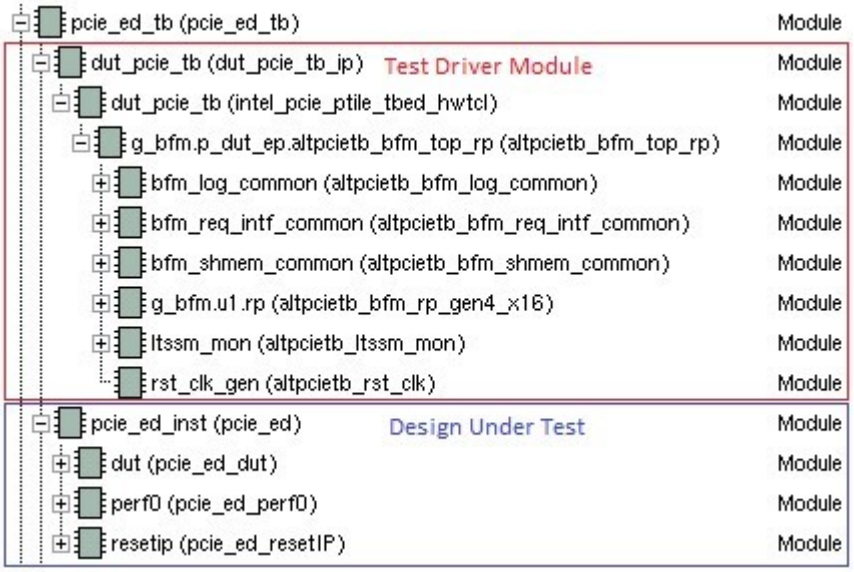
The SR-IOV testbench supports up to two Physical Functions (PFs) and 32 Virtual Functions (VFs) per PF. The testbench starts with link training and then accesses the configuration space of the IP for enumeration. After that, it performs the following steps:

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.

3.3.2.4. Performance Design Example Testbench

The figure below shows the Performance design example simulation design hierarchy. The tests for the Performance 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 `perf_ed_dma_write`/`perf_ed_dma_read`.

Figure 25. Performance Design Example Simulation Design Hierarchy



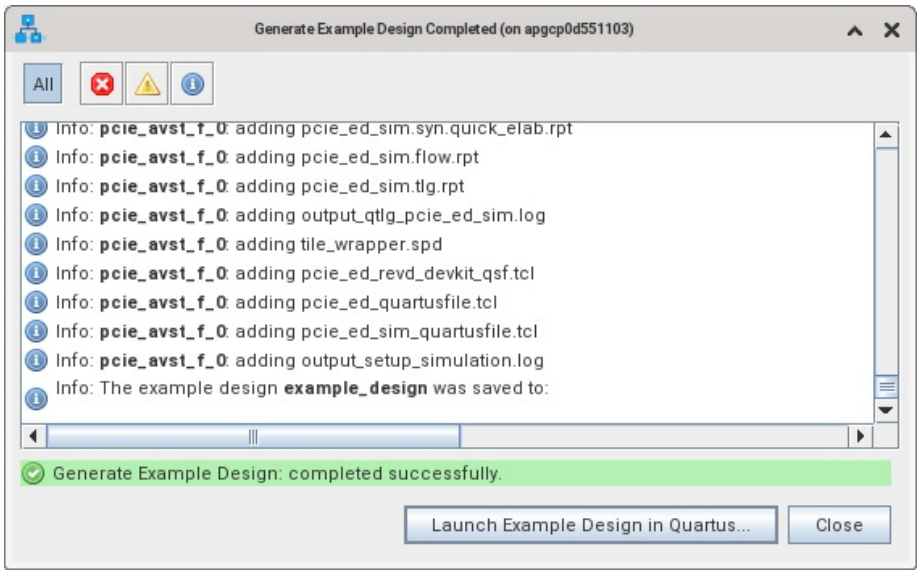
The testbench starts with link training and then accesses the configuration space of the IP for enumeration. A task called `perf_ed_dma_write` and `perf_ed_dma_read` (defined in the Root Port PCIe BFM **altpcietb_bfm_rp_gen4_x16.sv**) then performs the PCIe link test. This test consists of the following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

1. Issue a memory write command to set up the Performance design example's target memory write address.
2. Issue a memory write command to trigger the Performance design example to send 10 memory writes with 128 bytes in length.
3. Issue a memory write command to set up the Performance design example's target memory read address.
4. Issue a memory write command to trigger the Performance design example to send 10 memory reads with 128 bytes in length.

