



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

Updated for Intel® Quartus® Prime Design Suite: **18.1**



Subscribe

Send Feedback

AN-813 | 2018.09.24

Latest document on the web: [PDF](#) | [HTML](#)



Contents

1. Hierarchical Partial Reconfiguration over PCI Express* Reference Design for Arria® 10 Devices.....	3
1.1. Reference Design Overview.....	4
1.1.1. Clocking Scheme.....	5
1.1.2. Memory Address Mapping.....	6
1.1.3. Floorplanning.....	7
1.2. Getting Started.....	8
1.2.1. Hardware and Software Requirements.....	8
1.2.2. Installing the Arria 10 GX FPGA Development Kit.....	8
1.2.3. Installing the Linux Driver.....	8
1.2.4. Bringing Up the Reference Design.....	10
1.3. Reference Design Components.....	10
1.3.1. BSP Top.....	10
1.4. Compiling the Reference Design.....	14
1.5. Testing the Reference Design.....	15
1.5.1. program_fpga_jtag.....	15
1.5.2. fpga-configure.....	16
1.5.3. example_host_uio.....	17
1.6. Extending the Reference Design with Custom Persona.....	19
1.7. Document Revision History for AN 813: Hierarchical Partial Reconfiguration over PCI Express Reference Design for Arria 10 Devices.....	21
A. Reference Design Files.....	22

1. 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

Intel® Quartus® Prime Pro Edition software v.18.1 introduces a new and simplified compilation flow for partial reconfiguration.

Related Information

- [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 FPGA Development Kit User Guide](#)
For information on Arria 10 GX FPGA development board overview and setup procedure.

