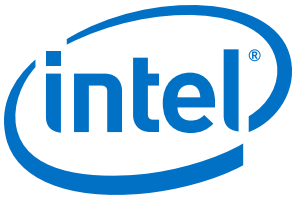


Intel® Arria® 10 and Intel® Cyclone® 10 GX Avalon streaming Hard IP for PCIe* Design Example User Guide

Updated for Intel® Quartus® Prime Design Suite: **17.1**

Subscribe
Send Feedback

UG-20039 | 2020.05.13
Latest document on the web: **PDF** | **HTML**

Contents

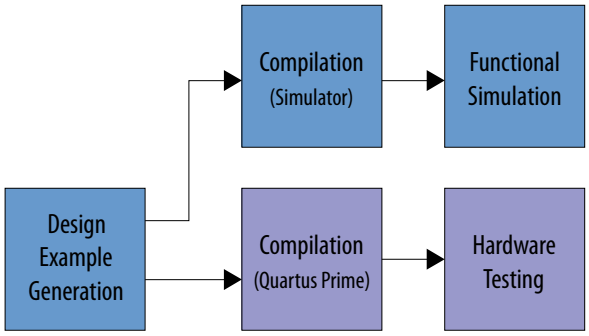
1. Quick Start Guide

The Intel® Arria® 10 or Intel Cyclone® 10 GX Hard IP for PCI Express* IP core includes a programmed I/O (PIO) design example to help you understand usage. The PIO example transfers data from a host processor to a target device. It is appropriate for low-bandwidth applications. The design example includes an Avalon-ST to Avalon-MM Bridge. This component translates the TLPs received on the PCIe* link to Avalon-MM memory reads and writes to the on-chip memory.

This design example automatically creates the files necessary to simulate and compile in the Quartus® Prime software. You can download the compiled design to the Intel Arria 10 GX FPGA Development Kit. The design examples cover a wide range of parameters. However, the automatically generated design examples do not cover all possible parameterizations of the PCIe IP Core. If you select an unsupported parameter set, generations fails and provides an error message.

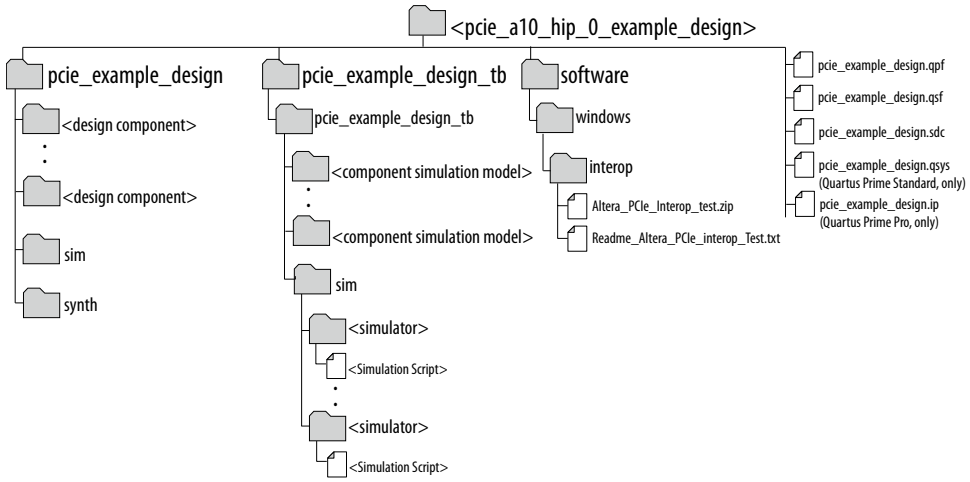
In addition, many static design examples for simulation are only available in the `<install_dir>/ip/altera/altera_pcie/altera_pcie_a10_ed/example_design/a10` and `<install_dir>/ip/altera/altera_pcie/altera_pcie_a10_ed/example_design/c10` directories.