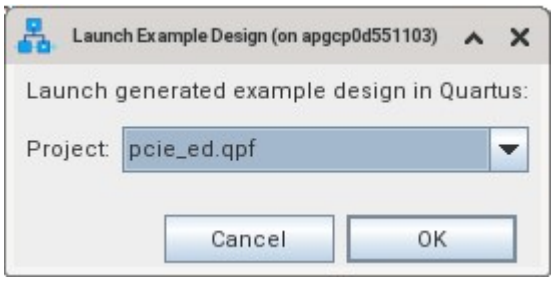
Trace the transition of signal `p0_rx_st_sop_o[0]` (for example h'0 to h'1) for the first memory write. The 10 memory writes and 10 memory reads with completions from the RP BFM appear shortly after the memory write requests at the Avalon-ST RX interface.

3.4. Compiling the Design Example

1. After the design example generation is completed, click **Launch Example Design in Quartus**.



2. Select `pcie_ed.qpf` and click **OK** to open the newly generated design example in the Quartus Prime Pro Edition software.



3. In the Quartus Prime Pro Edition software, click **Processing → Start Compilation** to compile the design example and generate the bitstream (.sof) file.
4. Examine the design compilation result like resource utilization and timing result.
5. Close your example design project.

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

3.5. Hardware and Software Requirements

Below are the hardware and software used to generate the test result as shown in Running the Design Example on page 41.

Hardware:

- Intel Ice Lake Server with 8 DDR4-3200 8GB RDIMMs installed as the host system
- Agilex 7 F-Series F-Tile FPGA Development Kit

Note: The Agilex 7 F-Series F-Tile FPGA Development Kit is configured to use the local 100MHz clock source for PCIe (SW4.3 is set to OFF) by default. You may observe an unstable PCIe link when testing the design example on hardware if your host system has the Spread Spectrum Clocking (SSC) feature enabled by default. The common clock source is recommended to get around the problem. This can be done by changing SW4.3 to ON to use the clock source from the host system,

Software:

- Quartus Prime Pro Edition Software version 25.3 and later
- Linux distribution with Kernel version up to 5.15
- Ubuntu 22.04.3 LTS (Kernel: 5.15.0-97-generic).
- Software driver generated along with the design example

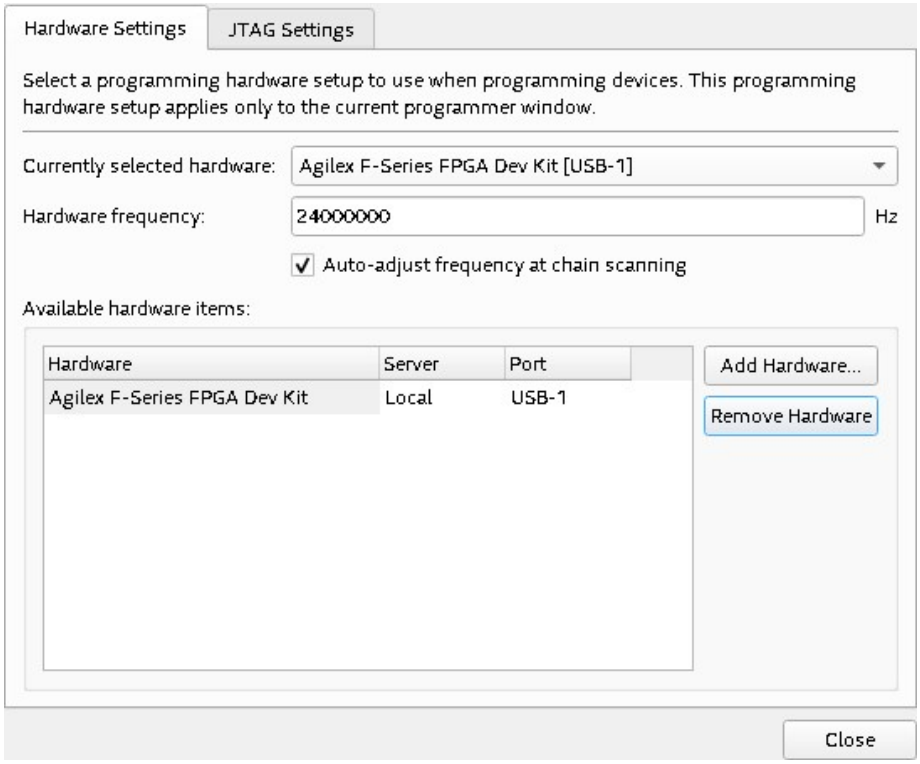
3.6. Program the FPGA

Prerequisite: Generate and compile the example design in the Quartus Prime Pro Edition software before starting to test the design example on hardware.

This section describes how to configure the Agilex 7 F-Series F-Tile FPGA Development Kit.

