Partial Reconfiguration Solutions IP User Guide

UG-20066
2017.05.08

Last updated for Intel® Quartus® Prime Design Suite: 17.0

Subscribe
Send Feedback

Contents

1 Introduction

Partial Reconfiguration (PR) allows dynamic reprogramming of a portion of an Intel® FPGA, while the remainder of the FPGA continues to operate. The Intel Quartus® Prime Pro Edition software includes a suite of PR IP cores that simplify partial reconfiguration implementation.

Figure 1. Partial Reconfiguration IP Components (Internal Host)

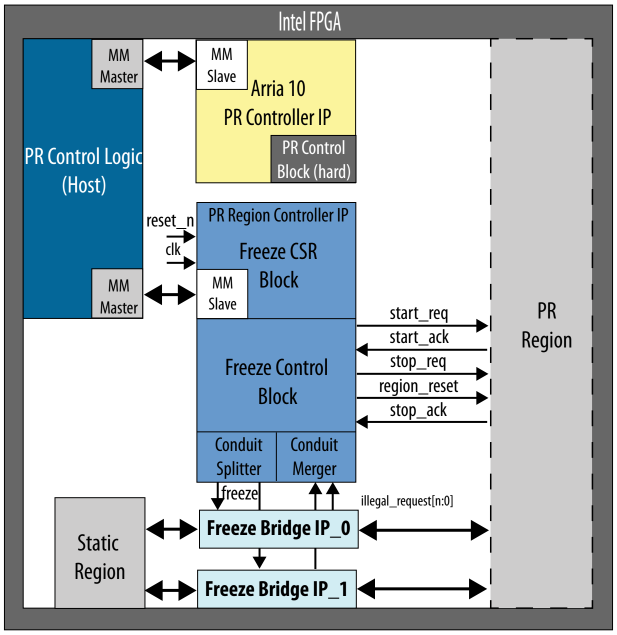


Table 1. Partial Reconfiguration IP Cores

IP	Description	Usage
Intel Arria® 10 Partial Reconfiguration Controller	Dedicated IP component that sends the partial reconfiguration bitstream to the FPGA. The PR bitstream performs reconfiguration by adjusting RAM (CRAM) bits in the FPGA.	One instance per FPGA.
Partial Reconfiguration Region Controller	Provides a standard interface to the to the PR Control block that controls handshaking with the PR region. Ensures that PR region transactions complete before freeze of the interface.	One instance per PR region.
Avalon®-MM Partial Reconfiguration Freeze Bridge	Provides freeze capabilities to the PR region for Avalon Memory Mapped (Avalon-MM) interfaces.	One instance for each interface in each PR region.
Avalon-ST Partial Reconfiguration Freeze Bridge	Provides freeze capabilities to the PR region for Avalon Streaming (Avalon-ST) interfaces.	One instance for each interface in each PR region.

Instantiate one or more of these IP cores to simplify PR implementation. Alternatively, create your own controller handshake, and freeze logic that interfaces with the PR region.

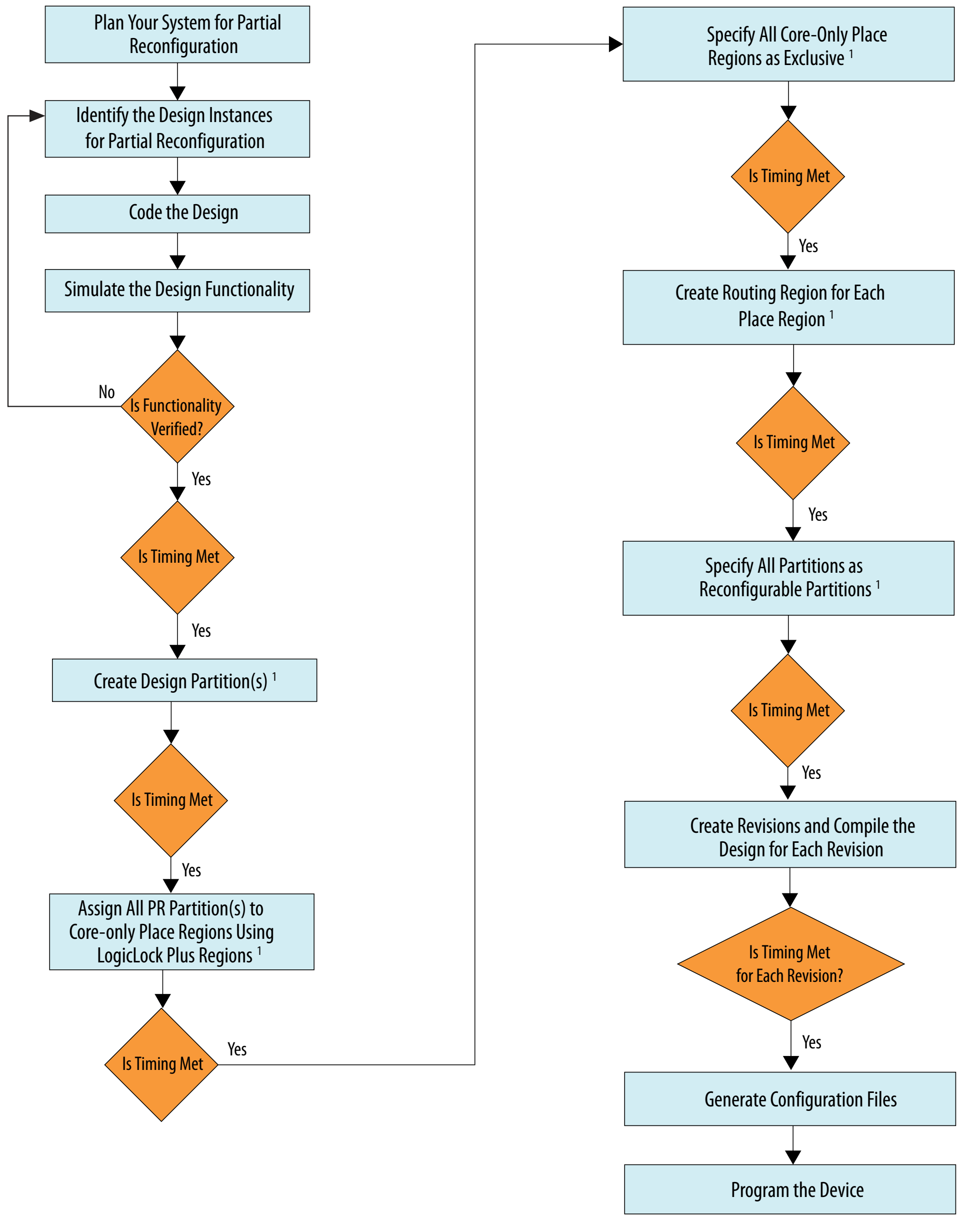
1.1 Partial Reconfiguration Concepts

Implementing a PR design requires understanding of the FPGA device capabilities and the Quartus Prime compilation flow. The following table defines common PR terminology.

Table 2. Partial Reconfiguration Terminology

Term	Description
PR partition	Design partition you designate for PR. A PR project can contain one or more partially reconfigurable PR partitions.
PR region	An area in the FPGA that you associate with a partially reconfigurable partition. A PR region contains the core locations of the device you wish to reconfigure. A device can contain more than one PR region. A PR region can be core-only, such as LAB, RAM, or DSP.
Static region	All areas outside the PR regions in your project. You associate the static region with the top-level partition of the design. The static region contains both the core and periphery locations of the device.
PR persona	A specific PR partition implementation in a PR region. A PR region can contain multiple personas. Static regions contain only one persona.
PR control block	A dedicated FPGA block. The PR control block processes the PR requests, handshake protocols, and verifies the cyclic redundancy check (CRC).
PR host	The system for coordinating PR. The PR host communicates with the PR control block. Implement the PR host within the FPGA (internal PR host) or in a chip or microprocessor (external PR host).

Figure 2. Partial Reconfiguration Design Flow


Plan Your System for Partial Reconfiguration
Identify the Design Instances for Partial Reconfiguration
Code the Design
Simulate the Design Functionality
No
Is Functionality Verified?
Yes
Is Timing Met
Yes
Create Design Partition(s) 1
Is Timing Met
Yes
Assign All PR Partition(s) to Core-only Place Regions Using LogicLock Plus Regions 1
Is Timing Met
Yes
Specify All Core-Only Place Regions as Exclusive 1
Is Timing Met
Yes
Create Routing Region for Each Place Region 1
Is Timing Met
Yes
Specify All Partitions as Reconfigurable Partitions 1
Is Timing Met
Yes
Create Revisions and Compile the Design for Each Revision
Is Timing Met for Each Revision?
Yes
Generate Configuration Files
Program the Device

(1) Recommended to compile the base revision before verifying timing closure

Related Links

Creating a Partial Reconfiguration Design

1.2 Internal or External PR Host Configuration

Perform partial reconfiguration through either an internal host residing in the core resources, or with an external host via dedicated device pins. Use of an internal host stores of all PR host logic on the FPGA device, rather than on an external device. The PR host interfaces with the control block through simple handshaking and data transfer.