Figure 1. Development Steps for the Design Example



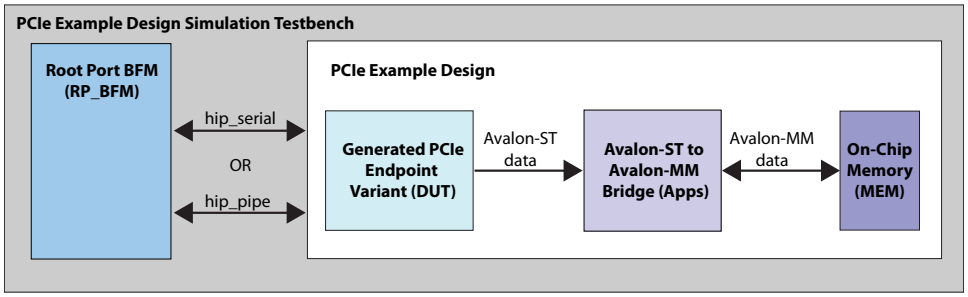
1.1. Directory Structure

Figure 2. Directory Structure for the Generated Design Example



1.2. Design Components

Figure 3. Block Diagram for the Platform Designer PIO Design Example Simulation Testbench

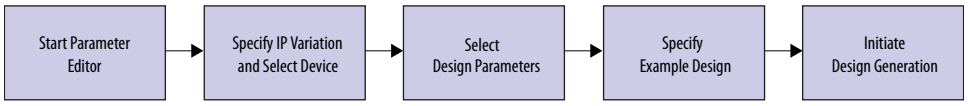


Related Information

Functional Description on page 11

1.3. Generating the Design

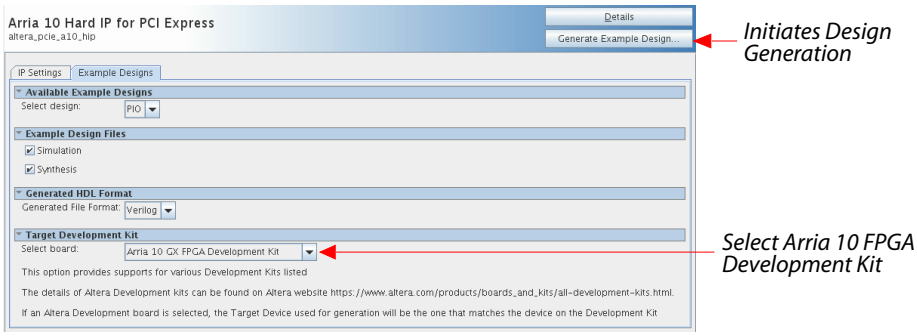
Figure 4. Procedure



Follow these steps to generate the design from the IP Parameter Editor:

1. In the IP Catalog (**Tools ➤ IP Catalog**) locate and select the **Intel Arria 10/ Cyclone 10 Hard IP for PCI Express**.
2. Starting with the Quartus Prime Pro 16.1 software, the **New IP Variation** dialog box appears.
3. Specify a top-level name and the folder for your custom IP variation, and the target device. Click **OK**
4. On the **IP Settings** tabs, specify the parameters for your IP variation.
5. On the **Example Designs** tab, the **PIO** design is available for your IP variation.

Figure 5. Example Design Tab

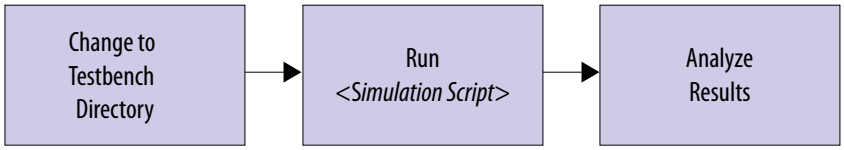


6. For **Example Design Files**, select the **Simulation** and **Synthesis** options.
7. For **Generated HDL Format**, only Verilog is available.
8. For **Target Development Kit** select the **Intel Arria 10 FPGA Development Kit** option.

   Note: Currently, you cannot select an **Intel Cyclone 10 GX Development Kit** when generating an example design.
9. Click the **Generate Example Design** button. The software generates all files necessary to run simulations and hardware tests on the **Intel Arria 10 FPGA Development Kit**. Click **Close** when generation completes.
10. Click **Finish**.
11. The prompt, **Recent changes have not been generated. Generate now?**, allows you to create files for simulation and synthesis. Click **No** to continue to simulate the design example you just generated.

1.4. Simulating the Design

Figure 6. Procedure



1. Change to the testbench simulation directory.
2. Run the simulation script for the simulator of your choice. Refer to the table below.
3. Analyze the results.

