AN 784: Partial Reconfiguration over PCI Express* Reference Design for Intel® Arria® 10 Devices

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

Contents

1 Partial Reconfiguration over PCI Express* Reference Design for Intel® Arria® 10 Devices

The Partial Reconfiguration (PR) over PCI Express* (PCIe*) reference design demonstrates reconfiguring the FPGA fabric through the PCIe link in Intel® Arria® 10 devices. This reference design runs on a Linux system with the Intel 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.

Intel 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.

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

Related Links

- Creating a Partial Reconfiguration Design
  For complete information on the partial reconfiguration design flow.

- Partially Reconfiguring a Design on Intel Arria 10 GX FPGA Development Board
  For a step-by-step tutorial on creating a partial reconfiguration design.

- Intel Arria 10 CvP Initialization and Partial Reconfiguration via Protocol User Guide
  For complete information on the Configuration via Protocol (CvP) configuration scheme.

- Intel Arria 10 FPGA Development Kit User Guide
  For information on Intel Arria 10 GX FPGA development board overview and setup procedure.

ISO 9001:2008 Registered

1.1 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.   Intel Arria 10 PCIe Reference Design Block Diagram**
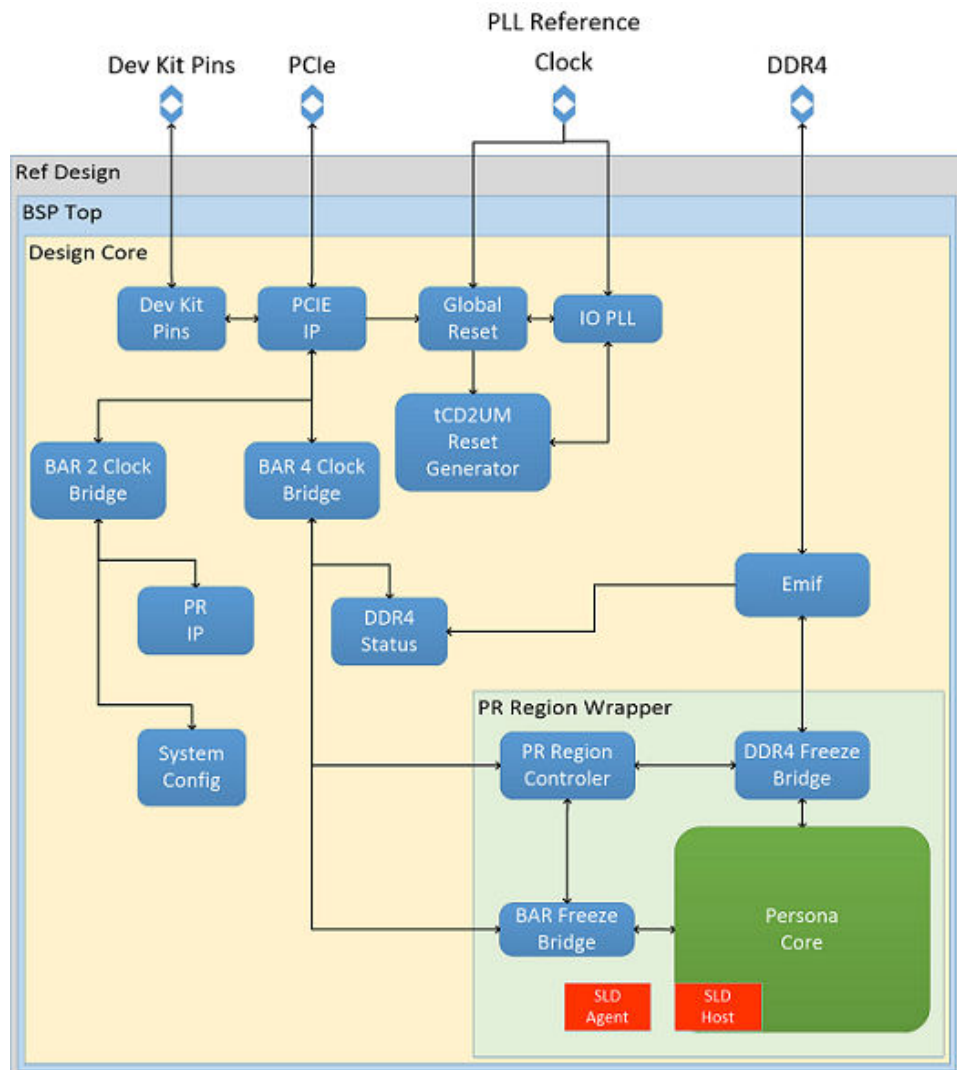
1.1.1 Clocking Scheme