Figure 3. Partial Reconfiguration with HPS (Internal Host)

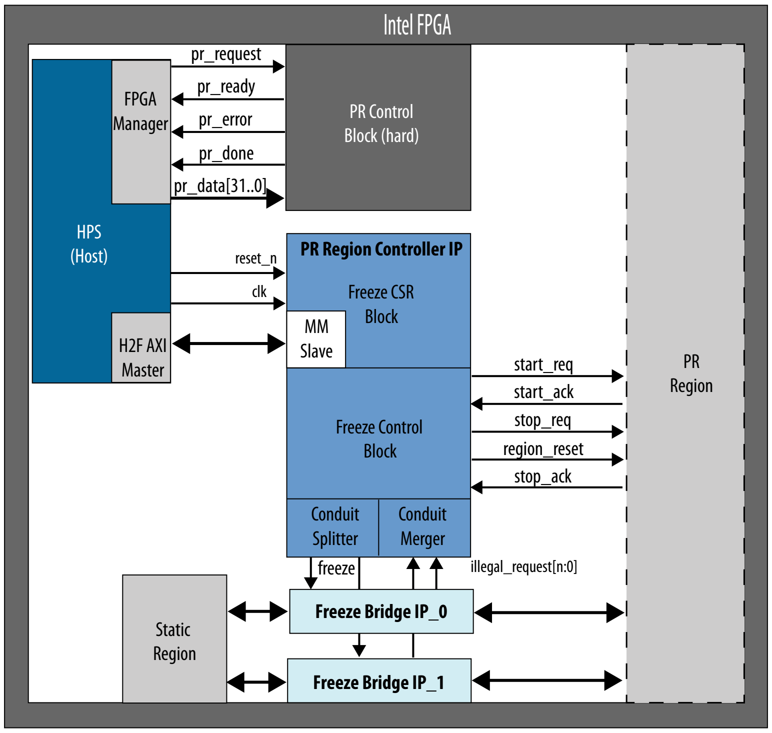
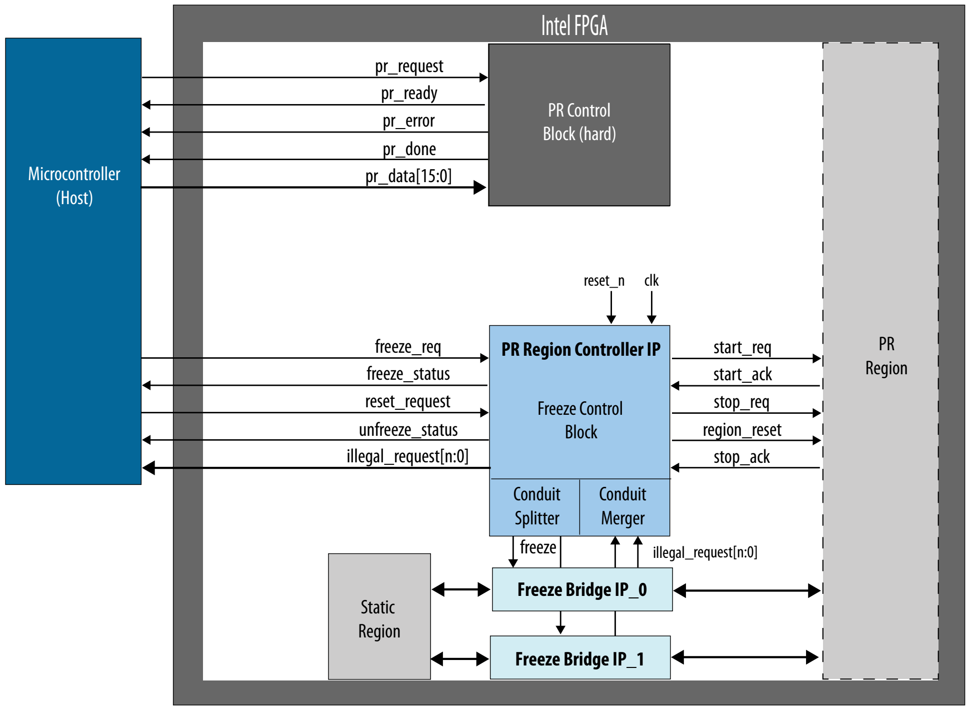


Figure 4. Partial Reconfiguration with Microcontroller (External Host)



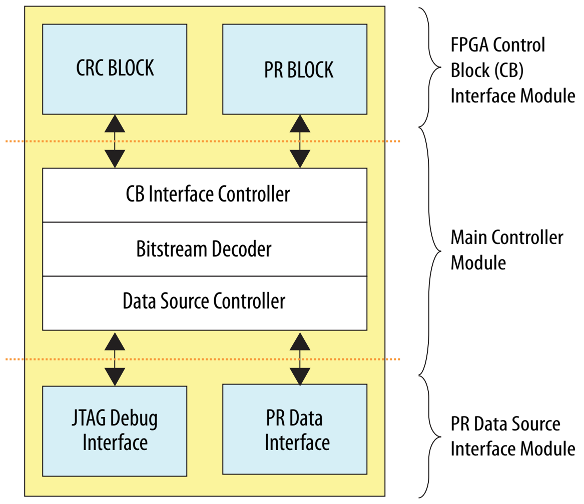
Related Links

AN 784: Partial Reconfiguration over Reference Design for Arria 10 Devices

2 Arria 10 Partial Reconfiguration Controller

The Arria 10 Partial Reconfiguration Controller IP core provides a standard interface to the partial reconfiguration functionality in the PR control block. Use this IP core to avoid manually instantiating a PR control block interface.

Figure 5. Arria 10 Partial Reconfiguration Controller



2.1 Partial Reconfiguration Controller Avalon-MM Slave Interface

The Partial Reconfiguration Controller IP core provides an Avalon-MM slave interface to read and write to PR configuration registers.

Table 3. Data/CSR Memory Map Format

Name	Address Offset	Access	Description
PR_DATA	0x0	Write	Every data `write` to this address indicates this bitstream was sent to the IP core. Performing a `read` on this address returns all 0's.
PR_CSR	0x1	Read or Write	Control and status registers.
`Version Register`	0x02	Read-Only	Read-only SW version register. Register is currently 0xAA500003
PR Bitstream ID	0x03	Read-Only	Read-only PR POF ID register

Table 4. PR_CSR Control and Status Registers

Bit Offset	Description
0	Read and write control register for `pr_start` signal. Refer to Arria 10 Partial Reconfiguration Controller Interface Ports on page 16 for details on the `pr_start` signal. `pr_start = PR_CSR[0]` `PR_CSR[0]` is de-asserted to value "0" by the IP core automatically, one clock cycle after it is asserted. This streamlines the flow to avoid manual assertion and de-assertion of this register to control `pr_start` signal.
1	Reserved.
2-4	Read-only status register for `status[2:0]` signal. `PR_CSR[4:2] = status[2:0]` Refer to Arria 10 Partial Reconfiguration Controller Interface Ports on page 16 for details on the status signals.
5	Read and clear bit for interrupt. If the interrupt interface is enabled, reading this bit returns the value of the `irq` signal. Writing a "1" clears the interrupt. If the interrupt interface is disabled, reading this bit always returns a value of "0".
6-31	Reserved. Depends on the Avalon-MM data bus width.

Related Links

Avalon Interface Specifications

2.1.2 Avalon-MM Interface Partial Reconfiguration Sequence

Partial reconfiguration occurs through the Avalon-MM slave interface in the following sequence:

1. Avalon-MM master component writes `0x01` to IP address offset 0x1 to trigger PR operation.
2. Optionally poll the status register until PR Operation in Progress. Not polling results in `waitrequest` on first word written.
3. Avalon-MM master component writes PR bitstream to IP address offset 0x0 until all the PR bitstream is written. When enhanced decompression is on, waitrequest is activated throughout the PR operation. Ensure that your master can handle `waitrequest` from the slave interface.
4. Avalon-MM master component reads the data from IP address offset 0x1 to check the `status[2:0]` value. Optionally, the Avalon-MM master component can read the `status[2:0]` of this IP during a PR operation to detect any early failure, for example, PR_ERROR.

2.2 Partial Reconfiguration Controller Interrupt Interface

If you enable the Avalon Memory Mapped Slave interface, you can use the optional interrupt interface.

The IP core asserts `irq` during the following events:

Table 5. Interrupt Interface Events

Status Code	Event
3'b001	PR_ERROR occurred.
3'b010	CRC_ERROR occurred.
3'b011	The IP core detects an incompatible bitstream.
3'b101	The result of a successful PR operation.

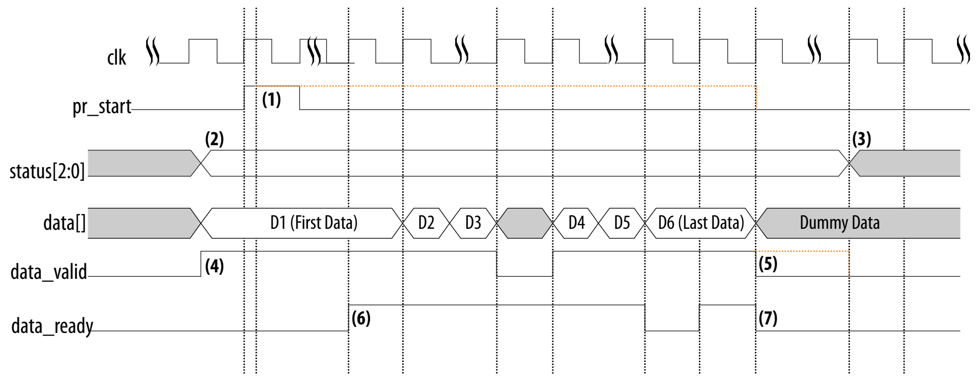
After `irq` asserts, the master performs one or more of the following:

- Query for the status of the PR IP core; PR_CSR`[4:2]`
- Carry out some action, such as error reporting.
- Once the interrupt is serviced, clear the interrupt by writing a "1" to PR_CSR`[5]`.

2.3 Partial Reconfiguration Controller Timing Specifications

The following timing diagram illustrates a successful PR operation. The `status[2:0]` output signal indicates whether the operations passes or fails. The PR operation initiates upon assertion of the `pr_start` signal. Monitor the `status[]` signal to detect the end of the PR operation.