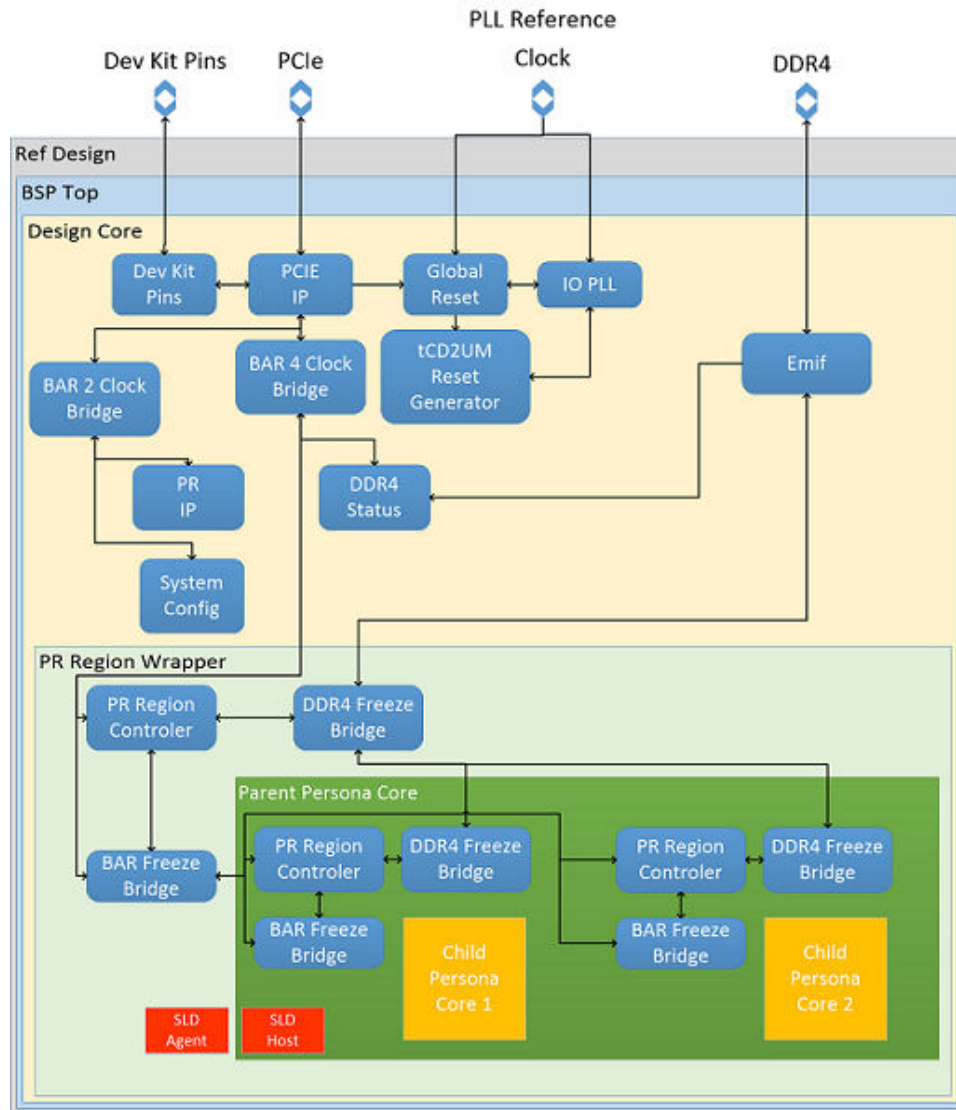
1.1. Reference Design Overview

The reference design consists of the following components:

- `a10_pcie_reference_design`—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® Memory-Mapped (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.



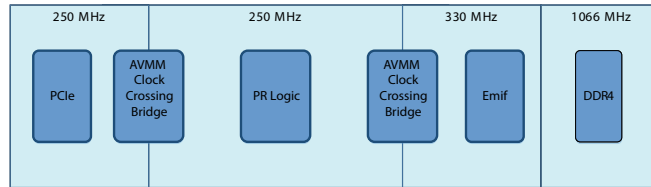
Figure 1. Arria 10 PCIe Reference Design Block Diagram



1.1.1. 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 external memory interface (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



1.1.2. Memory Address Mapping

The PCIe IP core connects to the design core through two Avalon-MM master interfaces. These Avalon-MM master interfaces are base address registers (BARs), BAR 2 and BAR 4.

BAR 2 connects the PR driver to the following components:

- The PR IP core
- The system description ROM

The BAR 4 Avalon-MM connects to the following components:

- The freeze bridges
- The PR region controller
- Up to 8 kilobytes (KB) of memory in the PR region

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 EMIF IP provides status on DDR4 calibration. During initialization, the EMIF IP performs training to reset the DDR4 interface. The EMIF calibration flag reports the training success or the failure to the host. The host takes the necessary action in the event of a DDR4 training failure.

The following table lists the memory address mapping from the EMIF IP to the PR logic:

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

Address Map	Base	End
DDR	0x0000_0000	0x7fff_ffff

The PR logic accesses the 2 gigabyte (GB) DDR4 memory space using an Avalon-MM master interface.

