

# **AN 813: Hierarchical Partial Reconfiguration over PCI Express\* Reference Design for Intel® Arria® 10 Devices**

***AN-813***  
***2017.05.22***



**Subscribe**



**Send Feedback**



## Contents

---

<b>Hierarchical Partial Reconfiguration over PCI Express* Reference Design for Arria®</b>	
<b>10 Devices.....</b>	<b>3</b>
Reference Design Overview.....	4
Clocking Scheme.....	5
Memory Address Mapping.....	5
Floorplanning.....	6
Getting Started.....	6
Hardware and Software Requirements.....	6
Installing the Arria 10 GX FPGA Development Kit.....	7
Installing the Linux Driver.....	7
Bringing Up the Reference Design.....	7
Reference Design Components.....	8
BSP Top.....	8
Compiling the Reference Design.....	11
Testing the Reference Design.....	14
program_fpga_jtag.....	14
program_fpga_pcie.....	15
example_host_uio.....	15
Extending the Reference Design with Custom Persona.....	17
Document Revision History.....	19
<b>Reference Design Files.....</b>	<b>20</b>



## Hierarchical Partial Reconfiguration over PCI Express\* Reference Design for Arria® 10 Devices

---

The Hierarchical Partial Reconfiguration (HPR) over PCI Express\* (PCIe\*) reference design demonstrates reconfiguring the FPGA fabric through the PCIe link in Arria® 10 devices. This reference design runs on a Linux system with the Arria 10 GX FPGA development board. Adapt this reference design to your requirements by implementing the PR region logic using the given template. Run your custom design on this fully functional system that enables communication over PCIe.

Arria 10 devices use the PR over PCIe solution to reconfigure the device, rather than Configuration via Protocol (CvP) update. Partial reconfiguration allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Create multiple personas for a particular region in your design, without impacting operation in areas outside this region. Partial reconfiguration enables the implementation of more complex FPGA systems.

You can also include multiple parent and child partitions, or create multiple levels of partitions in your design. This flow, referred to as the hierarchical partial reconfiguration (HPR), includes a static region that instantiates the parent PR region, and the parent PR region instantiating the corresponding child PR region. You can perform the same PR region reprogramming for either the child or the parent partition. Reprogramming a child PR region does not affect the parent or the static region. Reprogramming the parent region reprograms the associated child region with the default child persona, without affecting the static region.

**Note:** The HPR flow does not impose any restrictions on the number of sub-partitions you can create in your design.

Partial reconfiguration provides the following advancements to a flat design:

- Allows run-time design reconfiguration
- Increases scalability of the design through time-multiplexing
- Lowers cost and power consumption through efficient use of board space
- Supports dynamic time-multiplexing functions in the design
- Improves initial programming time through smaller bitstreams
- Reduces system down-time through line upgrades
- Enables easy system update by allowing remote hardware change

The Quartus® Prime Pro Edition software supports the partial reconfiguration feature for the Arria 10 device family.

### Related Links

- [Creating a Partial Reconfiguration Design](#)  
For complete information on the partial reconfiguration design flow.



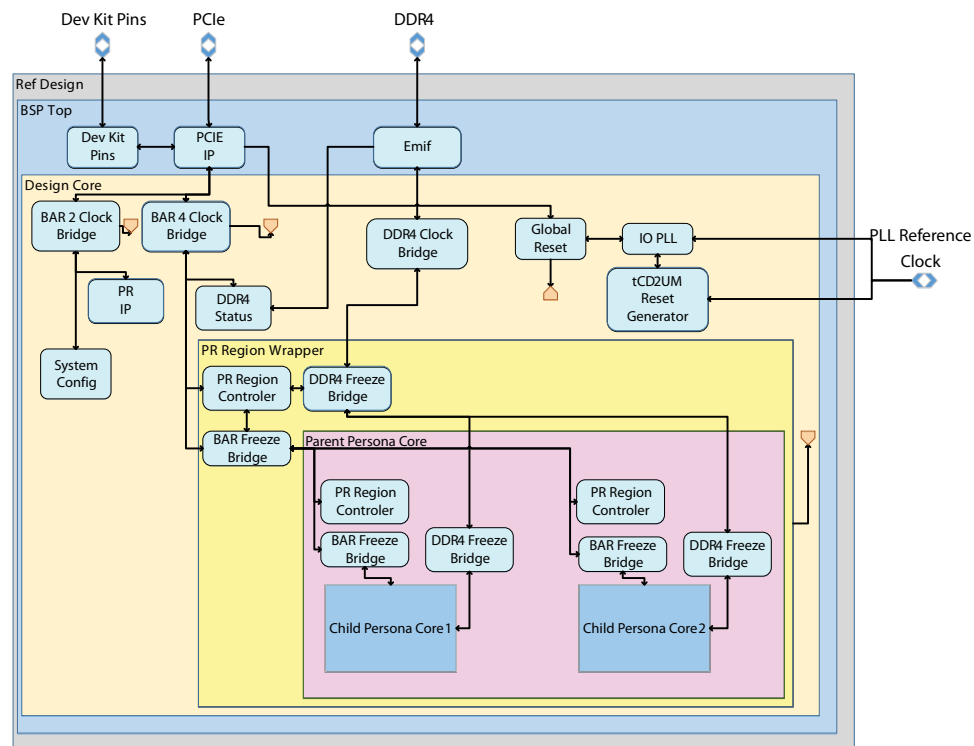
- [Hierarchical Partial Reconfiguration of a Design on Arria 10 GX FPGA Development Board](#)  
For a step-by-step tutorial on creating a hierarchical partial reconfiguration design.
- [Arria 10 CvP Initialization and Partial Reconfiguration via Protocol User Guide](#)  
For complete information on the Configuration via Protocol (CvP) configuration scheme.
- [Arria 10 FPGA Development Kit User Guide](#)  
For information on Arria 10 GX FPGA development board overview and setup procedure.