Table 1. Steps to Run Simulation

Simulator	Working Directory	Instructions
ModelSim*	*<example_design>*/pcie_example_design_tb/pcie_example_design_tb/sim/mentor/	1. Invoke vsim 2. 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_example_design_tb/pcie_example_design_tb/sim/synopsys/vcs	1. sh vcs_setup.sh USER_DEFINED_SIM_OPTIONS="" 2. A successful simulation ends with the following message, "Simulation stopped due to successful completion!"
NCSim*	*<example_design>*/pcie_example_design_tb/pcie_example_design_tb/sim/cadence	1. sh ncsim_setup.sh USER_DEFINED_SIM_OPTIONS="" 2. A successful simulation ends with the following message, "Simulation stopped due to successful completion!"
Xcelium* Parallel Simulator	*<example_design>*/pcie_example_design_tb/pcie_example_design_tb/sim/xcelium	1. sh xcelium_setup.sh USER_DEFINED_SIM_OPTIONS="" USER_DEFINED_ELAB_OPTIONS="-NOWARN\ CSINFI" 2. A successful simulation ends with the following message, "Simulation stopped due to successful completion!"

Figure 7. Partial Transcript from Successful Endpoint Avalon-ST PIO Simulation Testbench

```
# INFO:           60504 ns            New Link Speed: 8.0GT/s
# INFO:           60576 ns   RP PCI Express Link Control Register (0040):
# INFO:           60576 ns       Common Clock Config: System Reference Clock Used
# INFO:           61640 ns   RP PCI Express Link Capabilities Register (01606483):
# INFO:           61640 ns         Maximum Link Width: x8
# INFO:           61640 ns      Supported Link Speed: 8.0GT/s or 5.0GT/s or 2.5GT/s
# INFO:           61640 ns                 L0s Entry: Supported
# INFO:           61640 ns                 L1  Entry: Not Supported
# INFO:           61640 ns          L0s Exit Latency: 2 us to 4 us
# INFO:           61640 ns          L1  Exit Latency: Less Than 1 us
# INFO:           61640 ns               Port Number: 01
# INFO:           61768 ns   RP PCI Express Device Control Register (5010):
# INFO:           61768 ns   Error Reporting Enables: 0
# INFO:           61768 ns          Relaxed Ordering: Enabled
# INFO:           61768 ns               Max Payload: 128 Bytes
# INFO:           61768 ns              Extended Tag: Disabled
# INFO:           61768 ns          Max Read Request: 4KBytes
# INFO:           61768 ns   RP PCI Express Device Status Register (0000):
# INFO:           62096 ns Configuring Bus 000, Device 000, Function 00
# INFO:           62096 ns   RP Read Only Configuration Registers:
# INFO:           62096 ns                 Vendor ID: 1172
# INFO:           62096 ns                 Device ID: E001
# INFO:           62096 ns               Revision ID: 01
# INFO:           62096 ns                Class Code: FF0000
# INFO:           62096 ns             Interrupt Pin: INTA# used
# INFO:           62784 ns BAR Address Assignments:
# INFO:           62784 ns BAR    Size       Assigned Address  Type
# INFO:           62784 ns BAR0   Disabled
# INFO:           62784 ns BAR1   Disabled
# INFO:           62784 ns ExpROM Disabled
# INFO:           66680 ns Completed configuration of Endpoint BARs.
# INFO:           67728 ns TASK:downstream_loop
# INFO:           68584 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           69448 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           70296 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           71160 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           72008 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           72864 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           73720 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           74568 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           75432 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# INFO:           76280 ns Passed: 0004 same bytes in BFM mem addr 0x00000040 and 0x00000840
# SUCCESS: Simulation stopped due to successful completion!
```