Figure 6. Arria 10 Partial Reconfiguration Controller Timing Specifications



The following notes correspond to the numbers in the timing diagram:

1. Assert `pr_start` signal high for a minimum of one clock cycle to initiate PR. Deassert `pr_start` before sending the last data.
2. `status[]` signal resets when `pr_start` asserts. This signal changes during a PR operation if CRC_ERROR, PR_ERROR, or bitstream incompatibility error occurs.
3. `status[]` signal changes after a PR operation if CRC_ERROR asserts and no error occurs during the previous PR operation.
4. There is no requirement to assert the `data_valid` signal at the same time as the `pr_start` signal. Provide the `data[]` and assert `data_valid` when appropriate.

5. Either drive the `data_valid` signal low after sending the last data, or continue to assert `data_valid` high with dummy data on `data[]` until the end of PR is read from `status[]`.
6. `data[]` is transferred only when `data_valid` and `data_ready` are asserted on the same cycle. Do not drive new data on the data bus, when both `data_valid` and `data_ready` are not high.
7. The `data_ready` signal is driven low once the PR IP Controller core receives the last data, or when the PR IP Controller cannot accept data.

2.4 PR Bitstream Compression and Encryption

You can compress and encrypt the base bitstream and the PR bitstream for your PR project using options available in the Quartus Prime software.

Compress the base and PR programming bitstreams independently, based on your design requirements. When encrypting only the base image, specify whether or not to encrypt the PR images. The following guidelines apply to PR bitstream compression and encryption:

- You can encrypt the base and PR image independently. In addition, you can use different encryption keys in each case, For example, you can use a non-volatile encryption key for the base image, and a volatile encryption key for the PR image.
- Refer to Table 6 on page 13 to ensure the correct CD ratio setting for encryption or compression.

For partial reconfiguration with the PR Controller IP core, specify enhanced compression by turning on the **Enhanced compression** option when specifying the parameters in the **IP Catalog** or Qsys Pro parameter editors.

Note: You cannot use hardware or enhanced compression together with encryption simultaneously.

2.4.1 Generating an Encrypted PR Bitstream

To partially reconfigure your device with encrypted bitstream:

1. Create a 256-bit key file (`.key`).
2. To generate the key programming file (`.ekp`) from the Quartus Prime shell, type the following command:

```
quartus_cpf --key <keyfile>:<keyid> <base_sof_file> <output_ekp_file>
```

   For example:

```
quartus_cpf --key my_key.key:key1 base.sof key.ekp
```

3. To generate the encrypted PR bitstream (`.rbf`), run the following command:

```
quartus_cpf -c <pr_pmsf_file> <pr_rbf_file>
qcrypt -e --keyfile=<keyfile> --keyname=<keyid> –lockto=<qlk file> --
keystore=<battery|OTP> <pr_rbf_file> <pr_encrypted_rbf_file>
```

- `lockto`—specifies the encryption lock.
- `keystore`—specifies the volatile key (battery) or the non-volatile key (OTP).

For example:

```
quartus_cpf -c top_v1.pr_region.pmsf top_v1.pr_region.rbf
qcrypt -e --keyfile=my_key.key --keyname=key1 --keystore=battery
top_v1.pr_region.rbf top_v1_encrypted.rbf
```

4. To program the key file as volatile key (default) into the device, type the following command:

```
quartus_pgm -m jtag -o P;<output_ekp_file>
```

For example:

```
quartus_pgm -m jtag -o P;key.ekp
```

5. To program the base image into the device, type the following command:

```
quartus_pgm -m jtag -o P;<base_sof_file>
```

For example:

```
quartus_pgm -m jtag -o P;base.sof
```

6. To partially reconfigure the device with the encrypted bitstream, type the following command:

```
quartus_pgm -m jtag --pr <output_encrypted_rbf_file>
```

For example:

```
quartus_pgm -m jtag --pr top_v1_encrypted.rbf
```

For more information on the design security features in Arria 10 devices, refer to *Using the Design Security Features in Altera FPGAs*.

Related Links

Using the Design Security Features in Altera FPGAs

2.4.2 CD Ratio for Bitstream Encryption and Compression

When instantiating the Partial Reconfiguration Controller IP core in your Arria 10 design, you cannot use both data compression and encryption simultaneously. Enhanced decompression uses the same Clock-to-Data (CD) ratio as plain bitstreams (that is, with both encryption and compression off).

The following table lists the valid combinations of bitstream encryption and compression. when enhance compression is enabled, always refer to x16 data width. If you use compression and enhanced compression together, the CD ratio follows the compression bitstream - 4. If you use plain and enhanced compression together, the CD ratio follows the plain bitstream - 1.

Table 6. Valid combinations and CD Ratio for Bitstream Encryption and Compression

Configuration Data Width	AES Encryption	Basic Compression	CD Ratio
x8	Off	Off	1
	Off	On	2
	On	Off	1
x16	Off	Off	1
	Off	On	4
	On	Off	2
x32	Off	Off	1
	Off	On	8
	On	Off	4

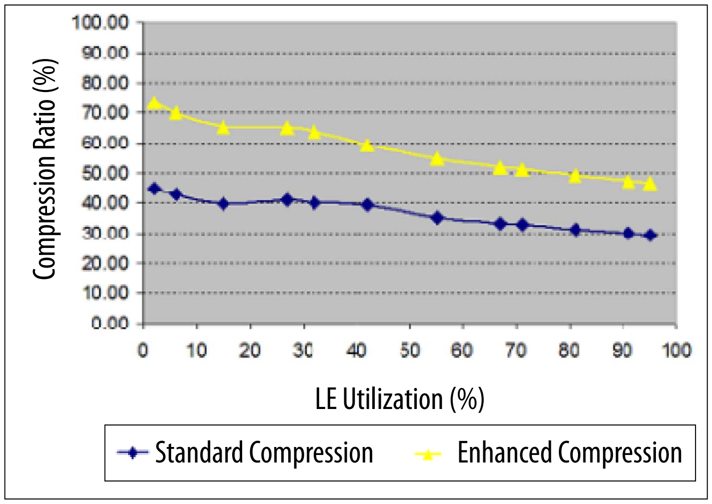
The CD ratio for the Partial Reconfiguration IP core must be exact for the bitstream type. The CD ratio for plain RBF must be 1. The CD ratio for compressed RBF must be 2, 4 or 8, depending on the width. Do not specify the CD ratio as the necessary minimum to support different bitstream types.

2.4.3 Data Compression Comparison

Standard compression results in a 30-45% decrease in RBF size. Use of the enhanced data compression algorithm results in 55-75% decrease in RBF size. The algorithm increases the compression at the expense of additional core area required to implement the compression algorithm.

The following figure shows the compression ratio comparison across PR designs with varying degrees of Logic Element (LE):

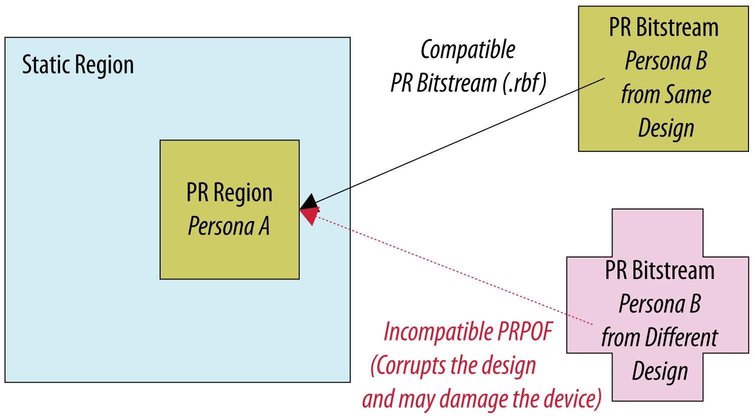
Figure 7. Compression Ratio Comparison between Standard Compression and Enhanced Compression



2.4.4 Bitstream Compatibility Check

Turn on the **Enable bitstream compatibility check** when instantiating the PR Controller IP core from either Qsys Pro or the IP Catalog for supported devices. The software then verifies the partial reconfiguration PR Bitstream file (`.rbf`). If an incompatible bitstream is detected, the PR operation aborts and the `status` output reports an error. The PR `.pof` ID encodes as the 71st word of the PR bitstream.

Figure 8. Bitstream Compatibility Check



This option prevents you from accidentally corrupting the static region of your design with a bitstream from an incompatible `.rbf` and risking damage to the chip you are programming.

When you turn on **Enable bitstream compatibility check**, the PR Controller IP core creates a **PR bitstream ID** and displays the bitstream ID in the configuration dialog box.

2.5 Arria 10 Partial Reconfiguration Controller Parameters

The Arria 10 Partial Reconfiguration Controller IP core supports customization of the following parameters.