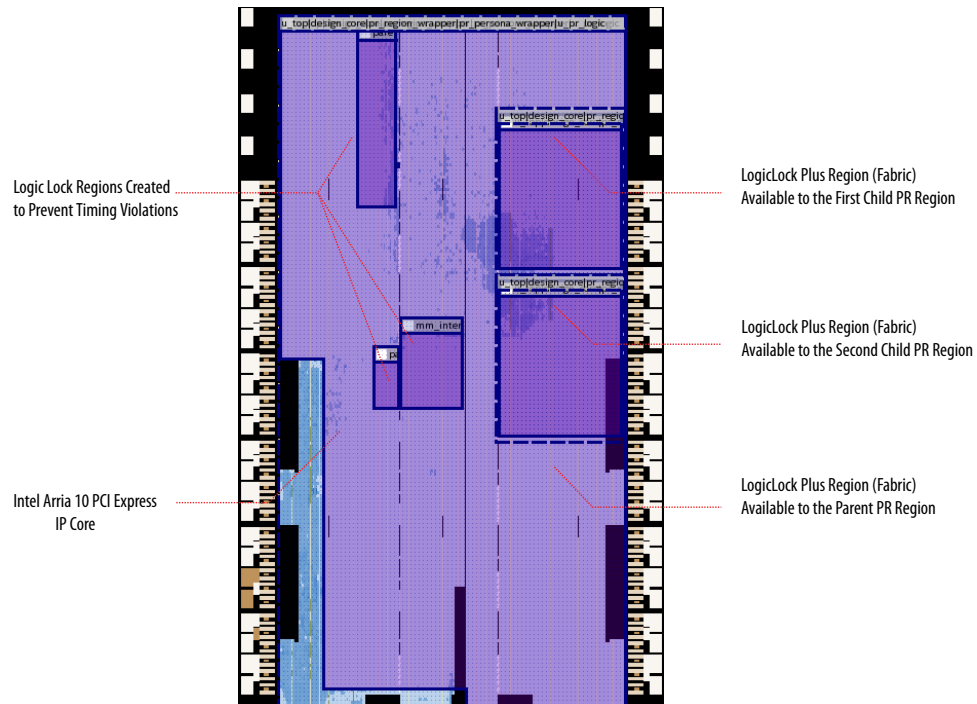


1.1.3. Floorplanning

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 about floorplanning, refer to *Floorplan the Partial Reconfiguration Design* section in Volume 1 of the *Intel Quartus Prime Pro Edition Handbook*.

Related Information

[Floorplan the Partial Reconfiguration Design](#)



1.2. Getting Started

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

1.2.1. Hardware and Software Requirements

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

- Arria 10 GX FPGA development board with the DDR4 module connected to Hi-Lo interface
- Linux Operating System - kernel version 3.10 or above
- Super user access on the host machine
- PCIe slot to plug-in the Arria 10 GX FPGA development board
- Open source driver for this PR over PCIe reference design
- Intel Quartus Prime Pro Edition software v.18.1
- Intel FPGA Download Cable driver

Note:

- Validation testing used CentOS 7 to test the open source driver for this PR over PCIe reference design.
- The Linux driver accompanying this reference design is not a production driver. You must adapt this driver based on your design.

1.2.2. 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 Information

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

1.2.3. 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.

The Linux driver for this design requires the `debugfs`. Run the following command verify that the `debugfs` is available:

```
mount | grep ^debugfs
```

For more information on the `debugfs` file system, please refer to the CentOS documentation.



Important: This driver only supports CentOS 7.

You must install all the prerequisite packages before installing this driver. To install the prerequisite packages, run the following commands:

```
yum groupinstall "Development Tools"
```

```
yum install ncurses-devel
```

```
yum install qt-devel
```

```
yum install hmaccalc zlib-devel binutils-devel elfutils-libelf-devel
```

To install the driver, follow these steps:

1. Ensure the driver source code is available on your machine. The source code is available at the following location:

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

2. To compile all the necessary driver modules, run the following command:

```
make DEVICE="a10"
```

Note: To enable verbose messaging, use the option `VERBOSE=true` with the `make` command.

Ensure that the following three kernel object files are present in the driver source directory after running this command:

- `fpga-mgr-mod.ko`
- `fpga-pcie-mod.ko`

3. To copy the modules to the right location and update the module dependency database, run the following command:

```
sudo make install
```

4. To deploy an instance of the driver for each Intel FPAG device, run the following command:

```
sudo modprobe fpga-pcie-mod
```

5. To verify successful installation of the driver, run the following command:

```
lspci -vvvd1172:
```

Upon successful installation, the resulting output displays the following at the end:

```
Kernel driver in use: fpga-pcie  
Kernel modules: fpga_pcie_mod
```

Note: The above command only works when the design is loaded onto the board and the computer has been power-cycled.

Uninstalling the Linux Driver

If you wish to uninstall the Linux driver, follow these steps:



- Run the following command:

```
sudo modprobe -r fpga-pcie-mod
```

This command stops the driver from executing and deactivates it. However, at this point, rebooting your machine continues to reload the driver.

- To permanently delete the driver, run the following commands:

```
cd /lib/modules/$(uname -r)/extra
rm -rf fpga-pcie-mod.ko
```

1.2.4. Bringing Up the Reference Design

The reference design is available in the following location:

<https://github.com/intel/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 `flash.pl` script.
7. Pass the JTAG cable number of your connected device as an argument to the script (for example, `perl flash.pl 1`).

Running this script configures the device with the contents of the `flash.pof` file. This parallel object file comes directly from the `a10_pcie_devkit_cvp.sof` file present in the `output_files` directory. The `flash.pof` file acts as the base image for the reference design.

Note: Ensure successful compilation of the design before running this script.

8. Power-cycle the host machine.

1.3. Reference Design Components

The reference design contains the following design components.