1. Install the Agilex 7 F-Series F-Tile FPGA Development Kit into a PCIe Gen4 x16 slot on the host system, connect the ATX 6-pins power supply.
2. Connect the Agilex 7 F-Series F-Tile FPGA Development Kit to a computer system on which the Quartus Prime Pro Edition software is installed using the USB cable shipped along with the development kit for FPGA configuration.
3. Power on the host system and turn on the power switch on the development kit.
4. In the Quartus Prime Pro Edition software, invoke the programmer by clicking **Tools** > **Programmer**.
5. In the Programmer, click **Hardware Setup** and verify the Agilex 7 F-Series F-Tile FPGA Development Kit is detected in the **Hardware Settings** tab.

Figure 26. Hardware Settings

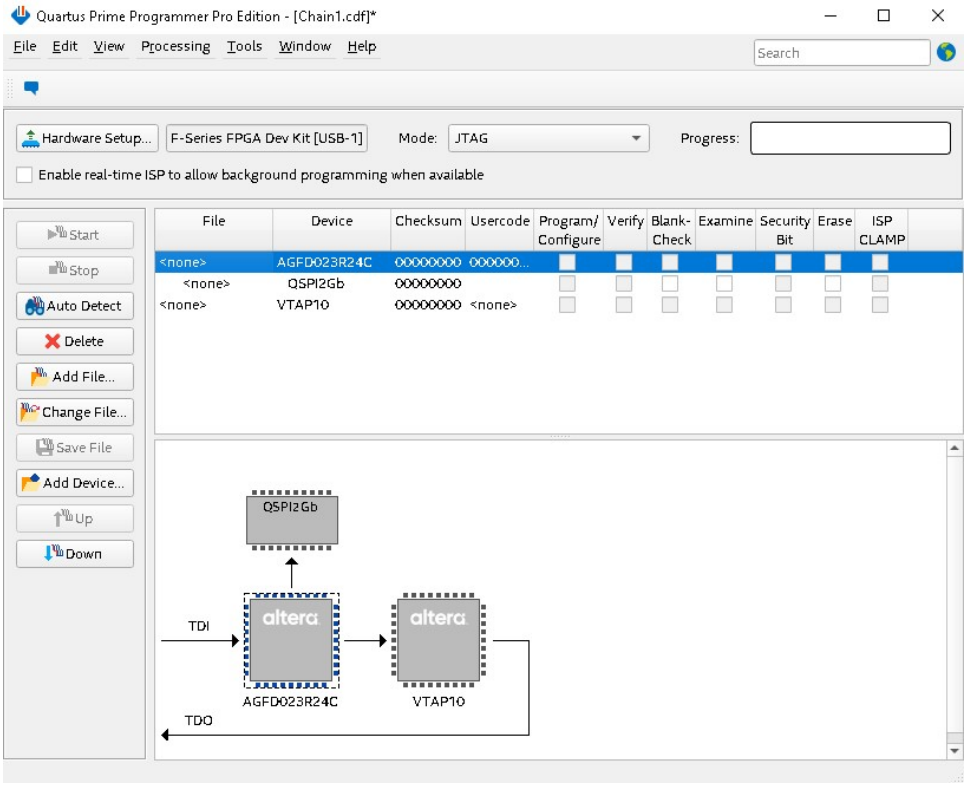


6. For **Currently selected hardware**, select the Agilex 7 F-Series F-Tile FPGA Development Kit and then click **Close**.
7. Click **Auto Detect** to detect the JTAG device chain.
8. Select the target FPGA device in the JTAG chain, click **Change File**, and select the FPGA configuration file, pcie_ed.sof. Then, click **Open**.

9. Check the **Program/Configure** option, and click **Start** to start the FPGA configuration.
10. Perform a warm reboot for the host system once the Agilex 7 FPGA is successfully configured.
11. Check the enumeration of the PCIe Endpoint device (Agilex 7 F-Series F-Tile FPGA Development Kit) on the host system by running the "lspci –d 1172:" command in a Linux Terminal.

    Expected result:

    BDF Unassigned class [ff00]: Altera Corporation Device 0000 (rev 01)



3.7. Installing the Linux Kernel Driver

Before you can test the design example in hardware, you must configure the FPGA and then restart your computer to allow the enumerate to take place for the PCIe device. After that, 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

Note: Throughout this user guide, the term word DWORD has the same meaning as in the PCI Express Base Specification. A word is 16 bits and a DWORD is 32 bits.