## Reference Design Overview

The reference design consists of the following components:

- `a10_pcie_reference_design.sv`—top-level wrapper for the reference design, connecting the board support package (BSP) subsystem to the device pins.
- `bsp_top`—top-level of the design that contains all subsystems of the design. This module consists of three main sub-components - the PCIe IP core, the DDR4 External Memory Interfaces IP core, and the design top module. This layer of abstraction allows simulation of the design top module through simulated Avalon-MM transactions.
- `design_core`—core of the design that handles generation of the PR region, the interface components such as clock crossing Avalon-MM logic and pipeline logic, clocks, and the global reset.

**Figure 1. Arria 10 PCIe Reference Design Block Diagram**

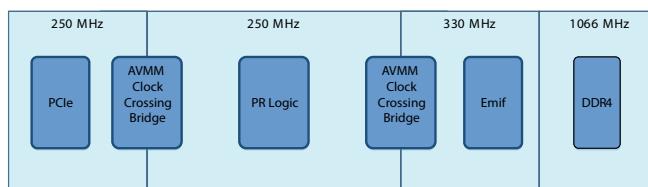




## Clocking Scheme

The reference design creates a separate Altera IOPLL IP core-generated clock. This clock creation decouples the PR logic clocking from both the PCIe clocking domain that runs at 250 MHz, and the EMIF clocking domain that runs at 330 MHz. The clock for PR Logic is set at 250 MHz. To ensure timing closure, modify the parameterization of the IOPLL IP core to a lower clock frequency.

**Figure 2. Timing Closure**



## Memory Address Mapping

The PCIe IP core connects to the design core through two BARs (base address registers) - BAR 2 and BAR 4, which in turn connect to their exclusive Avalon-MM interface.

In addition to connecting to the interface controls such as the Avalon-MM freeze bridges and the PR region controller for the PR region, BAR 4 also connects directly to up to 8 kB of memory in the PR region. Driver accessed components, such as the PR IP core and the system description ROM use BAR 2. The following table lists the memory address mapping for the PCIe IP core:

**Table 1. PCIe Memory Address Map**

Domain	Address Map	Base	End
BAR 2	System Description ROM	0x0000_0000	0x0000_0FFF
BAR 2	PR IP	0x0000_1000	0x0000_103F
BAR 4	PR Region	0x0000_0000	0x0000_FFFF
BAR 4	PR Region Controller	0x0001_0000	0x0001_000F
BAR 4	DDR4 Calibration Export	0x0001_0010	0x0001_001F

The reference design exports the status of the DDR4 calibration, provided by the External Memory Interfaces IP core. Upon initialization, the EMIF IP core performs training to reset the DDR4 interface. The EMIF reports the success or the failure of the reset in its calibration flag. The host decides the necessary action in the event of DDR4 failing the reset training, so this interface is exported to the host. The following table lists the memory address mapping from the External Memory Interfaces IP core, to the PR logic:

**Table 2. DDR4 External Memory Interfaces (EMIF) Memory Address Map**

Address Map	Base	End
DDR	0x0000_0000	0x7fff_ffff



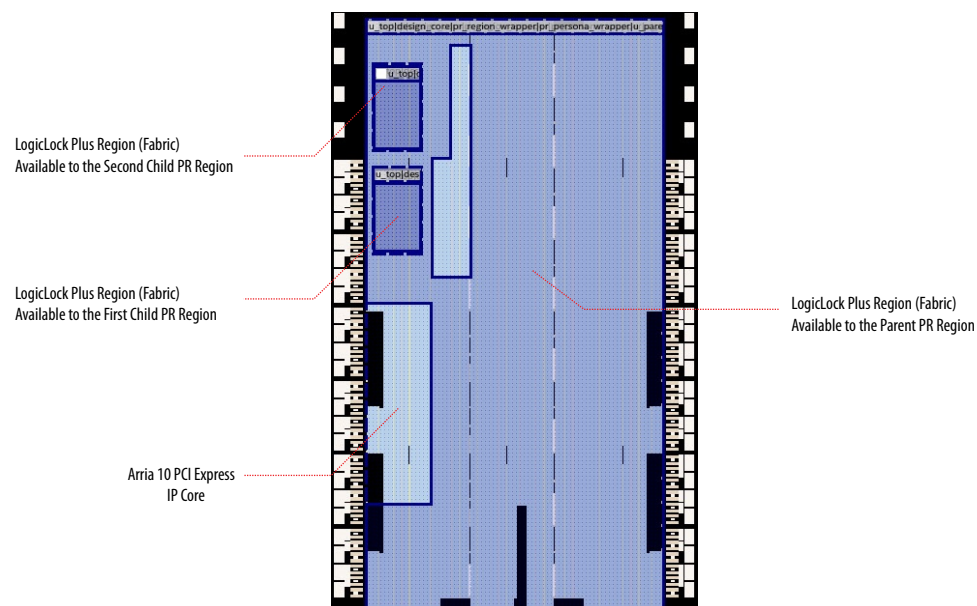
The 2 GB DDR4 memory space is available to PR logic through an Avalon-MM interface.