1.3.1. BSP Top

This Platform Designer system contains all the subsystems of this reference design. The system comprises the following three main components:



- The top-level design
- The PCIe IP
- The DDR4 EMIF IP

The system connects to external pins through the `a10_pcie_ref_design.sv` wrapper.

1.3.1.1. PCI Express IP core

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

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

Table 3. PCI Express IP Core Configuration

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

1.3.1.2. 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.



The following table lists the Arria 10 External Memory Interfaces IP parameters 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

1.3.1.3. 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

1.3.1.3.1. 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 core 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 μ s delay. Until the timer reaches this delay, the PLL is held in reset, deasserting the locked signal. This action freezes the design. Because the PLL locked signal is `ORed` with the PCIe reset, the design also is held in reset. Once the timer reaches 830 μ s, the design functions normally, and enters a known state.

1.3.1.3.2. PR Region Wrapper

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

1.3.1.3.3. 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.



1.3.1.3.4. 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

1.3.1.3.5. Partial Reconfiguration Logic

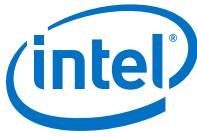
The reference design provides the following personas:

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.



1.4. Compiling the Reference Design

1. To compile the base revision of the reference design, run the following command from the project directory level:

```
quartus_sh --flow compile a10_pcie_devkit_cvp -c a10_pcie_devkit_cvp
```

All the implementation revisions, except the base revision contain the following QDB file partition assignment in their respective .qsf files:

```
set_instance_assignment -name QDB_FILE_PARTITION \  
    output_files/a10_pcie_devkit_cvp_static.qdb -to |
```

This assignment imports the .qdb file representing the reference design static region logic into the subsequent PR persona implementation compile. Each implementation revision also contains one or two ENTITY_REBINDING assignment. This assignment links the hierarchy of the static region and the hierarchy of the PR persona. For example, a10_pcie_devkit_cvp_ddr4_access.qsf contains the following entity rebinding assignment:

```
set_instance_assignment -name ENTITY_REBINDING \  
    parent_persona_top -to \  
    u_top|design_core|pr_region_wrapper|pr_persona_wrapper|u_pr_logic
```

For more information, please refer to the *Partial Reconfiguration Design Flow* section in the *Partial Reconfiguration User Guide*.

2. To compile all the non-HPR personas, run the following commands:

```
quartus_sh --flow compile a10_pcie_devkit_cvp -c \  
    a10_pcie_devkit_cvp_normal_ddr4_access  
quartus_sh --flow compile a10_pcie_devkit_cvp -c \  
    a10_pcie_devkit_cvp_normal_basic_arithmetic  
quartus_sh --flow compile a10_pcie_devkit_cvp -c \  
    a10_pcie_devkit_cvp_normal_basic_dsp  
quartus_sh --flow compile a10_pcie_devkit_cvp -c \  
    a10_pcie_devkit_cvp_normal_gol
```

3. Ensure that the child Logic Lock region is defined in this revision with the same settings as the base revision.
4. To compile the HPR parent persona, run the following command:

```
quartus_sh --flow compile a10_pcie_devkit_cvp -c \  
    a10_pcie_devkit_cvp_ddr4_access
```

Note: After this step, you must manually export the parent PR partition's .qdb file.

All the HPR child implementation revisions contain an additional QDB FILE PARTITION assignment:

```
set_instance_assignment -name QDB_FILE_PARTITION \  
    output_files/a10_pcie_devkit_cvp_ddr4_access_pr_partition_final.qdb -  
    to \  
    u_top|design_core|pr_region_wrapper|pr_persona_wrapper|u_pr_logic
```



This assignment imports the .qdb file representing the HPR parent region into the subsequent into the subsequent HPR child region compilation. Because the HPR child revisions comprise of two child regions, they contain two ENTITY REBINDING assignments.

```
set_instance_assignment -name ENTITY_REBINDING \  
  basic_arithmetic_persona_top -to \  
  u_top|design_core|pr_region_wrapper| \  
  pr_persona_wrapper|u_pr_logic|u0|child_region_0|u_child_pr_logic  
  
set_instance_assignment -name ENTITY_REBINDING \  
  basic_arithmetic_persona_top -to \  
  u_top|design_core|pr_region_wrapper|pr_persona_wrapper| \  
  u_pr_logic|u0|child_region_1|u_child_pr_logic
```