1.5. Compiling and Testing the Design in Hardware

Figure 8. Procedure

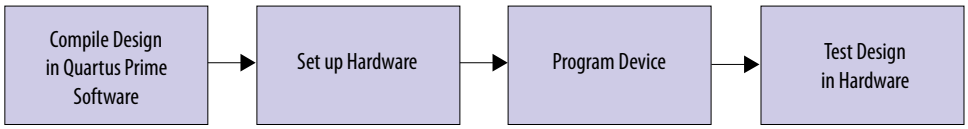
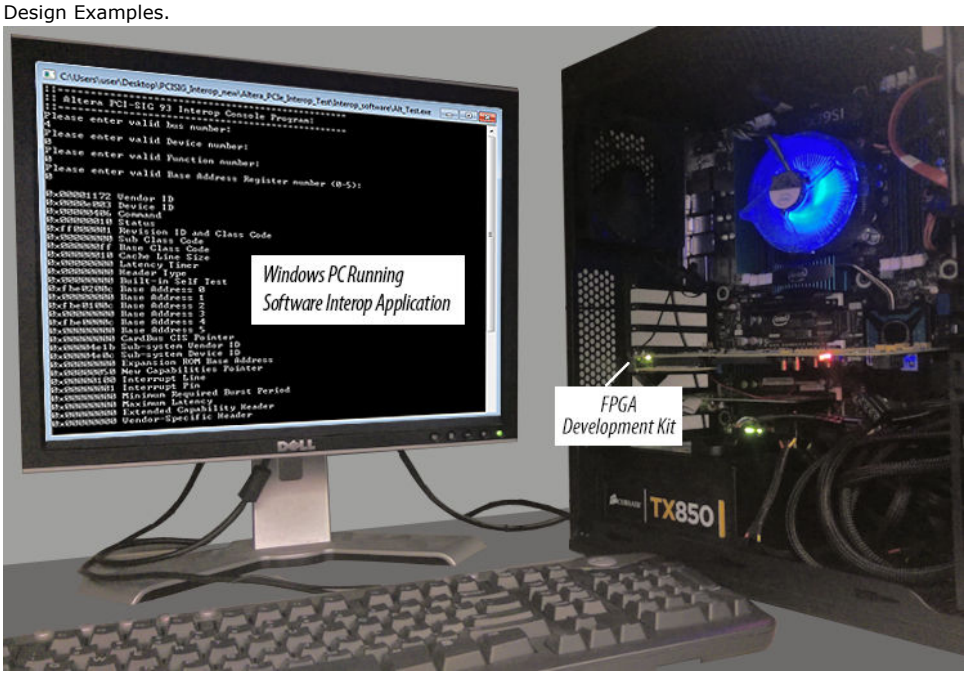


Figure 9. Software Application to Test the PCI Express Design Example on the Intel Arria 10 GX FPGA Development Kit

A software application running on a Windows PC performs the same hardware test for all of the PCI Express Design Examples.



The software application to test the PCI Express Design Example on the Intel Arria 10 GX FPGA Development Kit is available on both 32- and 64-bit Windows platforms. This program performs the following tasks:

1. Prints the Configuration Space, lane rate, and lane width.
2. Writes 0x00000000 to the specified BAR at offset 0x00000000 to initialize the memory and read it back.
3. Writes 0xABCD1234 at offset 0x00000000 of the specified BAR. Reads it back and compares.

If successful, the test program displays the message 'PASSED'

Follow these steps to compile the design example in the Quartus Prime software:

1. Launch the Quartus Prime software and open the `pcie_example_design.qpf` file for the example design created above.
2. On the **Processing** > menu, select **Start Compilation**.

   The timing constraints for the design example and the design components are automatically loaded during compilation.

Follow these steps to test the design example in hardware:

1. In the *<example_design>*`/software/windows/interop` directory, unzip `Altera_PCIe_Interop_Test.zip`.

Note: You can also refer to `readme_Altera_PCIe_interop_Test.txt` file in this same directory for instructions on running the hardware test.

2. Install the Intel FPGA Windows Demo Driver for PCIe on the Windows host machine, using **altera_pcie_win_driver.inf**.

   Note: If you modified the default Vendor ID (0x1172) or Device ID (0x0000) specified in the component parameter editor GUI, you must also modify them in **altera_pcie_win_driver.inf**.

   a. In the <*example_design*> directory, launch the Quartus Prime software and compile the design (**Processing** > **Start Compilation**).

   b. Connect the development board to the host computer.

   c. Configure the FPGA on the development board using the generated `.sof` file (**Tools** > **Programmer**).

   d. Open the Windows Device Manager and scan for hardware changes.

   e. Select the Intel FPGA listed as an unknown PCI device and point to the appropriate 32- or 64-bit driver (**altera_pice_win_driver.inf**) in the **Windows_driver** directory.

   f. After the driver loads successfully, a new device named **Altera PCI API Device** appears in the Windows Device Manager.