## Floorplanning

The floorplan constraints in your partial reconfiguration design physically partitions the device. This partitioning ensures that the resources available to the PR region are the same for any persona that you implement.

To maximize the fabric amount available for the PR region, the reference design constrains the static region to the smallest possible area. This reference design contains two child PR regions of the parent PR region.

**Figure 3. Reference Design Floorplan**



**Note:** The child regions contained within the parent PR region can be of any size. The small sizing of the child PR regions in the above figure is for demonstration purposes only.

For more information on floorplanning, refer to *Floorplan the Partial Reconfiguration Design* section in Volume 1 of the *Quartus Prime Pro Edition Handbook*.

### Related Links

[Floorplan the Partial Reconfiguration Design](#)

## Getting Started

This section describes the requirements and the procedure to run the reference design.

## Hardware and Software Requirements

The reference design requires and uses the following hardware and software tools:



- Arria 10 GX FPGA development board
- Linux Operating System - kernel version 3.10 or above
- PCIe slot to plug-in the Arria 10 GX FPGA development board
- Open source driver for this PR over PCIe reference design
- Quartus Prime Pro Edition software v.17.0

**Note:** The open source driver developed for this PR over PCIe reference design is tested using CentOS 7.

## Installing the Arria 10 GX FPGA Development Kit

For complete instructions on installing and powering the Arria 10 GX FPGA development board in your Linux system, refer to *Arria 10 FPGA Development Kit User Guide*.

**Note:** Before powering the board, set the switch 4 (FACTORY) of the DIP switch bank (SW6) to **ON**. Setting this switch to **ON** loads the factory image area of the flash memory at boot time. Program the reference design into this factory image area. For complete instructions on flashing the reference design onto the board, refer to *Bringing Up the Reference Design*.

### Related Links

- [Arria 10 FPGA Development Kit User Guide](#)
- [Bringing Up the Reference Design](#) on page 7

## Installing the Linux Driver

The reference design includes the complete source code for the open source Linux driver, developed and tested for this reference design.

To compile and load the Linux driver, follow the instructions in the README file in the following location:

<https://github.com/01org/fpga-partial-reconfig/tree/master/software/drivers>

## Bringing Up the Reference Design

The reference design is available in the following location:

<https://github.com/01org/fpga-partial-reconfig>

To access the reference design, navigate to the `ref_designs` sub-folder. Copy the `a10_pcie_devkit_cvp_hpr` folder to the home directory in your Linux system.

To bring up the reference design on the board:



1. Plug-in the Arria 10 GX FPGA development board to an available PCIe slot in your host machine.
2. Connect the host machine's ATX auxiliary power connector to the 12 V ATX input J4 of the development board.
3. Power-up the host machine.
4. Verify the micro-USB cable connection to the FPGA development board. Ensure that no other applications that use the JTAG chain are running.
5. Navigate to the `a10_pcie_devkit_cvp_hpr/software/installation` folder in your system.
6. To overwrite the existing factory image on the board with the reference design, execute the `run_program.sh` script.

Running this script configures the device with the contents of the `flash.pof` file. This parallel object file acts as the base image for the reference design.

7. Power-cycle the host machine.

## Reference Design Components

The reference design contains the following design components.

### BSP Top

This Qsys Pro system contains all the subsystems of this reference design. The system comprises of three main components - the top-level design, the PCIe IP core, and the DDR4 External Memory Interfaces IP core. The system connects to external pins through the `a10_pcie_ref_design.sv` wrapper.

### PCI Express IP core

The Arria 10 Hard IP for PCI Express IP core is a Gen3x8, with a 256-bit interface and running at 250 MHz.

The following table provides information on the configuration fields of the PCI Express IP core that the reference design uses that are different from the default settings:

**Table 3. PCI Express IP Core Configuration**

Setting	Parameter	Value
<b>System Settings</b>	<b>Application interface type</b>	Avalon-MM with DMA
	<b>Hard IP mode</b>	Gen3:x8, Interface: 256-bit, 250 MHz
	<b>Port type</b>	Native endpoint
	<b>RX buffer credit allocation for received requests vs completions</b>	Low
<b>Avalon-MM Settings</b>	<b>Enable control register access (CRA) Avalon-MM slave port</b>	Disable
<b>Base Address Registers - BAR2</b>	<b>Type</b>	32-bit non-prefetchable memory
<b>Base Address Registers - BAR4</b>	<b>Type</b>	32-bit non-prefetchable memory
<b>Device Identification Registers</b>	<b>Vendor ID</b>	0x00001172
continued...		