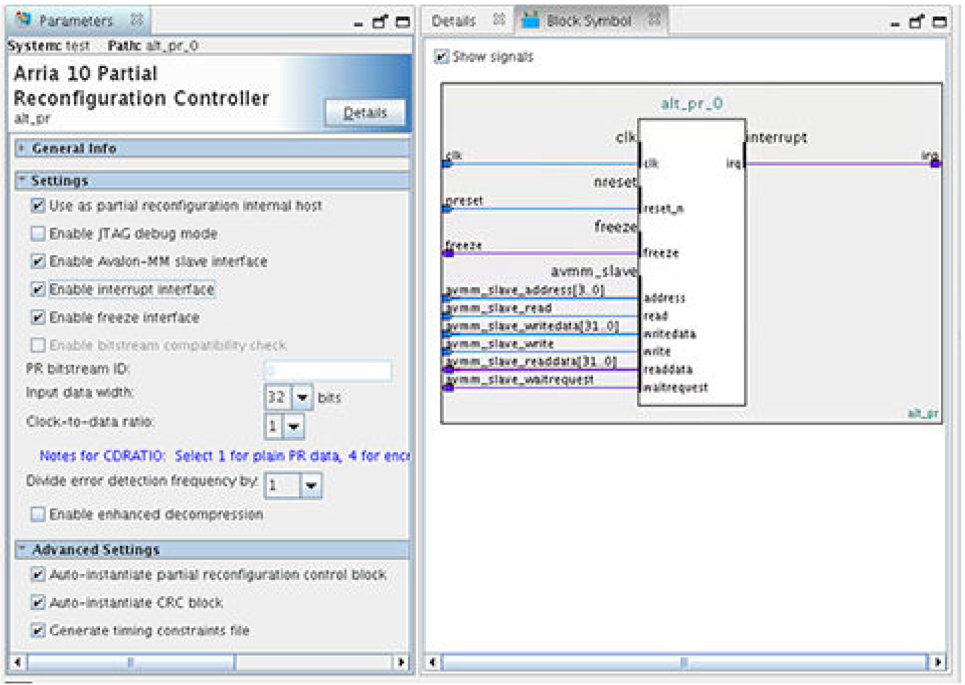
Table 7. Arria 10 Partial Reconfiguration Controller Parameter Settings

Parameter	Value	Description
Use as partial reconfiguration internal host	**On/Off**	Enables the controller for use as an internal host. When enabled, both `prblock` and `crcblock` WYSIWYG are auto-instantiated as part of your design. Disable this option to use the controller as an external host. Connect additional interface signals to the dedicated partial reconfiguration pins.
Enable JTAG debug mode	**On/Off**	Enables access to the controller by the Quartus Prime Programmer for partial reconfiguration over a JTAG interface.
Enable Avalon-MM slave interface	**On/Off**	Enables the controller's Avalon-MM slave interface.
Enable interrupt interface	**On/Off**	Enables interrupt assertion for detection of incompatible bitstream, CRC_ERROR, PR_ERROR, or successful partial reconfiguration. Upon interrupt, query PR_CSR`[4:2]` for status. Write a 1 toPR_CSR`[5]` to clear the interrupt. Use only together with the Avalon-MM slave interface.

continued...

Parameter	Value	Description
Enable freeze interface	**On/Off**	Enables the controller's single-bit freeze interface. This interface identifies whether any region in the design is active or frozen for partial reconfiguration operations. Intel recommends use of the Partial Reconfiguration Region Controller IP rather than the Arria 10 Partial Reconfiguration Controller IP `freeze` signal.
Enable bitstream compatibility check	**On/Off**	Enables bitstream compatibility checks during partial reconfiguration operation of the external host. Bitstream compatibility check is always enabled for partial reconfiguration by internal host. Specify the partial reconfiguration bitstream ID value if you enable this option for partial reconfiguration by external host.
PR bitstream ID	*<32-bit integer>*	Specifies a signed, 32-bit integer value of the partial reconfiguration bitstream ID for the external host. This value must match the partial reconfiguration bitstream ID that the Compiler generates for the target partial reconfiguration design. Locate the partial reconfiguration bitstream ID of the target partial reconfiguration design in the Assembler report (`.asm.rpt`).
Input data width	<bits>	Specifies the size of the controller's data conduit interface in bits.
Clock-to-data ratio	`1|8|16|32`	Specifies the clock-to-data ratio that corresponds with the partial reconfiguration bitstream data type. Refer to the *Valid combinations and CD Ratio for Bitstream Encryption and Compression* table
Divide error detection frequency by	`1..256`	Specifies the divide value of the internal clock. This value determines the frequency of the error detection CRC. The divide value must be a power of two. Refer to device documentation to determine the frequency of the internal clock for the selected device.
Enable enhanced decompression	**On/Off**	Enable enhanced decompression of partial reconfiguration bitstreams.

Figure 9. Arria 10 Partial Reconfiguration Controller Parameter Editor



System: test Path: alt_pr_0

Arria 10 Partial Reconfiguration Controller
alt_pr

- [✓] Use as partial reconfiguration internal host
- [ ] Enable JTAG debug mode
- [✓] Enable Avalon-MM slave interface
- [✓] Enable interrupt interface
- [✓] Enable freeze interface
- [ ] Enable bitstream compatibility check

PR bitstream ID:
Input data width: 32 bits
Clock-to-data ratio: 1

Notes for CDRATIO: Select 1 for plain PR data, 4 for enc

Divide error detection frequency by: 1

- [ ] Enable enhanced decompression

- [✓] Auto-instantiate partial reconfiguration control block
- [✓] Auto-instantiate CRC block
- [✓] Generate timing constraints file

clk, nreset, freeze, avmm_slave_address[3..0], avmm_slave_read, avmm_slave_writedata[31..0], avmm_slave_write, avmm_slave_readdata[31..0], avmm_slave_waitrequest, irq

Table 8. Arria 10 Partial Reconfiguration Controller Advanced Settings

Parameter	Value	Description
Auto-instantiate partial reconfiguration control block	**On/Off**	Automatically includes the partial reconfiguration control block in the controller. When using the controller as an internal host, disable this option to share the partial reconfiguration block with other IP cores. Rather, manually instantiate the partial reconfiguration control block, and connect the relevant signals to the controller.
Auto-instantiate CRC block	**On/Off**	Automatically includes the CRC block within the controller. When using the controller as an internal host, disable this option to share the CRC block with other IP cores.
Generate timing constraints file	**On/Off**	Automatically generates an appropriate Synopsys Design Constraints (`.sdc`) file to constrain the timing of the controller. Disable this option when providing timing constraints in another file.

Related Links

Clock Networks and PLLs in Arria 10 Devices

2.6 Arria 10 Partial Reconfiguration Controller Interface Ports

The Arria 10 Partial Reconfiguration Controller includes the following interface ports.

Table 9. Clock/Reset Ports

These ports are always available.

Port Name	Width	Direction	Function
`nreset`	1	Input	Asynchronous reset for the PR Controller IP core. Resetting the PR Controller IP core during a partial reconfiguration operation can cause the device to lock up.
`clk`	1	Input	User input clock to the PR Controller IP core. This signal is ignored during JTAG debug operations. The input clock must be free-running.

Table 10. Freeze Interface Port

Port Name	Width	Direction	Function
`freeze`	1	Output	Active high signal used to freeze the PR interface signals of any region undergoing partial reconfiguration. De-assertion of this signal indicates the end of PR operation. Intel recommends use of the Partial Reconfiguration Region Controller IP rather than the PR Controller IP `freeze` signal.

Table 11. Conduit Interface Ports

These ports are available when **Enable Avalon-MM slave interface** is **Off**.

Port Name	Width	Direction	Function
`pr_start`	1	Input	A signal arriving at this port asserted high initiates a PR event. You must assert this signal high for a minimum of one clock cycle and de-assert it low prior to the end of the PR operation. This operation allows the PR Controller IP core to be ready to accept the next `pr_start` trigger event when the `freeze` signal is low.

continued...

Port Name	Width	Direction	Function
			The PR Controller IP core ignores this signal during JTAG debug operations.
`data[]`	1, 8, 16, or 32	Input	Selectable input PR data bus width, either x1, x8, x16, or x32. Once a PR event is triggered, it is synchronous with the rising edge of the `clk` signal whenever the `data_valid` signal is high and the `data_ready` signal is high. The PR Controller IP core ignores this signal during JTAG debug operations.
`data_valid`	1	Input	A signal arriving at this port asserted high indicates the `data[]` port contains valid data. The PR Controller IP core ignores this signal during JTAG debug operations.
`data_ready`	1	Output	A signal arriving at this port asserted high indicates the PR Controller IP core is ready to read the valid data on the `data[]` port whenever the `data_valid` signal is asserted high. The data sender must stop sending valid data if this port is low. This signal deasserts low during JTAG debug operations.

`status[2..0]`	1	Output	A 3-bit error output used to indicate the status of PR event. Once an error is detected (PR_ERROR, CRC_ERROR, or incompatible bitstream error), this signal latches high and only resets at the beginning of the next PR event, when `pr_start` is high and `freeze` is low. For example:
			`3’b000 – power-up or nreset asserted`
			`3’b001 – PR_ERROR was triggered`
			`3’b010 – CRC_ERROR was triggered`
			`3’b011 – Incompatible bitstream error detected`
			`3’b100 – PR operation in progress`
			`3’b101 – PR operation passed`
			`3'b110 – Reserved`
			`3'b111 – Reserved`

Table 12. Avalon-MM Slave Interface Ports

These signals are available when **Enable Avalon-MM slave interface** is **On**.

Port Name	Width	Direction	Function
`avmm_slave_address`	4	Input	Avalon-MM address bus. The address bus is in the unit of Word addressing. The PR Controller IP core ignores this signal during JTAG debug operations.
`avmm_slave_read`	1	Input	Avalon-MM read control. The PR Controller IP core ignores this signal during JTAG debug operations.
`avmm_slave_readdata`	32	Output	Avalon-MM read data bus. The PR Controller IP core ignores this signal during JTAG debug operations.
`avmm_slave_write`	1	Input	Avalon-MM write control. The PR Controller IP core ignores this signal during JTAG debug operations.
`avmm_slave_writedata`	32	Input	Avalon-MM write data bus.

continued...

Port Name	Width	Direction	Function
			The PR Controller IP core ignores this signal during JTAG debug operations.
avmm_slave_waitrequest	1	Output	Asserted to indicate that the IP is busy. Also indicates that the IP core is unable to respond to a read or write request. This signal is pulled high during JTAG debug operations.

Table 13. Interrupt Interface Ports

These ports are available when **Enable interrupt interface** is **On**.

Port Name	Width	Direction	Function
irq	**1**	Output	The interrupt signal.

Table 14. CRC BLOCK Interface

These ports are available when **Use as Partial Reconfiguration Internal Host** is **Off**, or when you instantiate the CRCBLOCK manually for an internal host.