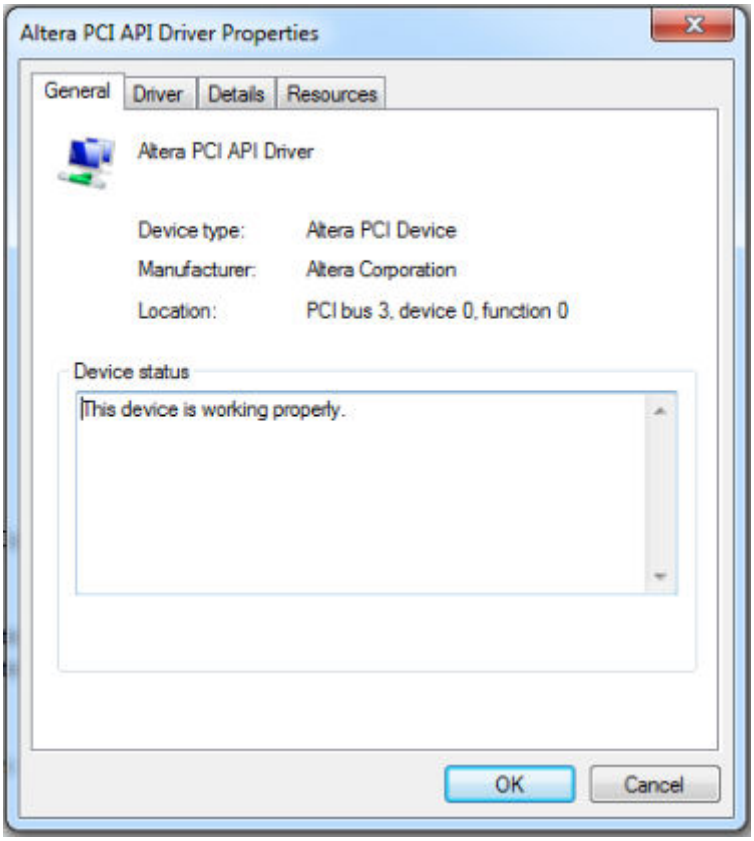
   g. Determine the bus, device, and function number for the **Altera PCI API Device** listed in the Windows Device Manager.

      i. Expand the tab, **Altera PCI API Driver** under the devices.

      ii. Right click on **Altera PCI API Device** and select **Properties**.

      iii. Note the bus, device, and function number for the device. The following figure shows one example.

Figure 10. Determining the Bus, Device, and Function Number for New PCIe Device



3. In the <*example_design*>/software/windows/interop/Altera_PCIe_Interop_Test/Interop_software directory, click Alt_Test.exe.
4. When prompted, type the bus, device, and function numbers and select the BAR number (0-5) you specified when parameterizing the IP core.

   Note: The bus, device, and function numbers for your hardware setup may be different.
5. The test displays the message, PASSED, if the test is successful.

Note: For more details on additional design implementation steps such as making pin assignments and adding timing constraints, refer to the *Design Implementation* chapter.

Related Information

Intel Arria 10 GX FPGA Development Kit

2. Design Example Description

2.1. Functional Description

The testbench illustrates PIO traffic between the host and Endpoint. The PIO design example consists of memory transfers from a host processor to a target device. In this example, the host processor issues single-dword MemRd and MemWr TLPs.

The Endpoint (DUT) and PIO application (APPS) perform the necessary translation between the PCI Express TLPs and simple Avalon-MM reads and writes to memory.

The PIO testbench includes the following components:

- The Root Port BFM that drives downstream TLPs to the Endpoint.