Setting	Parameter	Value
	Device ID	0x00005052
	Revision ID	0x00000001
	Class code	0x00ea0001
	Subsystem Vendor ID	0x00001172
	Subsystem Device ID	0x00000001
PCI Express/PCI Capabilities - Device	Maximum payload size	256 Bytes
Configuration, Debug, and Extension Options	Enable Arria 10 GX FPGA Development Kit Connection	Enable
PHY Characteristics	Requested equalization far-end TX preset	Preset 9

**Note:** Instantiate the PCI Express IP core as part of a Qsys Pro system.

## Arria 10 DDR4 External Memory Interfaces IP Core

The `ddr4_emif` logic includes the Arria 10 External Memory Interfaces IP core. This IP core interfaces to the DDR4 external memory, with a 64-bit interface that runs at 1066.0 MHz. Also, the IP core provides 2 GB of DDR4 SDRAM memory space. The EMIF Avalon-MM slave runs at 300 MHz clock.

The following table lists the configuration fields of the Arria 10 External Memory Interfaces IP core that are different from the Arria 10 GX FPGA Development Kit with DDR4 HILO preset settings:

**Table 4. Arria 10 DDR4 External Memory Interfaces IP Configuration**

Setting	Parameter	Value
Memory - Topology	DQ width	64
	DQ pins per DQS group	8
	Number of DQS groups	8
	Alert# pin placement	I/O Lane with Address/Command Pins
	Address/Command I/O lane of ALERT#	3
	Pin index of ALERT#	0

## Design Top

This component forms the core of the design, and includes the following:

- Reset logic
- PR region
- Partial Reconfiguration IP core
- Clock crossing and pipe-lining for Avalon MM Transactions
- System description ROM
- PLL



## Global Reset Logic

The PLL generates the main clock for this design. All logic, excluding the `pcie_ip`, `pr_ip`, and `ddr4_emif` run using this 250 MHz clock. The PCIe generates the global reset, along with the PLL reset signal. On power up, a countdown timer, `tcd2um` counts down using the internal 50 MHz oscillator, to a 830  $\mu$ s delay. Until the timer reaches this delay, the PLL is held in reset, deasserting the locked signal. This action freezes the design, and because the PLL locked signal is ORed with the PCIe reset, the design also is held in reset. Once the timer reaches 830  $\mu$ s, the design functions normally, and enters a known state.

## PR Region Wrapper

The PR region wrapper contains the PR region controller, freeze bridges, and the personas. The region controller interacts with the driver over the Avalon-MM interface to initiate PR, and act as a bridge for communicating to the PR region for freeze and start requests initiation.

## Parent PR Region

The parent PR region contains two sets of PR region controllers and freeze bridges, because of the two child personas within the parent PR region. Each set of PR region controller and freeze bridge is dedicated to one child persona.

## Arria 10 Partial Reconfiguration Region Controller IP Core

Use the Arria 10 Partial Reconfiguration Region Controller IP core to initiate a freeze request to the PR region. The PR region finalizes any actions, on freeze request acknowledgment. The freeze bridges also intercept the Avalon-MM interfaces to the PR region, and correctly responds to any transactions made to the PR region during partial reconfiguration. Finally, on PR completion, the region controller issues a stop request, allowing the region to acknowledge, and act accordingly. The `fpga-region-controller` program provided with this reference design performs these functions.

The reference design configures the Partial Reconfiguration Region Controller IP core to operate as an internal host. The design connects this IP core to the PCI Express IP core, via an instance of the Avalon-MM interface. The PR IP core has a clock-to-data ratio of 1. Therefore, the PR IP core is not capable of handling encrypted or compressed PR data.

The following table lists the configuration fields of the Arria 10 Hard IP for PCI Express IP core that are different from the preset settings:

Parameters	Value
Enable JTAG debug mode	Disable
Enable Avalon-MM slave interface	Enable
Input data width	32

## Partial Reconfiguration Logic

The reference design provides the following personas:

**Table 5. Reference Design Personas**

Persona	Description
DDR4 access	Performs a sweep across a memory span, first writing, and then reading each address.
Basic DSP	Provides access to a 27x27 DSP multiplier, and demonstrates hardware acceleration.
Basic arithmetic	Includes a basic 32-bit unsigned adder, and demonstrates hardware acceleration.
Game of Life	Includes an 8x8 Conway's Game of Life, and demonstrates hardware acceleration.
Parent persona	A wrapper that instantiates two child partitions. The parent persona also connects the two child personas to the static region with their own PR region controller, BAR freeze bridge, and DDR4 freeze bridge.

Each persona has an 8-bit `persona_id` field in the `pr_data` register to indicate a unique identification number. A 32-bit control register and 16 I/O registers follow the 8-bit `persona_id`. The 16 I/O registers are 32-bit each, with 8 bits for device inputs, and 8 bits for device outputs. Each persona uses these registers in different ways. For more information, refer to the source code for each of the personas.

Additionally, the reference design provides a template to implement your custom persona. This template persona allows you modify the RTL, create a wrapper to interface with the register file, compile, and run your design.