Port Name	Width	Direction	Function
crc_error_pin	**1**	Input	Available when you use the PR Controller IP core as an External Host. Connect this port to the dedicated CRC_ERROR pin of the FPGA undergoing partial reconfiguration.

Table 15. PR Block Interface

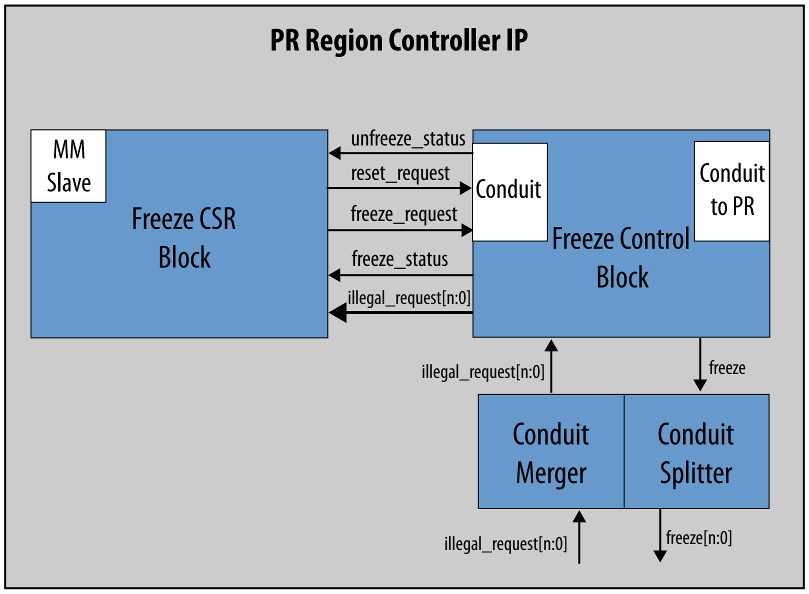
These options are available when **Use as Partial Reconfiguration Internal Host** is **Off**, or when you instantiate the PRBLOCK manually for an internal host.

Port Name	Width	Direction	Function
pr_ready_pin	**1**	Input	Connect this port to the dedicated PR_READY pin of the FPGA undergoing partial reconfiguration.
pr_error_pin	**1**	Input	Connect this port to the dedicated PR_ERROR pin of the FPGA undergoing partial reconfiguration.
pr_done_pin	**1**	Input	Connect this port to the dedicated PR_DONE pin of the FPGA undergoing partial reconfiguration.
pr_request_pin	**1**	Output	Connect this port to the dedicated PR_REQUEST pin of the FPGA undergoing partial reconfiguration.
pr_clk_pin	**1**	Output	Connect this port to the dedicated DCLK of the FPGA undergoing partial reconfiguration.
pr_data_pin[31..0]	**32**	Output	Connect this port to the dedicated DATA[31..0] pins of the FPGA undergoing partial reconfiguration.

3 Partial Reconfiguration Region Controller

The Partial Reconfiguration Region Controller provides a standard interface to the to the Freeze Control block that controls handshaking with the PR region. The PR handshake ensures that PR region transactions complete before freeze of the interface.

Figure 10. Partial Reconfiguration Region Controller IP Core



The Partial Reconfiguration Region Controller IP core performs the following operations in a partial reconfiguration:

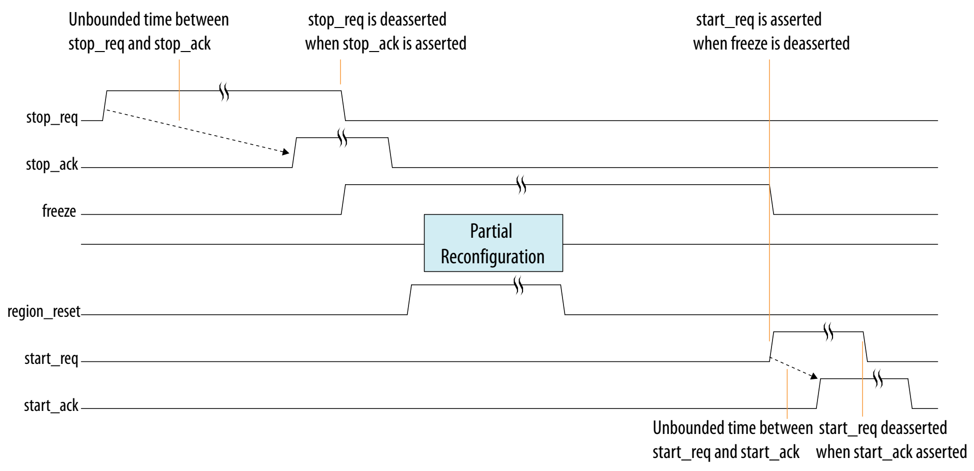
Table 16. Partial Reconfiguration Region Controller Operation

Operation	Description
1. Freeze Control Block performs PR handshaking with the PR region	Writing `1` to the `freeze_csr_ctrl` register `freeze_req` bit causes the controller to issue `stop_req` to request the PR region terminate or complete current transactions.
2. Freeze CSR Block generates the `freeze` output signal	Upon `stop_ack` response, the controller asserts high on the freeze signal connected to all the PR region's interface bridges. The PR region interfaces enter the freeze state when `freeze` signal is high. The freeze signal deasserts when the `freeze_csr_ctrl` register `freeze_request` bit is low. After the freeze signal is deasserted, the region controller issues `start_req` to request the PR region to start executing the PR persona. The PR region then responds with `start_ack` to indicate successful start and readiness for system use.

continued...

Operation	Description
3. Freeze Control Block resets the PR region	The Freeze Control Block `region_reset` output resets the PR region. Reset the PR region before the new persona starts executing. This reset ensures that the persona loads into a defined state. PR does not support initial conditions. Assertion of reset can happen at any time. Deassertion of reset can only happen after the PR configuration process completes.
4. The Conduit Splitter connects the controller's `freeze` signal to one or more Freeze Bridge components	The Conduit Splitter receives the `freeze` signal from the freeze control block and assigns the `freeze` input signal to one or more `freeze` output signals.
5. The Conduit Merger connects the `illegal_request` signal from one or more Freeze Bridge components to the region controller	The `illegal_request` is a single-bit output signal from the Freeze Bridge. Conduit Merger concatenates the single-bit signal from multiple Freeze Bridges into a multi-bit bus. The Conduit Merger then connects the bus to the Freeze Control Block.

Figure 11. Freeze Control Block PR Handshake Timing


Unbounded time between stop_req and stop_ack
stop_req is deasserted when stop_ack is asserted
start_req is asserted when freeze is deasserted
stop_req
stop_ack
freeze
Partial Reconfiguration
region_reset
start_req
start_ack
Unbounded time between start_req and start_ack
start_req deasserted when start_ack asserted

3.1 Partial Reconfiguration Region Controller Parameters

The Partial Reconfiguration Region Controller IP core supports customization of the following parameters.

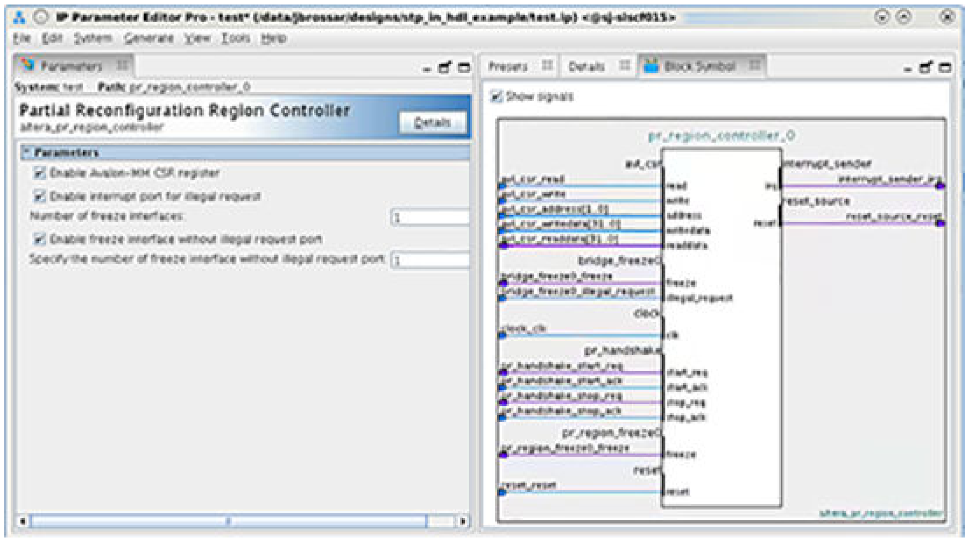
Table 17. Partial Reconfiguration Region Controller Parameter Settings

Parameter	Value	Default	Description
Enable Avalon-MM CSR register	On/Off	On	Enables Avalon-MM CSR registers in the PR region controller. Disable this option to expose a conduit interface and not instantiate the CSR block.
Enable interrupt port for illegal request	On/Off	On	Enables the interrupt port for illegal operations in the PR region controller.

continued...

Parameter	Value	Default	Description
Number of freeze interfaces	*number*		Specifies the number of freeze interfaces for freeze operations. Each freeze interface connects to a freeze bridge.
Enable freeze interface without illegal request port	On/Off	Off	Removes the illegal request port from the freeze interface.
Specify the number of freeze interface without illegal request port	*number*		Specifies the number of freeze interfaces without the `illegal_request` port.