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. In the host system, navigate to the `./software/kernel/linux` directory of the software driver.
2. Change the permissions on the install, load, and unload files

```
$ sudo chmod 777 install load unload
```

3. Install the driver

```
$ sudo ./install
```

4. Verify the driver installation

```
$ sudo lsmod | grep intel_fpga_pcie_drv
```

Expected Result

```
intel_fpga_pcie_drv 17792 0
```

5. Verify that Linux recognizes the PCIe design example

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

Expected result. Kernel driver in use:

```
intel_fpga_pcie_drv
```

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

3.8. Running the Design Example

Table 3. Test Operations Supported by the F-Tile Avalon-ST IP for PCI Express Design Examples

Operations	Required BAR	Supported by F-Tile Avalon Streaming IP for PCIe Design Examples		
		PIO	SR-IOV	Performance
0: Link test - 100 writes and reads	0	Yes	Yes	No
1: Write memory space	0	Yes	Yes	No
2: Read memory space	0	Yes	Yes	No
3: Write configuration space	N/A	No	No	No
4: Read configuration space	N/A	No	No	No

continued...

Operations	Required BAR	Supported by F-Tile Avalon Streaming IP for PCIe Design Examples		
		PIO	SR-IOV	Performance
5: Change BAR for PIO	N/A	Yes	Yes	No
6: Change device	N/A	Yes	Yes	No
7: Enable SR-IOV	N/A	No	Yes	No
8: Do a link test for every enabled virtual function belonging to the current device	N/A	No	Yes	No
9: Perform DMA for Throughput	0	No	No	Yes
10: Quit program	N/A	Yes	Yes	Yes

3.8.1. Running the PIO Design Example

1. In the host system, navigate to the `./software/user/example` directory of the software driver.
2. Compile the design example software application by running the make command: `$ make`
3. Run the test by running the following command: `$ sudo ./intel_fpga_pcie_link_test`

   You can run the 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 Altera 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 Agilex 7 Development Kit, you can determine the BDF by typing the following command:

   ```
   $ sudo lspci -d 1172:
   ```

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

Figure 27. Automatic Mode

```
************************************************************
Intel FPGA PCIe Link Test
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 0x8a00

************************************************************
 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 for PIO
 6: Change device
 7: Enable SRIOV
 8: Do a link test for every enabled virtual function
    belonging to the current device
 9: Perform DMA for Throughput
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
```

Figure 28. 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:
> 8a
Enter device number, in hex:
> 0
Enter function number, in hex:
> 0
BDF is 0x8a00
B:D.F, in hex, is 8a:0.0
Enter BAR number (-1 for none):
> 0
Opened a handle to BAR 0 of a device with BDF 0x8a00

************************************************************
 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 for PIO
 6: Change device
 7: Enable SRIOV
 8: Do a link test for every enabled virtual function
    belonging to the current device
 9: Perform DMA for Throughput
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
```

3.8.2. Running the SR-IOV Design Example

Before you can test the design example in hardware, you must enable SR-IOV and ARI (Alternative Routing ID Interpretation) in the BIOS, configure the FPGA, and then restart your computer to allow the enumeration to take place for the PCIe device. After that, install the Linux kernel driver as described in Installing the Linux Kernel Driver.

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

1. In the host system, navigate to the `./software/user/example` directory of the software driver.
2. Compile the design example software application by running the make command: `$ make`.
3. Run the test by running the following command: `$ sudo ./intel_fpga_pcie_link_test`.
4. Select **Option 1: Manually Select a Device** to specify the Bus-Device-Function (BDF) of the Physical Function (PF) to which the Virtual Functions (VFs) are assigned.
5. Enter BAR "0" to proceed to the test menu.
6. Enter option 7 to enable SR-IOV for the current device.
7. Enter the number of virtual functions to be enabled for the current device.

Figure 29. Link Test Menu

```
*********************************************************
Intel FPGA PCIe Link Test
Version 2.0
0: Automatically select a device
1: Manually select a device
*********************************************************
> 1
Enter bus number, in hex:
> 8a
Enter device number, in hex:
> 0
Enter function number, in hex:
> 0
BDF is 0x8a00
B:D.F, in hex, is 8a:0.0
Enter BAR number (-1 for none):
> 0
Opened a handle to BAR 0 of a device with BDF 0x8a00

*********************************************************
 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 for PIO
 6: Change device
 7: Enable SRIOV
 8: Do a link test for every enabled virtual function
    belonging to the current device
 9: Perform DMA for Throughput
10: Quit program
********************************************************* |

> 7
Enter the number of VFs to enable for the current device:
> 32
Enabled 32 VFs.
Type 'lspci -d 1172:' in a new terminal to determine newly enabled devices
' BDFs.
```

Note: To disable VF for the current device, repeat Step 6: Select option 7 to enable SR-IOV, enter 0 for the number of VFs to be enabled.

Note: Type '`lspci -d 1172:`' in a new terminal to determine the BDFs of the newly enabled VFs.

8. In a new terminal, run the

```
sudo 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. In this example, there are 2 PFs and 32 VFs are enabled for each PF. Running this command returns 66 as shown in the figure below.