## Compiling the Reference Design

Quartus Prime provides a flow script to compile and run your PR personas. To generate and run the hierarchical partial reconfiguration flow script:

1. From the Quartus Prime command shell, create a flow template by running the following command:

```
quartus_sh --write_flow_template -flow a10_hier_partial_reconfig
```

Quartus Prime generates the `a10_hier_partial_reconfig/flow.tcl` file.

2. Rename the generated `a10_hier_partial_reconfig/setup.tcl.example` to `a10_hier_partial_reconfig/setup.tcl`, and modify the script to specify your partial reconfiguration project details:



- a. To define the name of the project, update the following line:

```
define_project a10_pcie_devkit_cvp
```

- b. To define the base revision, update the following line:

```
define_base_revision a10_pcie_devkit_cvp
```

3. To define each of the partial reconfiguration implementation revisions, along with the parent and child PR partition names and the synthesis revision that implements the revisions, update the following lines:

```
define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_ddr4_access \  
-partition_name pr_parent_partition \  
-source_rev_name synth_parent_persona \  
-source_partition root_partition \  
-source_snapshot synthesized  
  
define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_ddr4_access \  
-partition_name pr_child_partition_0 \  
-source_rev_name synth_ddr4_access \  
-source_partition root_partition \  
-source_snapshot synthesized  
  
define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_ddr4_access \  
-partition_name pr_child_partition_1 \  
-source_rev_name synth_ddr4_access \  
-source_partition root_partition \  
-source_snapshot synthesized  
  
define_pr_impl_partition -impl_rev_name  
a10_pcie_devkit_cvp_normal_ddr4_access \  
-partition_name pr_parent_partition \  
-source_rev_name synth_ddr4_access \  
-source_partition root_partition \  
-source_snapshot synthesized  
  
define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_basic_arithmetic \  
-partition_name pr_parent_partition \  
-source_rev_name a10_pcie_devkit_cvp_ddr4_access \  
-source_partition pr_parent_partition \  
-source_snapshot final  
  
define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_basic_arithmetic \  
-partition_name pr_child_partition_0 \  
-source_rev_name synth_basic_arithmetic \  
-source_partition root_partition \  
-source_snapshot synthesized  
  
define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_basic_arithmetic \  
-partition_name pr_child_partition_1 \  
-source_rev_name synth_basic_arithmetic \  
-source_partition root_partition \  
-source_snapshot synthesized  
  
define_pr_impl_partition -impl_rev_name  
a10_pcie_devkit_cvp_normal_basic_arithmetic \  
-partition_name pr_parent_partition \  
-source_rev_name synth_basic_arithmetic \  
-source_partition root_partition \  
-source_snapshot synthesized  
  
define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_basic_dsp \  
-partition_name pr_parent_partition \  
-source_rev_name a10_pcie_devkit_cvp_ddr4_access \  
-source_partition pr_parent_partition \
```



```

-source_snapshot final

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_basic_dsp \
-partition_name pr_child_partition_0 \
-source_rev_name synth_basic_dsp \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_basic_dsp \
-partition_name pr_child_partition_1 \
-source_rev_name synth_basic_dsp \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name
a10_pcie_devkit_cvp_normal_basic_dsp_stp \
-partition_name pr_parent_partition \
-source_rev_name synth_basic_dsp_stp \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_gol \
-partition_name pr_parent_partition \
-source_rev_name a10_pcie_devkit_cvp_ddr4_access \
-source_partition pr_parent_partition \
-source_snapshot final

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_gol \
-partition_name pr_child_partition_0 \
-source_rev_name synth_gol \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_gol \
-partition_name pr_child_partition_1 \
-source_rev_name synth_gol \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_normal_gol \
-partition_name pr_parent_partition \
-source_rev_name synth_gol \
-source_partition root_partition \
-source_snapshot synthesized

```

**Note:** All the revision projects must be in the same directory as a10\_pcie\_devkit\_cvp.qpf. Otherwise, update the flow script accordingly.

4. Click **Tools** ► **Tcl Scripts**. The **Tcl Scripts** dialog box appears.
5. Click **Add to Project**, browse and select the a10\_hier\_partial\_reconfig/flow.tcl.
6. Select the a10\_hier\_partial\_reconfig/flow.tcl in the Libraries pane, and click **Run**.

This script runs the synthesis for the four personas. Quartus Prime generates a SRAM Object File (.sof), a Partial-Masked SRAM Object File (.pmsf), and a Raw Binary File (.rbf) for each of the personas.

To run the script from the Quartus Prime command shell, type the following command:

```

quartus_sh -t a10_hier_partial_reconfig/flow.tcl -setup_script
a10_hier_partial_reconfig/setup.tcl

```



## Testing the Reference Design

The reference design provides the following utilities for programming the FPGA board:

- `program-fpga-jtag`
- `program_fpga_pcie`
- `fpga-region-controller`

The design also includes an `example_host_uio` application to communicate with the device, and demonstrate each of the personas.

### `program_fpga_jtag`

Use the `program_fpga_jtag` script to program the entire device (full-chip programming) without any requirement for reboot, and for overwriting the flash memory using JTAG.