Figure 12. Partial Reconfiguration Region Controller Parameter Editor


IP Parameter Editor Pro - test* (/data/jbrossar/designs/stp_in_hdl_example/test.ip) <@sj-slsc f015>

System: test Path: pr_region_controller_0

Partial Reconfiguration Region Controller
altera_pr_region_controller

Parameters
- [✓] Enable Avalon-MM CSR register
- [✓] Enable interrupt port for illegal request
- Number of freeze interfaces: 1
- [✓] Enable freeze interface without illegal request port
- Specify the number of freeze interface without illegal request port: 1

3.2 Partial Reconfiguration Region Controller Interface Ports

Table 18. Freeze CSR Block Ports

These ports are available when **Enable Avalon-MM CSR Register** is **On**.

Port	Width	Direction	Description
Clock			
`clock_clk`	1	Input	IP core input clock.
Reset			
`reset_reset`	1	Input	Synchronous reset.
Avalon-MM Slave			
`avl_csr_addr`	2	Input	Avalon-MM address bus. The address bus is in word addressing.
`avl_csr_read`	1	Input	Avalon-MM read control to CSR block.
`avl_csr_write`	1	Input	Avalon-MM write control to CSR.
`avl_csr_writedata`	32	Input	Avalon-MM write data bus to CSR.

continued...

Port	Width	Direction	Description
`avl_csr_readdata`	32	Output	Avalon-MM read data bus from CSR.
Interrupt			
`interrupt_sender_irq`	1	output	Trigger by illegal read or illegal write.

Table 19. Freeze Control Block Ports

Port	Width	Direction	Description
Conduit Handshaking with PR Region			
`pr_handshake_stop_req`	1	Output	An assertion on this output requests that the PR persona stop executing.
`pr_handshake_stop_ack`	1	Input	A value of 1 on this input acknowledges that the executing PR persona stops executing and a new persona can replace it.
`pr_handshake_start_req`	1	Output	An assertion on this output requests that the new PR persona starts executing.
`pr_handshake_start_ack`	1	Input	A value of 1 on this input acknowledges that the new PR persona starts executing and can stop executing on a `pr_handshake_stop_req`.
Conduit Interface with CSR Block Disabled			
`conduit_control_freeze_req`	1	Input	Write 1 on this bit to start freezing the PR region interfaces.
`conduit_control_unfreeze_req`	1	Input	Write 1 on this bit to stop freezing the PR region interfaces.
`conduit_control_freeze_status`	1	Output	High on this bit indicates that the PR region is successfully goes into freezing state.
`conduit_control_reset`	1	Input	Write 1 on this bit to reset the PR region.
`conduit_control_unfreeze_status`	1	Output	High on this bit indicates that the PR region successfully leaves freezing state.
`conduit_control_illegal_req`	n	Output	High on this bit indicates illegal reads/writes on the Avalon-MM Slave Bridge.

Table 20. Conduit Splitter and Merger Interface Ports

Signal	Width	Direction	Description
`bridge_freeze0_freeze`	1	Output	This output connects to freeze input signal of freeze bridge block. (Multiple interfaces generate according to the number of freeze interfaces)
`bridge_freeze0_illegal_request`	1	Input	This input connects to `illegal_request` output signal from freeze bridge block

3.3 Partial Reconfiguration Region Controller Registers

The following describes the operation of the Partial Reconfiguration Region Controller registers.

Figure 13. Partial Reconfiguration Region Controller Register States and Programming Model

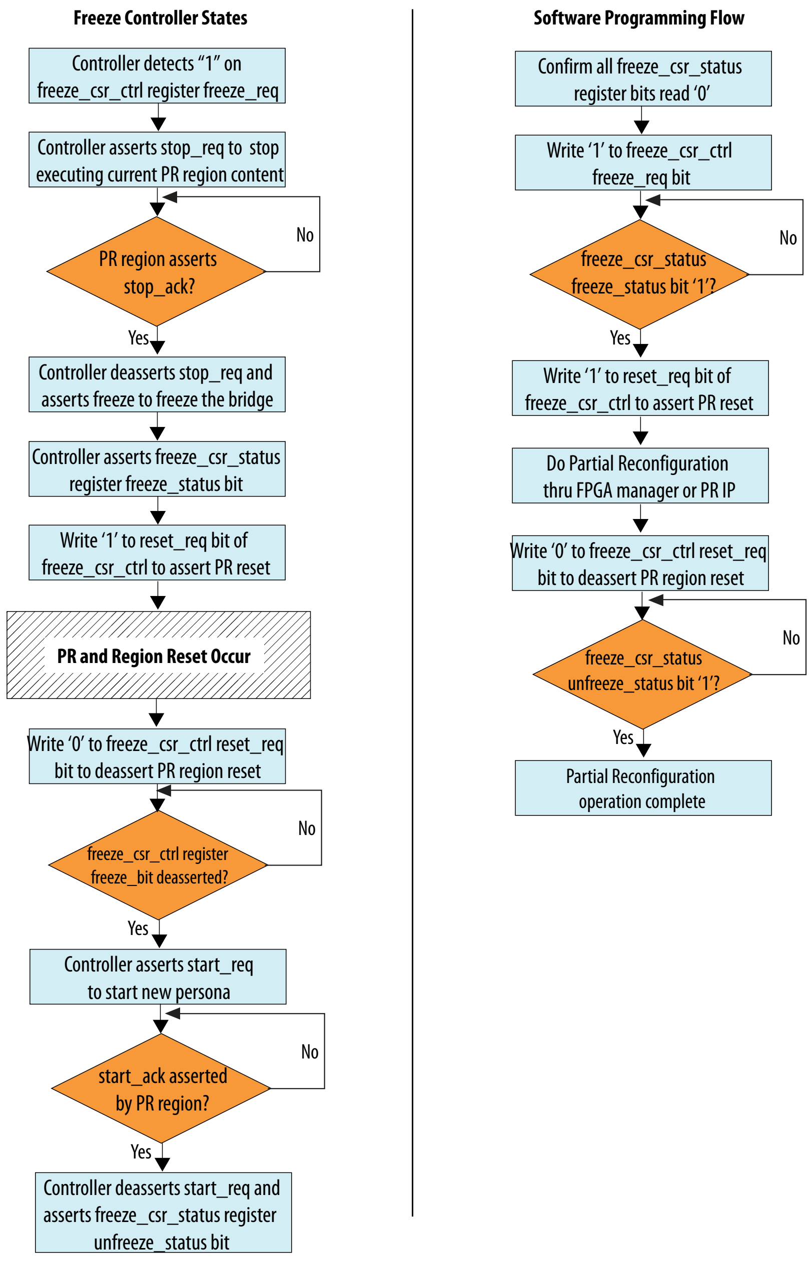

Freeze Controller States
Controller detects "1" on freeze_csr_ctrl register freeze_req
Controller asserts stop_req to stop executing current PR region content
PR region asserts stop_ack?
No
Yes
Controller deasserts stop_req and asserts freeze to freeze the bridge
Controller asserts freeze_csr_status register freeze_status bit
Write '1' to reset_req bit of freeze_csr_ctrl to assert PR reset
PR and Region Reset Occur
Write '0' to freeze_csr_ctrl reset_req bit to deassert PR region reset
freeze_csr_ctrl register freeze_bit deasserted?
No
Yes
Controller asserts start_req to start new persona
start_ack asserted by PR region?
No
Yes
Controller deasserts start_req and asserts freeze_csr_status register unfreeze_status bit
Software Programming Flow
Confirm all freeze_csr_status register bits read '0'
Write '1' to freeze_csr_ctrl freeze_req bit
freeze_csr_status freeze_status bit '1'?
No
Yes
Write '1' to reset_req bit of freeze_csr_ctrl to assert PR reset
Do Partial Reconfiguration thru FPGA manager or PR IP
Write '0' to freeze_csr_ctrl reset_req bit to deassert PR region reset
freeze_csr_status unfreeze_status bit '1'?
No
Yes
Partial Reconfiguration operation complete

Table 21. PR Region Controller Register Map

Name	Address Offset	Access	Description
`freeze_csr_status`	0x0	Read-Only	Freeze status register.
`csr_ctrl`	0x1	Read or Write	Control register to enable and disable freeze.
`freeze_illegal_req`	0x02	Read or Write	High on any bit indicates an illegal request during the freeze state.
`freeze_reg_version`	0x03	Read-Only	Version register

Table 22. freeze_csr_status

Bit	Fields	Access	Default Value	Description
31:2	Reserved	N/A	0x0	Reserved. Reading these bits always returns zeros.
1	unfreeze_status	R	0	Hardware sets this bit to '1 after the PR region return `start_ack` to indicate that it has successfully kicked off the persona. Hardware clears this bit to '0 once `unfreeze_req` bit is low. This bit is '1 when bridge is released from reset.
0	freeze_status	R	0	Hardware sets this bit to '1 after the freeze bridge return `freeze_ack` signal to indicate that it has successfully goes into frozen state. Hardware clears this bit to '0 once `freeze_req` bit is low. This bit is '0 when bridge is released from reset.

Table 23. freeze_csr_ctrl

Bit	Fields	Access	Default Value	Description
31:3	Reserved	N/A	0x0	Reserved. Reading these bits always returns zeros.
2	unfreeze_req	R/W	0	Write 1 to this bit to stop freezing the PR region interfaces. Hardware clears this bit after `unfreeze_status` is high. Write 0 to this bit to terminate the unfreeze request. If `start_ack` never return by the PR region after this bit was asserted for some time. Do not assert this bit and`freeze_req` bit at the same time. If both `freeze_req` and `unfreeze_req` assert at the same time, it is an invalid operation.
1	reset_req	R/W	0	Write 1 to start resetting the PR persona. Write 0 to stop resetting the PR persona.
0	freeze_req	R/W	0	Write 1 to this bit to start freezing the PR region interfaces. Hardware clears this bit after `freeze_status` is high.

continued...