Figure 30. lspci command

```
[root@WilsonCity example]# lspci -d 1172: | grep -c "Altera"
66
```

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

Figure 31. Test Result

```
Number of write errors:       0
Number of read errors:        0
Number of dword mismatches:   0
Testing VF with BDF 0x4b1e...
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 0x4b1f...
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 0x4b20...
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 0x4b21...
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 32 VFs
```

3.8.3. Running the Performance Design Example

Note: The Performance Design Example hardware test is currently supported at 500 MHz only.

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 FPGA IP PCIe link test application in manual or automatic mode. Select from:

- In automatic mode, the application automatically selects the device. The test selects the Altera 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 Agilex 7 Development Kit, you can determine the BDF by typing the following command:

```
$ sudo lspci -d 1172:
```

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

Figure 32. Automatic Mode

```
************************************************************
Intel FPGA PCIe Link Test
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 0x8a00
```

Figure 33. 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:
> 8a
Enter device number, in hex:
> 0
Enter function number, in hex:
> 0
BDF is 0x8a00
B:D.F, in hex, is 8a:0.0
Enter BAR number (-1 for none):
> 0
Opened a handle to BAR 0 of a device with BDF 0x8a00
```

5. To perform DMA for throughput test, enter option "9" and press Enter key to proceed.
6. Enter number of DMA iterations to be carried out and then press Enter key to select the type of DMA operation, "0" for write operation (device to host), "1" for read operation (host to device) and "2" for simultaneous write and read operations.
7. Select payload size for the TLPs used for the throughput test.

Here are block diagrams and sample test results for write only, read only and simultaneous write and read for 10 iterations respectively.

Figure 34. Write Only Operation

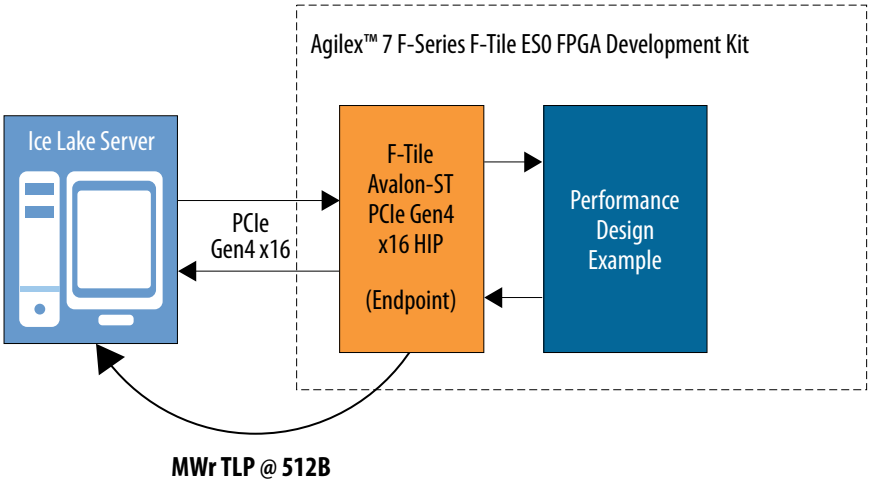


Figure 35. Write Only Test Result

```
************************************************************
 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 for PIO
 6: Change device
 7: Enable SRIOV
 8: Do a link test for every enabled virtual function
    belonging to the current device
 9: Perform DMA for Throughput
10: Quit program
************************************************************
> 9
Enter number of loop (between 1-1000; 1 second per loop):
> 10
Enter type of operation (0 for write,1 for read and 2 for simultaneous write and read):
> 0
Choose payload size:
1. 16B
2. 32B
3. 64B
4. 128B
5. 256B
6. 512B
> 6
DMA WRITE only (MemWr)

************************************************************
PERF WRITE GB/s:    29.9788
PERF WRITE GB/s:    29.9788
PERF WRITE GB/s:    29.9683
PERF WRITE GB/s:    29.9683
PERF WRITE GB/s:    29.9788
PERF WRITE GB/s:    29.9788
PERF WRITE GB/s:    29.9683
PERF WRITE GB/s:    29.9788
PERF WRITE GB/s:    29.9788
PERF WRITE GB/s:    29.9788