`program_over_jtag` performs the following functions:

- Uses the Quartus Prime Programmer to program the device.
- Accepts an SRAM Object File (`.sof`) and configures the target device over a JTAG interface.
- Communicates with the driver to perform the following functions:
  - Disable upstream AER (advanced error reporting)
  - Save state
  - Restore AER
  - Restore state

**Table 6.** `program_fpga_jtag` Command-Line Options

Option	Description
<code>-f=, --file=[&lt;filename&gt;]</code>	Specifies the <code>.sof</code> file name.
<code>-c=, --cable=[&lt;cable number&gt;]</code>	Specifies the programmer cable.
<code>-d=, --device=[&lt;device index&gt;]</code>	Specifies the PCIe device number. For example, <code>-d=0000:03:00.0</code>
<code>-h, --help</code>	Provides help for <code>program_fpga_jtag</code> script.

**Note:** Use the following command to obtain the device index:

```
/sbin/lspci -d1172:
```

For example, consider that the command returns the following output:

```
03:00.0 Class ea00: Altera Corporation Device 5052 (rev 01)
```

The first value is the device index. Prepend 0000 to this value. In this case, your device index is `0000.03:00.0`.



## program\_fpga\_pcie

Use the `program_fpga_pcie` utility to perform partial reconfiguration. The script accepts a `.rbf` file for a given persona. The script performs the following functions:

- Communicates with the driver to remove device sub-drivers, if any
- Communicates with the `fpga-region-controller` script to assert/de-assert freeze
- Writes the `.rbf` to the Partial Reconfiguration Controller IP core
- Re-deploys the sub-drivers, if any, that are required upon successful PR

**Table 7. program\_fpga\_pcie Command-Line Options**

Option	Description
<code>-f, --file=[&lt;filename&gt;]</code>	Specifies the <code>.rbf</code> file name.
<code>-d, --device=[&lt;device index&gt;]</code>	Specifies the PCIe device number. For example, <code>-d 0000:03:00.0</code>
<code>-r, --region=[&lt;region offset&gt;]</code>	Specifies the target FPGA region controller offset, used for freeze bridge enable. This option performs the following operations: <ul style="list-style-type: none"> <li>• Initiates the freeze request exchange</li> <li>• Asserts/deasserts reset for the PR region, during and after partial reconfiguration</li> </ul> For example: <pre>program-fpga-pcie --file=&lt;region.rbf&gt; --device=0000:03:00.0 --region=10000</pre> <p><i>Note:</i> The region offset value must be a hexadecimal number.</p> <p>The PR region is reset during this operation.</p>
<code>-h, --help</code>	Provides help for <code>program_jtag_pcie</code> script.

## example\_host\_uio

The `example_host_uio` module demonstrates the FPGA device access. This application interacts with each persona, verifying the contents and functionality of the personas.

The program requires a PCIe device number, followed by optional parameters of the seed for generating test data, number of tests performed, and verbosity for displaying extra information.

**Table 8. example\_host\_uio Command-Line Options**

Option	Description
<code>-s, --seed [&lt;seed&gt;]</code>	Specifies the seed to use for number generation. Default value is 1.
<code>-d, --device [&lt;device index&gt;]</code>	Specifies the PCIe device number. For example, <code>-d 0000:03:00.0</code>
<i>continued...</i>	



Option	Description
-v, --verbose	Allows you to print additional information during execution.
-n, --iterations	Allows you to specify the iterations for the test you wish to perform.
-h, --help	Provides help for example_host application.

## Signal Tap

The current version of the Quartus Prime Pro Edition software does not support the debug of hierarchical PR regions using Signal Tap Logic Analyzer. However, for the `a10_pcie_devkit_cvp_normal_basic_dsp_stp` revision, Signal Tap support is preserved since that revision compiles a traditional PR design without the parent PR persona. The static region contains the SLD agent, that communicates with an SLD host. You must instantiate the SLD host in the persona that you wish to signal tap. You must include the `.stp` file in the synthesis-only revision of a given persona. Do not include the signal tap file in the base revision, or the `.qsf` file of other personas, if the `.stp` file is specific to a persona.

For complete information on the debug infrastructure using hierarchical hubs, refer to *Debugging Partial Reconfiguration Designs Using Signal Tap Logic Analyzer* section in Volume 3 of the Quartus Prime Pro Edition handbook.

### Related Links

[Debugging Partial Reconfiguration Designs Using Signal Tap Logic Analyzer](#)

## Compiling the Example Applications

The reference design software applications are available in the `software/util` directory. Each application has a respective sub-directory structure, with a corresponding Makefile.

To build the example application:

1. To compile the `example_host` module, type the following from the Linux shell:

```
cd source/util
make
```

This command generates the executable within the sub-directory. For example:

```
./example_host_uio -d 0000:03:00.0 -s 1 -n 100 -v
```

This command uses the device `0000:03:00.0`, seed the input generation with a value of 1, perform 100 iterations, and print more information on the current status.