5. To compile the HPR child personas, run the following commands:

```
quartus_sh --flow compile a10_pcie_devkit_cvp -c \  
  a10_pcie_devkit_cvp_basic_arithmetic  
quartus_sh --flow compile a10_pcie_devkit_cvp -c \  
  a10_pcie_devkit_cvp_basic_dsp  
quartus_sh --flow compile a10_pcie_devkit_cvp -c \  
  a10_pcie_devkit_cvp_gol
```

Related Information

- [Partial Reconfiguration Design Flow](#)
- [AN 806: Hierarchical Partial Reconfiguration Tutorial for Intel Arria 10 GX FPGA Development Board](#)

1.5. Testing the Reference Design

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

- program-fpga-jtag
- fpga-configure
- fpga-region-controller

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

1.5.1. program_fpga_jtag

Use the `program_fpga_jtag` script to program the entire device (full-chip programming) without any requirement for reboot.

`program_fpga_jtag` performs the following functions:



- Uses the Intel 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
-f=, --file=[<filename>]	Specifies the .sof file name.
-c=, --cable=[<cable number>]	Specifies the programmer cable.
-i, --index	Specifies the index of the target device in the JTAG chain.
-h, --help	Provides help for program_fpga_jtag 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.

1.5.2. fpga-configure

Use the fpga-configure 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

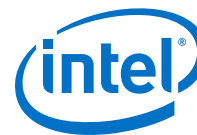
To perform PR over PCIe, run the following command:

```
fpga-configure -p <path-to-rbf> <region_controller_addr>
```

Table 7. fpga-configure Command-Line Options

Option	Description
-p	Performs partial reconfiguration over PCIe programming.
-d	Disables the advanced error reporting on the PCIe link. Advanced error reporting generally reports any critical errors along the PCIe link, directly to the kernel. If the PCIe link is completely disabled, the kernel responds by crashing

continued...



Option	Description
	the system. You must disable advanced error reporting during full chip configuration, as full chip configuration brings down the PCIe link.
-e	Enables the advanced error reporting for the PCIe link. Use this option after full chip configuration to ensure the integrity of the PCIe link.
-r	Prints the contents of a debug ROM within the reference design. Use for debug purposes.

1.5.3. 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
-s=, --seed [<i><seed></i>]	Specifies the seed to use for number generation. Default value is 1.
-v, --verbose	Allows you to print additional information during execution. This option is disabled by default.
-n, --iterations	Allows you to specify the iterations for the test you wish to perform. Default value is 3.
-h, --help	Provides help for <code>example_host</code> application.

Note: Running `example_host_uio` without any command-line arguments uses seed value of 1, 3 iterations.

Signal Tap

The reference design supports signal tapping the PR regions, through hierarchical hubs. This feature facilitates the on-chip debugging of a specific persona. The static region and the parent PR 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.

Follow these guidelines when signal tapping the PR logic:

- Use pre-synthesis registers
- An instance cannot cross partition boundaries
- Do not trigger across partition boundaries

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 Intel Quartus Prime Pro Edition handbook.



Related Information

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

1.5.3.1. 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 -s 1 -n 100 -v
```

This command seeds the input generation with a value of 1, perform 100 iterations, and print more information on the current status.

1.5.3.2. Programming the Design Using Example Applications

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

1. Program the base revision `.sof` file using the programmer. Power cycle the host PC to allow the PCIe link to enumerate. To ensure that the FPGA shows up as a PCIe device, type the following from the Linux shell:

```
lspci -vvvd1172:
```

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

```
./example_host_uio
```

3. To replace the parent PR partition in the design with any of the following single function PR persona, type the following from the Linux shell:

```
fpga-configure -p <rbf file from list> 10000
```

where `<rbf file from list>` is one of the following files:

- `a10_pcie_devkit_cvp_normal_basic_arithmetic.pr_partition`
- `a10_pcie_devkit_cvp_normal_basic_dsp.pr_partition`
- `a10_pcie_devkit_cvp_normal_ddr4_access.pr_partition`
- `a10_pcie_devkit_cvp_normal_gol.pr_partition`

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