************************************************************
```

Figure 36. Read Only Operation

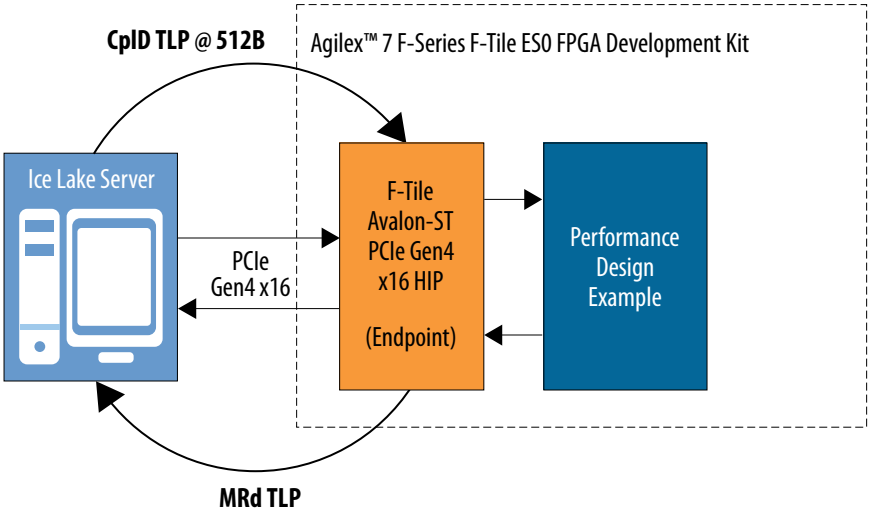


Figure 37. Read Only Test Result

```
************************************************************
 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 for PIO
 6: Change device
 7: Enable SRIOV
 8: Do a link test for every enabled virtual function
    belonging to the current device
 9: Perform DMA for Throughput
10: Quit program
************************************************************
> 9
Enter number of loop (between 1-1000; 1 second per loop):
> 10
Enter type of operation (0 for write,1 for read and 2 for simultaneous write and read):
> 1
Choose payload size:
1. 16B
2. 32B
3. 64B
4. 128B
5. 256B
6. 512B
> 6
DMA READ only (MemRd)

************************************************************
PERF READ GB/s:     29.119
PERF READ GB/s:     29.119
PERF READ GB/s:     29.119
PERF READ GB/s:     29.119
PERF READ GB/s:     29.119
PERF READ GB/s:    29.1399
PERF READ GB/s:    29.1294
PERF READ GB/s:    29.1294
PERF READ GB/s:    29.1294
PERF READ GB/s:    29.1294

************************************************************
```

Figure 38. Simultaneous Write and Read Operation

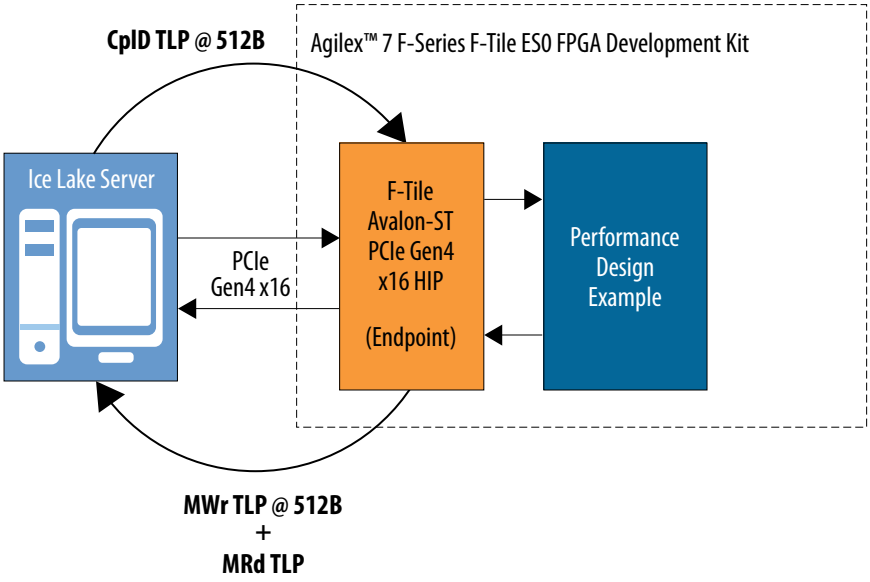


Figure 39. Simultaneous Write and Read Test Result

```
************************************************************
 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 for PIO
 6: Change device
 7: Enable SRIOV
 8: Do a link test for every enabled virtual function
    belonging to the current device
 9: Perform DMA for Throughput
10: Quit program
************************************************************
> 9
Enter number of loop (between 1-1000; 1 second per loop):
> 10
Enter type of operation (0 for write,1 for read and 2 for simultaneous write and read):
> 2
Choose payload size:
1. 16B
2. 32B
3. 64B
4. 128B
5. 256B
6. 512B
> 6
DMA READ & WRITE