## Programming the Design Using Example Applications

The following steps describe programming your design using the provided scripts:

1. To program the DDR access persona's .sof file after design compilation, type the following from the Linux shell:

```
program-fpga-jtag -f=a10_pcie_devkit_cvp_ddr4_access.sof -c=1 -d=0000:03:00.0
```

2. To verify the functionality of the design, type the following from the Linux shell:

```
source/util/example_host/example_host_uio -d 0000:03:00.0
```

3. To program the register file child PR persona's .rbf file after design compilation, type the following from the Linux shell:

```
program-fpga-pcie -
f=a10_pcie_devkit_cv_basic_dsp.pr_parent_partition.pr_child_partition_0.rbf -
d=0000:03: 00.0 -r=FFFF
program-fpga-pcie -
f=a10_pcie_devkit_cv_basic_dsp.pr_parent_partition.pr_child_partition_1.rbf -
d=0000:03:00.0 -r=10000
```

4. To verify the functionality of the design, type the following from the Linux shell:

```
software/util/example_host/example_host_uio -d 0000:03:00.0
```

5. To program the basic arithmetic persona's .rbf file after design compilation, type the following from the Linux shell:

```
program-fpga-pcie -
f=a10_pcie_devkit_cv_basic_arithmetic.pr_parent_partition.pr_child_partition_0
.rbf -
d=0000:03:00.0 -r=FFFF
program-fpga-pcie -
f=a10_pcie_devkit_cv_basic_arithmetic.pr_parent_partition.pr_child_partition_1
.rbf -d=0000:03:00.0 -r=10000
```

6. To verify the functionality of the design, type the following from the Linux shell:

```
software/util/example_host/example_host_uio 0000:03:00.0
```

## Extending the Reference Design with Custom Persona

This reference design provides an example template to create your own personas for PR over PCIe. To extend the reference design with your custom persona:



1. Navigate to the `a10_pcie_devkit_cvp_hpr` folder:

```
cd a10_pcie_devkit_cvp_hpr
```

2. Create a copy of the `pr_logic_impl_template.qsf.template` implementation revision file, and the `pr_logic_synth_template.qsf.template` synthesis revision file:

```
cp pr_logic_impl_template.qsf.template <persona_impl_revision_name>.qsf
cp pr_logic_synth_template.qsf.template <persona_synth_revision_name>.qsf
```

3. Create a folder and copy your persona-specific RTL to this folder:

```
mkdir <persona_name>
cp <custom_persona>.sv <persona_name>/
```

4. Your custom top-level entity must match the ports for the `custom_persona` module, defined in the `source/templates/pr_logic_template.sv` file. The following example shows interfacing your design with the Avalon-MM interface, controlled over PCIe register file:

```
module custom_persona #(
    parameter REG_FILE_IO_SIZE = 8
)()
    //clock
    input wire clk,

    //active low reset, defined by hardware
    input wire rst_n,

    //Persona identification register, used by host in host program
    output wire [31:0] persona_id,

    //Host control register, used for control signals.
    input wire [31:0] host_cntrl_register,

    // 8 registers for host -> PR logic communication
    input wire [31:0] host_pr [0:REG_FILE_IO_SIZE-1],

    // 8 Registers for PR logic -> host communication
    output wire [31:0] pr_host [0:REG_FILE_IO_SIZE-1]
);
```

Utilize any of the parallel I/O port (PIO) register files for customization. The `host_pr` register sends the data from the persona to the host machine. The `pr_host` register sends the data from the host machine to the persona.

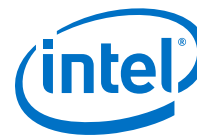
5. In your top-level entity file, specify the persona ID as any 32-bit value:

```
assign persona_id = 32'h0000_aeed;
```

**Note:** The example template uses only 8 bits, but you can specify any value, up to 32 bits.

6. Set the unused output ports of the `pr_host` register to 0:

```
generate
    genvar i;
    //Tying unused output ports to zero.
    for (i = 2; i < REG_FILE_IO_SIZE; i = i + 1) begin
```



```
assign pr_host [i] = 32'b0;
end
endgenerate
```

7. Modify your `persona_synth_revision_name.qsf` to include the following assignments:

```
set_global_assignment -name TOP_LEVEL_ENTITY <custom_persona>
set_global_assignment -name SYSTEMVERILOG_FILE <persona_name>/
<custom_persona>.sv
set_global_assignment -name QSYS_FILE <persona_specific_qsys_file>
set_global_assignment -name IP_FILE <persona_specific_ip_file>
```

8. Update the partial reconfiguration flow script to define your new PR implementation.
9. Update the `a10_pcie_devkit_cvp.qpf` project file to include your synthesis and implementation revisions:

```
PROJECT_REVISION = "<persona_synth_revision_name>"
PROJECT_REVISION = "<persona_impl_revision_name>"
```

10. Compile the revision.

For complete information on adding a custom persona to a PR design, refer to *Adding a New Persona to the Design* section in the *Partially Reconfiguring a Design on Arria 10 GX FPGA Development Board* application note.

### Related Links

[Adding a New Persona to the Design](#)

## Document Revision History

This document has the following revision history.