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 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
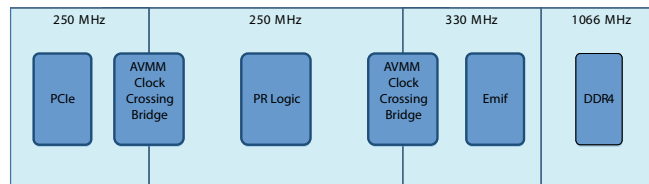
1.1.2 Memory Address Mapping

The PCIe IP core connects to the design core through two 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.

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.

Figure 3.   Reference Design Floorplan
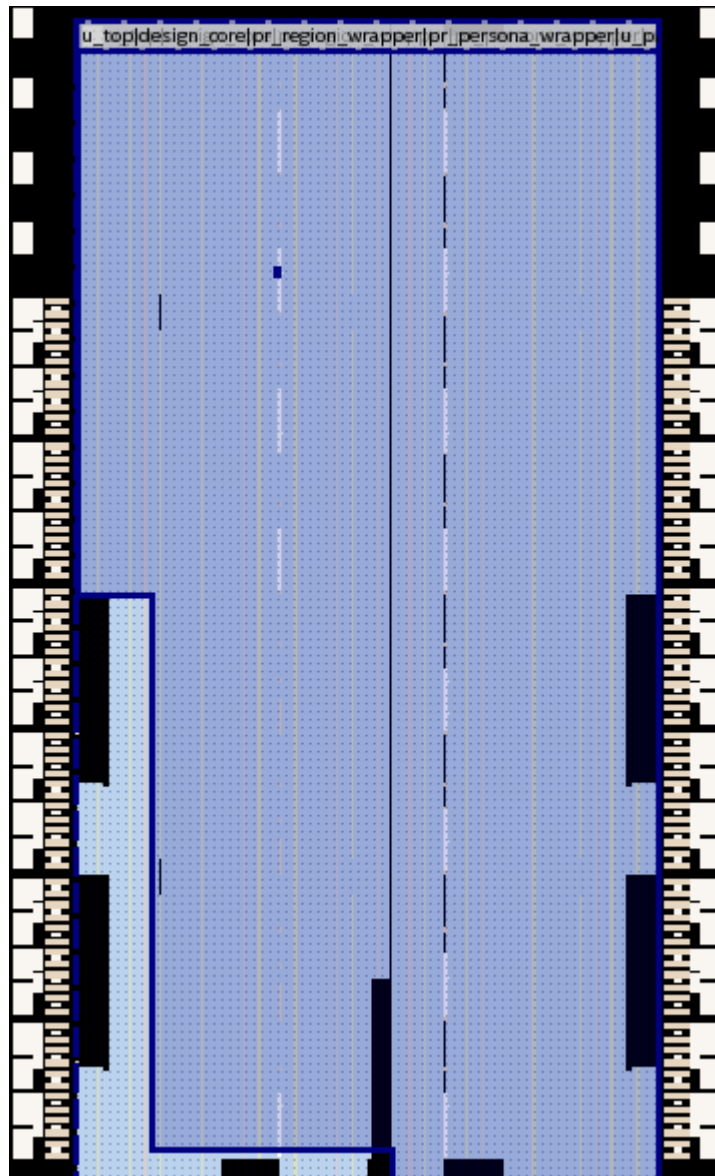
For more information on floorplanning, refer to *Floorplan the Partial Reconfiguration Design* section in Volume 1 of the *Intel Quartus Prime Pro Edition Handbook*.