```
./example_host_uio
```

5. To program a parent PR partition that contains two child partitions, type the following from the Linux shell:

```
fpga-configure -p a10_pcie_devkit_cvp_ddr4_access.pr_partition.rbf 10000
```

Both child partitions are DDR4 access personas.



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

```
./example_host_uio
```

7. Further, you can reprogram each of the child PR partitions with any combination of personas. The following are the files generated in the `output_files` directory:

- `a10_pcie_devkit_cvp_ddr4_access.pr_partition.pr_child_partition_1.rbf`
- `a10_pcie_devkit_cvp_basic_dsp.pr_partition.pr_child_partition_0.rbf`
- `a10_pcie_devkit_cvp_basic_dsp.pr_partition.pr_child_partition_1.rbf`
- `a10_pcie_devkit_cvp_basic_arithmetic.pr_partition.pr_child_partition_0.rbf`
- `a10_pcie_devkit_cvp_basic_arithmetic.pr_partition.pr_child_partition_1.rbf`
- `a10_pcie_devkit_cvp_gol.pr_partition.pr_child_partition_0.rbf`
- `a10_pcie_devkit_cvp_gol.pr_partition.pr_child_partition_1.rbf`

Each bitstream file is unique to the particular child PR region, and not interchangeable. For example, `*.pr_child_partition_0.rbf` file is only compatible with child PR region 0, and not 1.

8. To program each child region, type the following in the Linux shell:

For child region 0:

```
fpga-configure -p <persona>.pr_partition.pr_child_partition_0.rbf 10
```

For child region 1:

```
fpga-configure -p <persona>.pr_partition.pr_child_partition_1.rbf 20
```

The PR region controller for the larger parent PR region is at address 0x10000. The PR region controller for the two smaller child PR regions is at addresses 0x10 and 0x20 respectively.

1.6. 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:

```
cp pr_logic_impl_template.qsf.template <persona_impl_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_impl_revision_name.qsf` to include the following assignments:

```
set_global_assignment -name TOP_LEVEL_ENTITY a10_pcie_ref_design
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>
set_instance_assignment -name QDB_FILE_PARTITION \
    a10_pcie_devkit_cvp_static.qdb -to | \

set_instance_assignment -name ENTITY_REBINDING \
    <custom_persona_top_level_entity> -to u_top|design_core|
pr_region_wrapper|pr_persona_wrapper|u_pr_logic
```

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 implementation revisions:

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

10. Compile the revision.



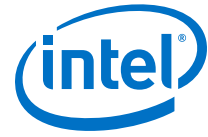
For complete information on adding a custom persona to a PR design, refer to *Adding a New Persona to the Design* section in the *Partially Reconfiguring a Design on Arria 10 GX FPGA Development Board* application note.

Related Information

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

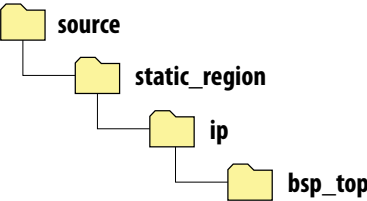
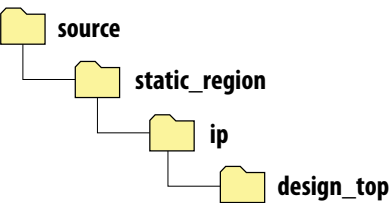
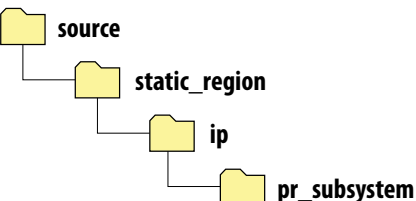
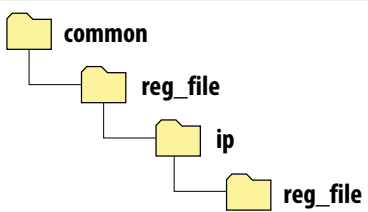
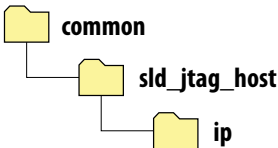
1.7. Document Revision History for AN 813: Hierarchical Partial Reconfiguration over PCI Express Reference Design for Arria 10 Devices