**Table 9. Document Revision History**

Date	Version	Changes
2017.05.22	17.0.0	Initial Release



## Reference Design Files
















**Table 10. Reference Design Files List**

Type	File/Folder	Description
IP files	<pre> graph TD     source --&gt; static_region     static_region --&gt; ip     ip --&gt; bsp_top           </pre>	Contains the IP files for the Arria 10 External Memory Interfaces IP core, Arria 10 Hard IP for PCI Express IP core, and devkit pins.
	<pre> graph TD     source --&gt; static_region     static_region --&gt; ip     ip --&gt; design_top           </pre>	Contains the IP file for the Arria 10 Partial Reconfiguration Controller IP core, system description ROM, calibration I/O, and all the interface components.
	<pre> graph TD     source --&gt; static_region     static_region --&gt; ip     ip --&gt; pr_subsystem           </pre>	Contains the freeze bridges, the region controller, and the JTAG SLD agent.
	<pre> graph TD     common --&gt; reg_file     reg_file --&gt; ip     ip --&gt; reg_file           </pre>	Contains all the IP files for the register file system, that is common across all personas.
	<pre> graph TD     common --&gt; sld_jtag_host     sld_jtag_host --&gt; ip           </pre>	Contains the JTAG SLD host for the PR region signal tapping. These files are applicable to all the personas.
continued...		






Type	File/Folder	Description
Qsys Pro System Files		<p>Contains the following three Qsys subsystems:</p> <ul style="list-style-type: none"> <li><code>bsp_top.qsys</code>—top-level subsystem, containing the PCIe IP core and the External Memory Interfaces IP core.</li> <li><code>design_top.qsys</code>—static region, encompassing all the Avalon-MM interface logic, reset logic, and PR Region Controller IP core.</li> <li><code>pr_subsystem.qsys</code>—contains all the logic to communicate and interact with the PR region.</li> </ul>
		<p>Contains a simple register file that each persona uses. The file contains the following 32-bit registers—ID, control, 8 input and 8 output registers.</p>
SystemVerilog design files		<p>Contains the top-level wrapper. Also contains the SystemVerilog description for generic components in the three subsystems, and the PR region wrapper.</p>
		<p>Contains all the source files for the basic arithmetic persona.</p>
		<p>Contains all the source files for the DDR4 access persona.</p>
		<p>Contains all the source files for the Game of Life persona.</p>
		<p>Contains all the source files for the parent persona.</p>
		<p>Example personas that use the template for persona configuration. These examples demonstrate integrating a custom persona RTL into the reference design.</p>
Memory files	<code>avalon_config.hex</code>	<p>Used for system description ROM.</p>
Synopsys Design Constraints Files	<code>a10_pcie_devkit_cvp.sdc</code>	<p>Synthesis constraints for the design.</p>
	<code>auxiliary.sdc</code>	<p>Provides exceptions.</p>
continued...		



Type	File/Folder	Description
	 <b>jtag.sdc</b>	Auto-generated constraints from pcie_subsystem_alt_pr.ip file.
Quartus Prime Project File	 <b>a10_pcie_devkit_cvp.qpf</b>	Contains all the revisions.
Quartus Prime Settings Files	 <b>a10_pcie_devkit_cvp.qsf</b>	Base revision settings file for single DDR4 access persona.
	 <b>synth_ddr4_access.qsf</b>	Synthesis revision settings file for DDR4 access persona.
	 <b>synth_basic_dsp.qsf</b>	Synthesis revision settings file for basic DSP persona.
	 <b>synth_basic_arithmetic.qsf</b>	Synthesis revision settings file for basic arithmetic persona.
	 <b>synth_basic_dsp_stp.qsf</b>	Synthesis revision settings file for basic DSP persona with Signal Tap support.
	 <b>synth_gol.qsf</b>	Synthesis revision settings file for Game of Life persona.
	 <b>synth_parent_persona.qsf</b>	Synthesis revision settings file for basic DSP persona with Signal Tap supported.
	 <b>a10_pcie_devkit_cvp_ddr4_access.qsf</b>	Implementation revision settings file for the parent persona with two DDR4 access personas.
	 <b>a10_pcie_devkit_cvp_basic_dsp.qsf</b>	Implementation revision settings file for the parent persona with two basic DSP personas.
	 <b>a10_pcie_devkit_cvp_basic_arithmetic.qsf</b>	Implementation revision settings file for the parent persona with two basic arithmetic personas.
	 <b>a10_pcie_devkit_cvp_gol.qsf</b>	Implementation revision settings file for the parent persona with two Game of Life personas.
	 <b>a10_pcie_devkit_cvp_normal_ddr4_access.qsf</b>	Implementation revision settings file for a single DDR4 access persona without the parent persona.
	 <b>a10_pcie_devkit_cvp_normal_basic_dsp_stp.qsf</b>	Implementation revision settings file for a single basic DSP persona without the parent persona. <i>Note:</i> Signal Tap support is enabled only for this revision, because this revision is a traditional PR compile without the parent persona. Quartus Prime software supports Signal Tap debugging in a traditional PR design.
<b>continued...</b>		



Type	File/Folder	Description
	 <b>a10_pcie_devkit_cvp_normal_basic_arithmetic.qsf</b>	Implementation revision settings file for a single basic arithmetic persona without the parent persona.
	 <b>a10_pcie_devkit_cvp_normal_gol.qsf</b>	Implementation revision settings file for a single Game of Life persona without the parent persona.
PR Setup Script	 <b>setup.tcl</b>	Specifies the PR flow compilation.