************************************************************
PERF WRITE GB/s:    28.5422
PERF READ GB/s:     28.018
PERF WRITE GB/s:    28.5318
PERF READ GB/s:     28.018
PERF WRITE GB/s:    28.5422
PERF READ GB/s:    28.0075
PERF WRITE GB/s:    28.5422
PERF READ GB/s:    28.0075
PERF WRITE GB/s:    28.5422
PERF READ GB/s:    28.0284
PERF WRITE GB/s:    28.5422
PERF READ GB/s:    28.0284
PERF WRITE GB/s:    28.5318
PERF READ GB/s:     28.018
PERF WRITE GB/s:    28.5422
PERF READ GB/s:    28.0284
PERF WRITE GB/s:    28.5318
PERF READ GB/s:     28.018
PERF WRITE GB/s:    28.5318
PERF READ GB/s:    28.0284

************************************************************
```

Performance Design Example Data Throughput

The following test result is obtained using the hardware and software as mentioned in Hardware and Software Requirements on page 38.

The data throughput values may vary especially when a different host system is used.

Table 4. Data Throughput

TLP Payload Size (Byte)	Data Throughput			
	Memory Write (GB/s)	Memory Read (GB/s)	Simultaneous Write and Read	
			Write (GB/s)	Read (GB/s)
64	22.80	21.08	17.86	17.63
128	26.40	25.67	22.72	22.38
256	28.69	27.85	26.31	25.88
512	29.97	29.12	28.02	28.54

Note: Data throughput at 16 and 32 bytes TLP payload size does not meet expected throughput values. It may be addressed in the future Quartus Prime software release.

4. F-Tile Avalon Streaming IP for PCI Express Design Example User Guide Archives

For the latest and previous versions of this document, refer to F-Tile Avalon Streaming IP for PCI Express Design Example User Guide. If an IP or software version is not listed, the user guide for the previous IP or software version applies.

5. Revision History for the F-Tile Avalon Streaming IP for PCI Express Design Example User Guide

Document Version	Quartus Prime Version	IP Version	Changes
2026.02.10	25.3	12.4.0	• Added two variants to the table in the *Design Example Description* section. • Removed an outdated Note in the *Configure the Testbench to Perform a Dword Address-Aligned Read and Write Operation* section.
2025.10.20	25.3	12.4.0	• Updated the table of supported design examples in *Design Example Description*. • Updated the steps in *Generating the Design Example*. • Added the sub-section *Configure the Testbench to Perform Dword Address Aligned Read and Write Operation* under the *PIO Design Example Testbench*. • Updated the steps in *Compiling the Design Example*. • Updated the *Hardware and Software Requirements* section. • Added the *Program the FPGA* section to the *Quick Start Guide* chapter. • Updated the steps in *Installing the Linux Kernel Driver*. • Updated the steps in *Running the PIO Design Example*. • Updated the steps in *Running the SR-IOV Design Example*.
2025.04.07	25.1	12.3.0	• Added a Note to *Hardware and Software Requirements*. • Updated the steps in *Installing the Linux Kernel Driver*. • Added information about FASTSIM mode and PIPE mode to *Simulating the Design Example*. • Updated the simulation commands for all simulators under *Steps to Run Simulation*. • Updated the steps in *Running the PIO Design Example*. • Updated the steps in *Running the SR-IOV Design Example* • Updated the steps in *Running the Performance Design Example*
2024.10.07	24.3	12.2.0	• Added information about the initialization clock frequency for the **Example Designs** tab to the *Generating the Design Example* section. • Removed the support for CentOS Linux from the *Hardware and Software Requirements* section. • Updated the simulation commands for all simulators in the *Steps to Run Simulation* section.

continued...

Document Version	Quartus Prime Version	IP Version	Changes
2024.05.23	24.1	12.0.0	• Added *Gen3* and *Gen4* terms in the *Terms and Definitions* table in the *Acronyms* section. • Updated supported simulators and development kit in the *F-Tile Avalon-ST IP Design Examples* table in the *Design Example Description* section. • Updated the *PIO Application* naming to *PIO Design Example for PCI Express G4* in all figures and sections using the name. • Updated the *SR-IOV Applications* naming to *F-Tile Avalon-ST SR-IOV Example Design* in all figures and sections using the name. • Updated *Performance Design Example* section with the supported frequency of only 500 MHz. • Updated *Generating the Design Example* section with PIPE mode simulation settings. • Removed note for the multiple driver issue for VCS and Xcelium simulators and added note for PIPE mode simulation in the *Steps to Run Simulation* section. • Added information about how to run FASTSIM + PIPE mode simulation in *Steps to Run Simulation : VCS* section. • Added information about how to run FASTSIM + PIPE mode simulation in *Steps to Run Simulation : VCS MX* section. • Added information about how to run FASTSIM + PIPE mode simulation in *Steps to Run Simulation : QuestaSIM* / ModelSim* - Altera® FPGA Starter Edition / Questa* – Altera FPGA Starter Edition* section. • Added information about how to run FASTSIM + PIPE mode simulation in *Steps to Run Simulation : Xcelium* section. • Added information about how to run FASTSIM + PIPE mode simulation in *Steps to Run Simulation : Riviera-PRO* section. • Updated *Compiling the Design Example* section with a note about PIPE mode simulation setting. • Updated *Hardware and Software Requirements* section with Linux software support and design example testing.