Document Version	Intel Quartus Prime Version	Changes
2018.09.24	18.1.0	<ul style="list-style-type: none"> Updated the <i>Compiling the Reference Design</i> to eliminate the need for manual export of finalized snapshot of the static region. Other minor text edits
2018.07.03	18.0.0	Updated the <i>Compiling the Reference Design</i> with the correct command for .qdb file export.
2018.05.07	18.0.0	<ul style="list-style-type: none"> Compilation flow change Other minor text edits
2017.11.06	17.1.0	<ul style="list-style-type: none"> Updated the <i>Hardware and Software Requirements</i> section with additional reference design requirements Updated the <i>Installing the Linux Driver</i> section with detailed instructions on installing the open source Linux driver Version update Minor text edits
2017.05.22	17.0.0	Initial Release



A. Reference Design Files

Table 9. Reference Design Files List

Type	File/Folder	Description
IP files		Contains the IP files for the Intel Arria 10 External Memory Interfaces IP core, Intel Arria 10 Hard IP for PCI Express IP core, and devkit pins.
		Contains the IP file for the Intel Arria 10 Partial Reconfiguration Controller IP core, system description ROM, calibration I/O, and all the interface components.
		Contains the freeze bridges, the region controller, and the JTAG SLD agent.
		Contains all the IP files for the register file system, that is common across all personas.
		Contains the JTAG SLD host for the PR region signal tapping. These files are applicable to all the personas.
















continued...



Type	File/Folder	Description
		Contains the JTAG SLD agent for the PR region signal tapping. These files are applicable to the static region.
		Contains the IP files for the EMIF logic inside the PR persona.
Platform Designer System Files		<p>Contains the following three Platform Designer (Standard) subsystems:</p> <ul style="list-style-type: none"> • <code>bsp_top.qsys</code>—top-level subsystem, containing the PCIe IP core and the External Memory Interfaces IP core. • <code>design_top.qsys</code>—static region, encompassing all the Avalon-MM interface logic, reset logic, and PR Region Controller IP core. • <code>pr_subsystem.qsys</code>—contains all the logic to communicate and interact with the PR region.
SystemVerilog design files		Contains the top-level wrapper. Also contains the SystemVerilog description for generic components in the three subsystems, and the PR region wrapper.
		Contains all the source files for the basic DSP persona.
		Contains all the source files for the basic arithmetic persona.
		Contains all the source files for the DDR4 access persona.
		Contains all the source files for the Game of Life persona.
		Contains all the source files for the parent persona.

continued...



Type	File/Folder	Description
	 source  templates	Example personas that use the template for persona configuration. These examples demonstrate integrating a custom persona RTL into the reference design.
Memory files	 avalon_config.hex	Used for system description ROM.
Synopsys Design Constraints Files	 a10_pcie_devkit_cvp.sdc	Synthesis constraints for the design.
	 auxiliary.sdc	Provides exceptions.
	 jtag.sdc	Auto-generated constraints from <code>pcie_subsystem_alt_pr.ip</code> file.
Intel Quartus Prime Project File	 a10_pcie_devkit_cvp.qpf	Contains all the revisions.
Intel Quartus Prime Settings Files	 a10_pcie_devkit_cvp.qsf	Base revision settings file for single DDR4 access persona.
	 a10_pcie_devkit_cvp_ddr4_access.qsf	Implementation revision settings file for the parent persona with two DDR4 access personas.
	 a10_pcie_devkit_cvp_basic_dsp.qsf	Implementation revision settings file for the parent persona with two basic DSP personas.
	 a10_pcie_devkit_cvp_basic_arithmetic.qsf	Implementation revision settings file for the parent persona with two basic arithmetic personas.
	 a10_pcie_devkit_cvp_gol.qsf	Implementation revision settings file for the parent persona with two Game of Life personas.
	 a10_pcie_devkit_cvp_normal_ddr4_access.qsf	Implementation revision settings file for a single DDR4 access persona without the parent persona.
	 a10_pcie_devkit_cvp_normal_basic_arithmetic.qsf	Implementation revision settings file for a single basic arithmetic persona without the parent persona.
	 a10_pcie_devkit_cvp_normal_gol.qsf	Implementation revision settings file for a single Game of Life persona without the parent persona.