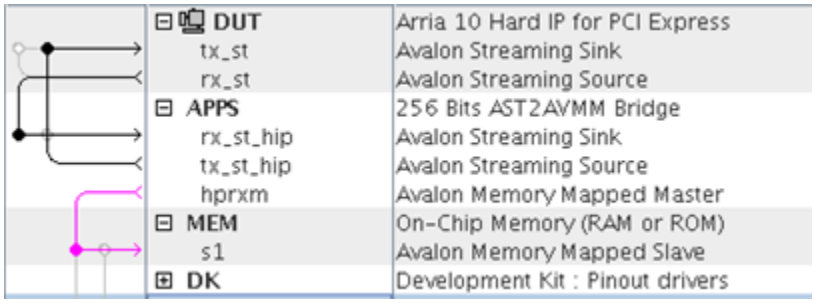
  Note: This Intel Root Port BFM provides a simple method to do basic testing of the Application Layer logic that interfaces to the DUT. However, the testbench and Root Port BFM are not intended to be a substitute for a full verification environment. To thoroughly test your application, obtain commercially available PCI Express verification IP and tools, or do your own extensive hardware testing or both.
- The Generated PCIe Endpoint Variant (DUT) with the parameters you specified. This component drives TLP data received to the PIO application.
- The PIO Application (APPS) component. Along with some additional logic, it translates Avalon-ST data to Avalon-MM data for writes and reads to the on-chip memory.
- For variants up to Gen3 x8, an on-chip memory (MEM) stores data. Gen3 x16 variants include the memory in the APPs component.
- A Development Kit (DK) conduit interface provides access to PCIe control and status signals for debugging using the Arria 10 FPGA GX Development Kit for ES Devices.

The test program writes and reads back data to 0x00000040 in the on-chip memory. It compares data read to the expected result. The test reports, "Simulation stopped due to successful completion" if no errors occur.

The log file, `altpcie_monitor_s10_dlhip_tlp_file_log.log` records each TLP for both the initial configuration of the PCIe Endpoint and the test program.

Figure 11. Platform Designer System Contents for Arria 10 PCI Express PIO Design Example

This view of the Arria 10 PCI Express PIO Design Example shows only the Avalon-ST and Avalon-MM interfaces.



2.2. Creating a Signal Tap Debug File to Match Your Design Hierarchy

For Intel Arria 10 and Intel Cyclone 10 GX devices, the Intel Quartus Prime software generates two files, `build_stp.tcl` and `<ip_core_name>.xml`. You can use these files to generate a Signal Tap file with probe points matching your design hierarchy.

The Intel Quartus Prime software stores these files in the `<IP core directory>/synth/debug/stp/` directory.

Synthesize your design using the Intel Quartus Prime software.

1. To open the Tcl console, click **View ➤ Utility Windows ➤ Tcl Console**.
2. Type the following command in the Tcl console:
   `source <IP core directory>/synth/debug/stp/build_stp.tcl`
3. To generate the STP file, type the following command:
   `main -stp_file <output stp file name>.stp -xml_file <input xml_file name>.xml -mode build`
4. To add this Signal Tap file (**.stp**) to your project, select **Project ➤ Add/Remove Files in Project**. Then, compile your design.
5. To program the FPGA, click **Tools ➤ Programmer**.
6. To start the Signal Tap Logic Analyzer, click **Quartus Prime ➤ Tools ➤ Signal Tap Logic Analyzer**.

   The software generation script may not assign the Signal Tap acquisition clock in `<output stp file name>.stp`. Consequently, the Intel Quartus Prime software automatically creates a clock pin called `auto_stp_external_clock`. You may need to manually substitute the appropriate clock signal as the Signal Tap sampling clock for each STP instance.
7. Recompile your design.
8. To observe the state of your IP core, click **Run Analysis**.

   You may see signals or Signal Tap instances that are red, indicating they are not available in your design. In most cases, you can safely ignore these signals and instances. They are present because software generates wider buses and some instances that your design does not include.

2.3. Intel Arria 10 Development Kit Conduit Interface

The Intel Arria 10 Development Kit conduit interface signals are optional signals that allow you to connect your design to the Intel Arria 10 FPGA Development Kit. Enable this interface by selecting **Enable Intel Arria 10 FPGA Development Kit connection** on the **Configuration, Debug, and Extension Options** tab of the component GUI. The `devkit_status` output port includes signals useful for debugging.