2024.01.29	23.4	11.0.0	• *PIO Design Example Testbench*: Simulation waveform updated • *Compiling the Design Example*: Information updated
2023.11.03	23.3	10.0.0	• *Steps to Run Simulation*: Note added about fix to address simulation errors for VCS and Xcelium • *Programmed Input/Output Design Example Limitation*: New section added • *SR-IOV Design Example Limitation*: New section added • *Steps to Run Simulation*: New section added
2023.04.27	23.1	8.1.0	• Updated product family name to "Intel Agilex 7." • *Design Example Description*: Hardware column updated in *F-Tile Avalon-ST IP Design Examples* table • *Design Example Description*: Note added about 10-bit tag completer feature & simulator support in FASTSIM. • *Generating the Design Example*: Step 11(c) development kit information updated

continued...

Document Version	Quartus Prime Version	IP Version	Changes
			• *Generating the Design Example*: *Example Designs Tab* screenshot updated • *Simulating the Design Example*: *Steps to run simulation* table removed • *Steps to Run Simulation*: New section added with sub-sections for individual simulators
2023.02.03	22.4	8.0.0	Added requirements for the reference clock for the System PLL to the *F-Tile Reference and System PLL Clocks IP* section.
2022.10.04	22.3	7.0.0	Sections Updated: • *Design Example Description*: New column *Hardware* added in table *F-Tile Avalon-ST IP Design Examples* • *Generating the Design Example*: Step 11(c) updated • *Simulating the Design Example*: Command updated for FASTSIM mode for VCS simulator in *Steps to Run Simulation* table • *Hardware and Software Requirements*: All contents updated • *Running the SR-IOV Design Example*: Steps (5) and (6) updated • *Running the Performance Design Example*: Note added about 500 MHz hardware test support • *Running the Performance Design Example*: Step (7) added • *Running the Performance Design Example*: Figures illustrating Write, Read, Simultaneous Write and Read operations added • *Running the Performance Design Example*: Test Results updated for Write, Read, Simultaneous Write and Read operations • *Running the Performance Design Example*: *Performance Design Example Data Throughput* information added
2022.07.14	22.2	6.0.0	• *Design Example Description*: *F-Tile Avalon-ST IP Design Examples* table updated • *Performance Design Example*: New design example information added • *Generating Design Example*: Step 11c added & *Example Designs Tab* GUI screenshot updated • *Simulating the Design Example*: Commands updated for all simulators in *Steps to Run Simulation* table • *Performance Design Example Testbench*: New section added • *Hardware and Software Requirements*: New section added • *Installing the Linux Kernel Driver*: Introductory description modified • *Running the Design Example*: *Test Operations Supported by the F-Tile Avalon-ST IP for PCI Express Design Examples* table reorganized and updated • *Running the PIO Design Example*: GUI screenshots for sample transcripts for automatic and manual modes updated • *Running the SR-IOV Design Example*: Link Test Menu GUI screenshot updated • *Running the Performance Design Example*: New section added

continued...

Document Version	Quartus Prime Version	IP Version	Changes
2021.12.17	21.4	4.0.0	• SR-IOV Design Example related information added to *Design Example Description* chapter • SR-IOV Design Example related information added to *Quick Start Guide* chapter • Generating tile files information added to *Quick Start Guide* chapter • Xcelium simulator information added to *Steps to run simulation* table • *Running the SR-IOV Design Example* section added to *Quick Start Guide* chapter
2021.10.04	21.3	3.0.0	• Updated *Configurations Supported by the F-Tile Avalon-ST Design Examples* Table • Added *PCIe Gen3/Gen4 x8 Design Example Variant* block diagram • Added PIO Design Example for PCI Express G4 (APPS) component information in Functional Description • Added *Block Diagram for the PCIe x8 PIO Design Example Simulation Testbench* block diagram • Added *Platform Designer System Contents for F-Tile Avalon®-ST IP for PCI Express PIO Design Example Gen4 x8 variant* screenshot • Update figure *Directory Structure for the Generated Design Example* • Procedure updated in *Simulating the Design Example* • Added VCSMX simulator information to *Steps to Run Simulation* table.
2021.07.31	21.2	2.0.0	Initial Release