Related Links

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:

- Intel 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 Intel Arria 10 GX FPGA development board

- Open source driver for this PR over PCIe reference design

- Intel Quartus Prime Pro Edition software v.17.1

- Intel FPGA Download Cable driver

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

1.2.2 Installing the Intel Arria 10 GX FPGA Development Kit

For complete instructions on installing and powering the Intel Arria 10 GX FPGA development board in your Linux system, refer to *Intel 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

- Intel Arria 10 FPGA Development Kit User Guide

- Bringing Up the Reference Design on page 9

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 that the debugfs to be mounted. To verify if the debugfs is mounted, run the following command:

```
mount | grep ^debugfs
```

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

Important: CentOS 7 is the only supported OS by this driver.

You must install all the per-requisite packages before installing this driver. To install the per-requisite 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 copied to your machine. The source code can be found 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
   ```

   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 has been loaded onto the board and the computer has been power-cycled.

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` folder to the home directory in your Linux system.

To bring up the reference design on the board:

1. Plug-in the Intel 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/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 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.

1.3.1.1 PCI Express IP core

The Intel 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:
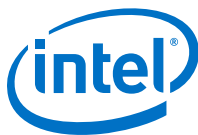
Table 3.   PCI Express IP Core Configuration

Setting	Parameter	Value
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 Intel Arria 10 DDR4 External Memory Interfaces IP Core

The `ddr4_emif` logic includes the Intel 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 Intel Arria 10 External Memory Interfaces IP core that are different from the Intel Arria 10 GX FPGA Development Kit with DDR4 HILO preset settings:

Table 4.   Intel 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 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 µs 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 µs, 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 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.

1.3.1.3.3 Intel Arria 10 Partial Reconfiguration Region Controller IP Core

Use the Intel 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 Intel 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

1.3.1.3.4 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.

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

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

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

   ```
   quartus_sh --write_flow_template -flow a10_partial_reconfig
   ```

   Intel Quartus Prime generates the `a10_partial_reconfig/flow.tcl` file.

2. Rename the generated `a10_partial_reconfig/setup.tcl.example` to `a10_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 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_partition\
   -source_rev_name synth_ddr4_access

   define_pr_impl_partition -impl_rev_name
   a10_pcie_devkit_cvp_basic_arithmetic \
   ```

```
    -partition_name pr_partition \
    -source_rev_name  synth_basic_arithmetic

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_basic_dsp \
    -partition_name pr_partition \
    -source_rev_name  synth_basic_dsp

define_pr_impl_partition -impl_rev_name a10_pcie_devkit_cvp_gol \
    -partition_name pr_partition \
    -source_rev_name  synth_gol
```

> *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_partial_reconfig/flow.tcl`.

6. Select the `a10_partial_reconfig/flow.tcl` in the Libraries pane, and click **Run**.

   This script runs the synthesis for the four personas. Intel 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 Intel Quartus Prime command shell, type the following command:

```
quartus_sh -t a10_partial_reconfig/flow.tcl -setup_script
a10_partial_reconfig/setup.tcl
```

1.5 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.

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_over_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.
`-d=, --device=[<device index>]`	Specifies the PCIe device number. For example, `-d=0000:03:00.0`
`-h, --help`	Provides help for `program_fpga_jtag` script.

Note: Use the following command to the 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 `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
`-f=, --file=[<filename>]`	Specifies the `.rbf` file name.
`-d=, --device=[<device index>]`	Specifies the PCIe device number. For example, `-d 0000:03:00.0`
`-r=, --region=[<region offset>]`	Specifies the target FPGA region controller offset, used for freeze bridge enable. This option performs the following operations:

continued...

Option	Description
	• Initiates the freeze request exchange • Asserts/deasserts reset for the PR region, during and after partial reconfiguration For example: ```program-fpga-pcie --file=<region.rbf> --device=0000:03:00.0 --region=10000``` *Note:* The region offset value must be a hexadecimal number. The PR region is reset during this operation.
`-h, --help`	Provides help for `program_jtag_pcie` script.

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 [<seed>]`	Specifies the seed to use for number generation. Default value is 1.
`-d, --device [<device index>]`	Specifies the PCIe device number. For example, `-d 0000:03:00.0`
`-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 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 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 Links

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 -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.