Table 2. The Intel Arria 10 Development Kit Conduit Interface

Signal Name	Direction	Description
`devkit_status[255:0]`	Output	The `devkit_status[255:0]` bus comprises the following status signals : • `devkit_status[1:0]: current_speed` • `devkit_status[2]: derr_cor_ext_rcv` • `devkit_status[3]: derr_cor_ext_rpl` • `devkit_status[4]: derr_err` • `devkit_status[5]: rx_par_err` • `devkit_status[7:6]: tx_par_err` • `devkit_status[8]: cfg_par_err` • `devkit_status[9]: dlup` • `devkit_status[10]: dlup_exit` • `devkit_status[11]: ev128ns` • `devkit_status[12]: ev1us` • `devkit_status[13]: hotrst_exit` • `devkit_status[17:14]: int_status[3:0]` • `devkit_status[18]: l2_exit` • `devkit_status[22:19]: lane_act[3:0]` • `devkit_status[27:23]: ltssmstate[4:0]` • `devkit_status[35:28]: ko_cpl_spc_header[7:0]` • `devkit_status[47:36]: ko_cpl_spc_data[11:0]` • `devkit_status[48]: rxfc_cplbuf_ovf` • `devkit_status[49]: reset_status` • `devkit_status[255:50]: Reserved`
`devkit_ctrl[255:0]`	Input	The `devkit_ctrl[255:0]` bus comprises the following status signals. You can optionally connect these pins to an on-board switch for PCI-SIG compliance testing, such as bypass compliance testing. • `devkit_ctrl[0]:test_in[0] is typically set to 1'b0` • `devkit_ctrl[4:1]:test_in[4:1] is typically set to 4'b0100` • `devkit_ctrl[6:5]:test_in[6:5] is typically set to 2'b01` • `devkit_ctrl[31:7]:test_in[31:7] is typically set to 25'h3` • `devkit_ctrl[63:32]:is typically set to 32'b0` • `devkit_ctrl[255:64]:is typically set to 192'b0`

2.4. Serial Data Signals

This differential, serial interface is the physical link between a Root Port and an Endpoint.

The Intel Cyclone 10 GX PCIe IP Core supports 1, 2, or 4 lanes. Each lane includes a TX and RX differential pair. Data is striped across all available lanes.

The Intel Arria 10 PCIe IP Core supports 1, 2, 4 or 8 lanes. Each lane includes a TX and RX differential pair. Data is striped across all available lanes.

Table 3. 1-Bit Interface Signals

In the following table <*n*> is the number of lanes.

Signal	Direction	Description
`tx_out[<n>-1:0]`	Output	Transmit output. These signals are the serial outputs of lanes <*n*>-1–0.
`rx_in[<n>-1:0]`	Input	Receive input. These signals are the serial inputs of lanes <*n*>-1–0.

Refer to *Pin-out Files for Intel Devices* for pin-out tables for all Intel devices in **.pdf**, **.txt**, and **.xls** formats.

Transceiver channels are arranged in groups of six. For GX devices, the lowest six channels on the left side of the device are labeled GXB_L0, the next group is GXB_L1, and so on. Channels on the right side of the device are labeled GXB_R0, GXB_R1, and so on. Be sure to connect the Hard IP for PCI Express on the left side of the device to appropriate channels on the left side of the device, as specified in the *Pin-out Files for Intel Devices*.

Related Information

- Creating a Signal Tap Debug File to Match Your Design Hierarchy
  Outlines the steps necessary to observe the state of the PCIe IP Core using a Signal Tap debug file.
- Arria 10 or Cyclone 10 PCIe Avalon-ST Interfaces and Signal Descriptions
  Provides descriptions for all top-level signals in the PCIe IP Core.
- Pin-out Files for Intel Devices

2.5. Registers

There are no control registers for the PIO design example. The *PCI Express Base Specification 3.0* defines a comprehensive set of configuration, control, and status registers to control and debug the design example.

Related Information

Registers in the Arria 10 or Cyclone 10 Hard IP for PCI Express

A. Document Revision History for Intel Arria 10 and Intel Cyclone 10 GX Avalon® streaming Hard IP for PCIe Design Example User Guide

Date	Version	Changes Made
2020.05.13	17.1	Added clarification that the Intel Cyclone 10 GX Development Kit is not currently supported.
2017.11.06	17.1	Made the following changes: • Added support for Intel Cyclone 10 GX devices. • Added explanation for the dummy connections made in the *Generating the Design* topic.
2017.03.15	16.1.1	Rebranded as Intel.
2016.10.31	16.1	Initial release.