Bit	Fields	Access	Default Value	Description
				Write '0 to this bit to terminate the freeze request if `stop_ack` never return by the PR region freeze_bridge after this bit asserts for some time. Do not assert this bit and `unfreeze_req` bit at the same time. If both `freeze_req` and `unfreeze_req` are asserted at the same time, it is an invalid operation.

Table 24. freeze_illegal_request

Bit	Fields	Access	Default Value	Description
31:n	Reserved	N/A	0x0	Reserved. Reading these bits will always return zeros.
n:0	illegal_request	R/W	0	High on any bit of this bus indicates a read or write issues by the static region master when the Avalon-MM slave freeze bridge is in freeze state. Identify which freeze bridge has an illegal request by checking each bit on the bus. For example, when `illegal_request` bit 2 is high, illegal request happens in freeze bridge that connects to interface `freeze_conduit_in2` This bus triggers interrupt signal. Write '1' to clear this bit. n – Number of interface bridges

Table 25. freeze_reg_version

Bit	Fields	Access	Default Value	Description
31:0	Version Register	Read-Only	AD000003	This register bit indicates the CSR register version number. Currently the CSR register is version AD000003.

4 Avalon-MM Partial Reconfiguration Freeze Bridge

The Avalon-MM Freeze Bridge component freezes the PR region interface when the `freeze` input signal is high. Each interface to a PR region requires an instance of the Freeze Bridge or your own freeze logic.

Figure 14. Avalon-MM Partial Reconfiguration Freeze Bridge

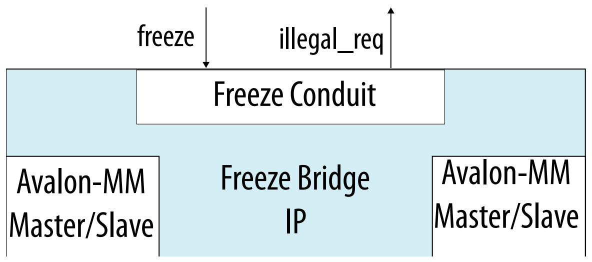


Table 26. Read and Write Request to PR Region Avalon-MM Slave Interface

The Freeze Bridge handles read and write transactions differently for each of the following possible interface configurations. The Freeze Bridge is in the freeze state as long as the `freeze` signal from the PR region or PR region controller is asserted.

Interface Connection	Behavior
Read request to Avalon-MM slave interface in PR region	1. During the `freeze` state any received read transaction responds with bogus data <'hDEADBEEF>. The corresponding `freeze_illegal_request` register bit is set. 2. During freeze state, `readrequest`, `writerequest`, `waitrequest`, `beginbursttransfer`, `lock`, and `debugaccess` signal in the PR region interface are tied low. 3. The Avalon-MM slave response signal constantly returns 2'b10 to indicate an unsuccessful transaction from an endpoint slave.
Write request to slave interface in PR region	1. During the `freeze` state, any received write transactions are ignored by the Freeze Bridge. The Freeze Bridge pulls the `waitrequest`, `beginbursttransfer`, `lock` and `debugaccess` signals low. The corresponding `freeze_illegal_request` register bit is set. 2. The Avalon-MM slave response signal updates with 2'b10 to indicate an unsuccessful transaction from an endpoint slave.

Table 27. Read and Write Request from PR Region Avalon-MM Master Interface

Interface Connection	Behavior
Read/Write request from Avalon-MM master interface in PR region	1. During the `freeze` state the read and write signals from the PR region are ignored. 2. The read and write signals to the static region are deasserted.

Table 28. Avalon-MM Partial Reconfiguration Freeze Bridge Signal Behavior

The table below summarizes the Avalon interface output signal behavior when the Freeze Bridge is in a frozen state. When not frozen, all signals are just pass-through.

Signal	Avalon-MM Slave Bridge	Avalon-MM Master Bridge
`write`	‘b0 (tie low)	‘b0 (tie low)
`read`	‘b0 (tie low)	‘b0 (tie low)
`address`	Pass through	Pass through
`writedata`	Pass through	Pass through
`readdata`	Return <h’DEADBEEF> always	Pass through
`byteenable`	Pass through	Pass through
`burstcount`	Pass through	Pass through
`beginbursttransfer`	‘b0 (tie low)	‘b0 (tie low)
`debugaccess`	‘b0 (tie low)	‘b0 (tie low)
`readdatavalid`	Return ‘b1 when there is a request, else ‘b0	Pass through
`waitrequest`	Return ‘b1 when there is a request, else ‘b0	‘b0 (tie low)
`response`	Return ‘b10 always	Pass through
`lock`	‘b0 (tie low)	‘b0 (tie low)
`writeresponsevalid`	Return ‘b1 when there is a request, else ‘b0	Pass through

4.1 Avalon-MM Partial Reconfiguration Freeze Bridge Parameters

The Avalon-MM Partial Reconfiguration Freeze Bridge IP core supports customization of the following parameters.

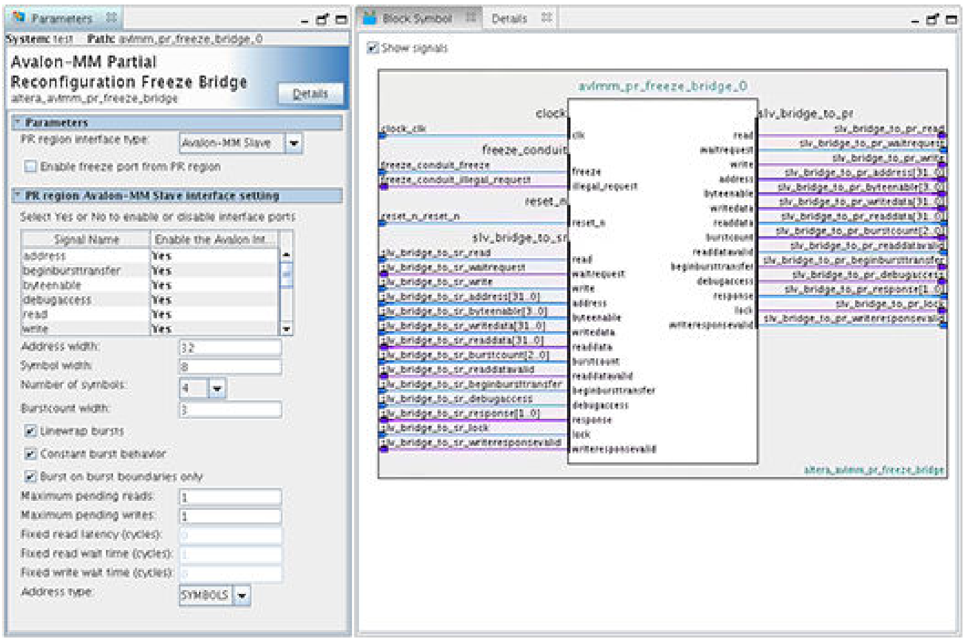
Table 29. Avalon-MM Partial Reconfiguration Freeze Bridge Parameters

Parameter	Values	Description
	Parameters	
PR region interface Type	**Avalon-MM Slave/Avalon-MM Master**	Specifies the interface type for interfacing with the Freeze Bridge.
Enable Freeze port from PR region	**On/Off**	Enables the `freeze` port that freezes all the outputs of each PR region to a known constant value. Freezing prevents the signal receivers in the static region from receiving undefined signals during the partial reconfiguration process. The freeze of a bridge is the logical OR of this signal from the PR region, and the freeze from the PR region controller.
	PR region Avalon-MM Slave Interface Settings	
Enabled Avalon Interface Signal	**Yes/No**	Enable (**Yes**) or disable (**No**) specific optional Freeze Bridge interface ports.
Address width	*<1-64>*	Address width in bits.
Symbol width	*<number>*	Data symbol width in bits. The symbol width should be 8 for byte-oriented interfaces.

continued...

Parameter	Values	Description
Number of symbols	*<number>*	Number of symbols per word.
Burstcount width	*<number>*	The width of the burst count in bits.
Linewrap burst	**On**/Off	When **On**, the address for bursts wraps instead of incrementing. With a wrapping burst, when the address reaches a burst boundary, it wraps back to the previous burst boundary. Consequently, only the low order bits are used for addressing.
Constant burst behavior	**On**/Off	When **On**, memory bursts are constant.
Burst on burst boundaries only	**On**/Off	When **On**, memory bursts are aligned to the address size.
Maximum pending reads	*<number>*	The maximum number of pending reads that the slave can queue.
Maximum pending writes	*<number>*	The maximum number of pending writes that the slave can queue.
Fixed read latency (cycles)	*<number>*	Sets the read latency for fixed-latency slaves. Not useful on interfaces that include the `readdatavalid` signal.
Fixed read wait time (cycles)	*<number>*	For master interfaces that do not use the `waitrequest` signal. The read wait time indicates the number of cycles before the master responds to a read. The timing is as if the master asserted `waitrequest` for this number of cycles.
Fixed write wait time (cycles)	*<number>*	For master interfaces that do not use the `waitrequest` signal. The write wait time indicates the number of cycles before the master accepts a write.
Address type	**WORDS/SYMBOLS**	Sets slave interface address type to symbols or words.