1.5.3.2 Programming the Design Using Example Applications

Assume that the device is `0000:03:00.0` and the cable number is 1.
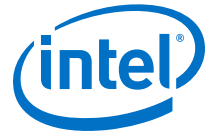
The following steps describe programming your design using the provided scripts:

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 persona's `.rbf` file after design compilation, type the following from the Linux shell:

   ```
   program-fpga-pcie -f=a10_pcie_devkit_cvp_basic_dsp.pr_partition.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_cvp_basic_arithmetic.pr_partition.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
   ```

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` folder:

   ```
   cd a10_pcie_devkit_cvp
   ```

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 Intel Arria 10 GX FPGA Development Board* application note.

Related Links

Adding a New Persona to the Design

1.7 Document Revision History

This document has the following revision history.

Table 9.　　Document Revision History

Document Version	Software Version	Changes
2017.11.06	17.1.0	• Updated the *Hardware and Software Requirements* section with additional reference design requirements • Updated the *Installing the Linux Driver* section with detailed instructions on installing the open source Linux driver • Version update • Minor text edits
2017.05.08	17.0.0	• Reference design flow updated • New persona - Game of Life added
2016.10.31	16.1.0	Initial Release
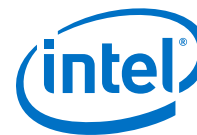
A Reference Design Files

Table 10. Reference Design Files List

Type	File/Folder	Description
IP files	📁 source 　📁 static_region 　　📁 ip 　　　📁 bsp_top	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.
	📁 source 　📁 static_region 　　📁 ip 　　　📁 design_top	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.
	📁 source 　📁 static_region 　　📁 ip 　　　📁 pr_subsystem	Contains the freeze bridges, the region controller, and the JTAG SLD agent.
	📁 common 　📁 reg_file 　　📁 ip 　　　📁 reg_file	Contains all the IP files for the register file system, that is common across all personas.
	📁 common 　📁 sld_jtag_host 　　📁 ip	Contains the JTAG SLD host for the PR region signal tapping. These files are applicable to all the personas.

continued...

Type	File/Folder	Description
Platform Designer System Files	📁 **source** └ 📁 **static_region** ├ 📄 **bsp_top.qsys** ├ 📄 **design_top.qsys** └ 📄 **pr_subsystem.qsys**	Contains the following three Platform Designer (Standard) subsystems: • `bsp_top.qsys`—top-level subsystem, containing the PCIe IP core and the External Memory Interfaces IP core. • `design_top.qsys`—static region, encompassing all the Avalon-MM interface logic, reset logic, and PR Region Controller IP core. • `pr_subsystem.qsys`—contains all the logic to communicate and interact with the PR region.
	📁 **source** └ 📁 **static_region** └ 📄 **reg_file.qsys**	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.
SystemVerilog design files	📁 **source** └ 📁 **static_region**	Contains the top-level wrapper. Also contains the SystemVerilog description for generic components in the three subsystems, and the PR region wrapper.
	📁 **source** └ 📁 **basic_arithmetic_persona**	Contains all the source files for the basic arithmetic persona.
	📁 **source** └ 📁 **ddr_access_persona**	Contains all the source files for the DDR4 access persona.
	📁 **source** └ 📁 **gol_persona**	Contains all the source files for the Game of Life persona.
	📁 **template_example**	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 `pcie_subsystem_alt_pr.ip` 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 DDR4 access persona.

continued...

Type	File/Folder	Description
	synth_ddr4_access.qsf	Synthesis revision settings file for DDR4 access persona.
	synth_basic_dsp.qsf	Synthesis revision settings file for basic DSP persona.
	synth_basic_arithmetic.qsf	Synthesis revision settings file for basic arithmetic persona.
	synth_gol.qsf	Synthesis revision settings file for Game of Life persona.
	a10_pcie_devkit_cvp_ddr4_access.qsf	Implementation revision settings file for DDR4 access persona.
	a10_pcie_devkit_cvp_basic_dsp.qsf	Implementation revision settings file for basic DSP persona.
	a10_pcie_devkit_cvp_basic_arithmetic.qsf	Implementation revision settings file for basic arithmetic persona.
	a10_pcie_devkit_cvp_gol.qsf	Implementation revision settings file for Game of Life persona.
PR Setup Script	setup.tcl	Specifies the PR flow compilation.