Figure 15. Avalon-MM Partial Reconfiguration Freeze Bridge Parameter Editor


Parameters
Block Symbol
Details
Show signals
avmm_pr_freeze_bridge_0
clock
clock_clk
clk
freeze_conduit
freeze_conduit_freeze
freeze_conduit_illegal_request
freeze
illegal_request
reset_n
reset_n_reset_n
reset_n
slv_bridge_to_sr
slv_bridge_to_sr_read
slv_bridge_to_sr_waitrequest
slv_bridge_to_sr_write
slv_bridge_to_sr_address[31..0]
slv_bridge_to_sr_byteenable[3..0]
slv_bridge_to_sr_writedata[31..0]
slv_bridge_to_sr_readdata[31..0]
slv_bridge_to_sr_burstcount[2..0]
slv_bridge_to_sr_readdatavalid
slv_bridge_to_sr_beginbursttransfer
slv_bridge_to_sr_debugaccess
slv_bridge_to_sr_response[1..0]
slv_bridge_to_sr_lock
slv_bridge_to_sr_writeresponsevalid
read
waitrequest
write
address
byteenable
writedata
readdata
burstcount
readdatavalid
beginbursttransfer
debugaccess
response
lock
writeresponsevalid
slv_bridge_to_pr
slv_bridge_to_pr_read
slv_bridge_to_pr_waitrequest
slv_bridge_to_pr_write
slv_bridge_to_pr_address[31..0]
slv_bridge_to_pr_byteenable[3..0]
slv_bridge_to_pr_writedata[31..0]
slv_bridge_to_pr_readdata[31..0]
slv_bridge_to_pr_burstcount[2..0]
slv_bridge_to_pr_readdatavalid
slv_bridge_to_pr_beginbursttransfer
slv_bridge_to_pr_debugaccess
slv_bridge_to_pr_response[1..0]
slv_bridge_to_pr_lock
slv_bridge_to_pr_writeresponsevalid
altera_avlmm_pr_freeze_bridge

System: test Path: avlmm_pr_freeze_bridge_0

Avalon-MM Partial Reconfiguration Freeze Bridge
altera_avlmm_pr_freeze_bridge

Parameters

PR region interface type: Avalon-MM Slave

☐ Enable freeze port from PR region

PR region Avalon-MM Slave interface setting

Select Yes or No to enable or disable interface ports

Signal Name	Enable the Avalon Int...
address	Yes
beginbursttransfer	Yes
byteenable	Yes
debugaccess	Yes
read	Yes
write	Yes

Address width: 32
Symbol width: 8
Number of symbols: 4
Burstcount width: 3
☑ Linewrap bursts
☑ Constant burst behavior
☑ Burst on burst boundaries only
Maximum pending reads: 1
Maximum pending writes: 1
Fixed read latency (cycles): 0
Fixed read wait time (cycles): 1
Fixed write wait time (cycles): 0
Address type: SYMBOLS

4.2 Avalon-MM Partial Reconfiguration Freeze Bridge Interface Ports

Table 30. Avalon-MM Partial Reconfiguration Freeze Bridge Interface Ports

Port	Width	Direction	Description
`clock_clk`	1	Input	Input clock for the IP.
`reset_n_reset_n`	1	Input	Synchronous reset for the IP.
`freeze_conduit_freeze`	1	Input	When this signal is high, the bridge handles any current transaction properly then freezes the PR interfaces.
`freeze_conduit_illegal_request`	1	Output	High on this bus indicates that an illegal request is issued to the bridge during freezing state.
`pr_freeze_pr_freeze`	1	Input	Enabled `freeze` port from the PR region.

Table 31. Avalon-MM Slave to PR Region Master Interface Ports

Port	Width	Direction	Description
`slv_bridge_to_pr_read`	1	Output	Optional Avalon-MM slave bridge to PR region `read` port.
`slv_bridge_to_pr_waitrequest`	1	Input	Optional Avalon-MM slave bridge to PR region `waitrequest` port.
`slv_bridge_to_pr_write`	1	Output	Optional Avalon-MM slave bridge to PR region `write` port.
`slv_bridge_to_pr_address`	32	Output	Optional Avalon-MM slave bridge to PR region `address` port.
`slv_bridge_to_pr_byteenable`	4	Output	Optional Avalon-MM slave bridge to PR region `byteenable` port.
`slv_bridge_to_pr_writedata`	32	Output	Optional Avalon-MM slave bridge to PR region `writedata` port.
`slv_bridge_to_pr_readdata`	32	Input	Optional Avalon-MM slave bridge to PR region `readdata` port.
`slv_bridge_to_pr_burstcount`	3	Output	Optional Avalon-MM slave bridge to PR region `burstcount` port.
`slv_bridge_to_pr_readdatavalid`	1	Input	Optional Avalon-MM slave bridge to PR region `readdatavalid` port.
`slv_bridge_to_pr_beginbursttransfer`	1	Output	Optional Avalon-MM slave bridge to PR region `readdatavalid` port.
`slv_bridge_to_pr_debugaccess`	1	Output	Optional Avalon-MM slave bridge to PR region `debugaccess` port.
`slv_bridge_to_pr_response`	2	Input	Optional Avalon-MM slave bridge to PR region `response` port.
`slv_bridge_to_pr_lock`	1	Output	Optional Avalon-MM slave bridge to PR region `lock` port.
`slv_bridge_to_pr_writeresponsevalid`	1	Input	Optional Avalon-MM slave bridge to PR region `writeresponsevalid` port.

Table 32. Avalon-MM Slave to Static Region Master Interface Ports

Note: Same setting as Avalon-MM master to PR region slave interface.

Port	Width	Direction	Description
`slv_bridge_to_sr_read`	1	Input	Avalon-MM slave bridge to static region `read` port.
`slv_bridge_to_sr_waitrequest`	1	Output	Avalon-MM slave bridge to static region `waitrequest` port.
`slv_bridge_to_sr_write`	1	Input	Avalon-MM slave bridge to static region `write` port.
`slv_bridge_to_sr_address`	32	Input	Avalon-MM slave bridge to static region `address` port.
`slv_bridge_to_sr_byteenable`	n	Input	Avalon-MM slave bridge to static region `byteenable` port.
`slv_bridge_to_sr_writedata`	32	Input	Avalon-MM slave bridge to static region `writedata` port.
`slv_bridge_to_sr_readdata`	32	Output	Avalon-MM slave bridge to static region `readdata` port.
`slv_bridge_to_sr_beginbursttransfer`	1	Input	Avalon-MM slave bridge to static region `beginbursttransfer` port.
`slv_bridge_to_sr_ debugaccess`	1	Input	Avalon-MM slave bridge to static region `debugaccess` port.
`slv_bridge_to_sr_response`	2	Output	Avalon-MM slave bridge to static region `response` port.
`slv_bridge_to_sr_lock`	1	Input	Avalon-MM slave bridge to static region `lock` port.
`slv_bridge_to_sr_writeresponsevalid`	1	Output	Avalon-MM slave bridge to static region `writereponsevalid` port.

Table 33. Avalon-MM Master to PR Region Slave Interface Ports

Port	Width	Direction	Description
`mst_bridge_to_pr_read`	1	Input	Optional Avalon-MM master bridge to PR region `read` port.
`mst_bridge_to_pr_waitrequest`	1	Output	Optional Avalon-MM master bridge to PR region `waitrequest` port.
`mst_bridge_to_pr_write`	1	Input	Optional Avalon-MM master bridge to PR region `write` port.
`mst_bridge_to_pr_address`	32	Input	Optional Avalon-MM master bridge to PR region `address` port.
`mst_bridge_to_pr_byteenable`	4	Input	Optional Avalon-MM master bridge to PR region `byteenable` port.
`mst_bridge_to_pr_writedata`	32	Input	Optional Avalon-MM master bridge to PR region `writedata` port.
`mst_bridge_to_pr_readdata`	32	Output	Optional Avalon-MM master bridge to PR region `readdata` port.

continued...

Port	Width	Direction	Description
mst_bridge_to_pr_burstcount	3	Input	Optional Avalon-MM master bridge to PR region burstcount port.
mst_bridge_to_pr_readdatavalid	1	Output	Optional Avalon-MM master bridge to PR region readdatavalid port.
mst_bridge_to_pr_beginbursttransfer	1	Input	Optional Avalon-MM master bridge to PR region beginbursttransfer port.
mst_bridge_to_pr_debugaccess	1	Input	Optional Avalon-MM master bridge to PR region debugaccess port.
mst_bridge_to_pr_response	2	Output	Optional Avalon-MM master bridge to PR region response port.
mst_bridge_to_pr_lock	1	Input	Optional Avalon-MM master bridge to PR region lock port.
mst_bridge_to_pr_writeresponsevalid	1	Output	Optional Avalon-MM master bridge to PR region writeresponsevalid port.

Table 34. Avalon-MM Master to Static Region Slave Interface Ports

Same setting as Avalon-MM slave to PR region master interface.

Port	Width	Direction	Description
mst_bridge_to_sr_read	1	Output	Avalon-MM master bridge to static region read port.
mst_bridge_to_sr_waitrequest	1	Input	Avalon-MM master bridge to static region read port.
mst_bridge_to_sr_write	1	Output	Avalon-MM master bridge to static region read port.
mst_bridge_to_sr_address	32	Output	Avalon-MM master bridge to static region address port.
mst_bridge_to_sr_byteenable	4	Output	Avalon-MM master bridge to static region byteenable port.
mst_bridge_to_sr_writedata	32	Output	Avalon-MM master bridge to static region writedata port.
mst_bridge_to_sr_readdata	32	Input	Avalon-MM master bridge to static region readdata port.
mst_bridge_to_sr_burstcount	3	Output	Avalon-MM master bridge to static region burstcount port.
mst_bridge_to_sr_readdatavalid	1	Input	Avalon-MM master bridge to static region readdatavalid port.
mst_bridge_to_sr_beginbursttransfer	1	Output	Avalon-MM master bridge to static region beginbursttransfer port.
mst_bridge_to_sr_debugaccess	1	Output	Avalon-MM master bridge to static region debugaccess port.
mst_bridge_to_sr_response	2	Input	Avalon-MM master bridge to static region response port.
mst_bridge_to_sr_lock	1	Output	Avalon-MM master bridge to static region lock port.
mst_bridge_to_sr_writeresponsevalid	1	Input	Avalon-MM master bridge to static region writeresponsevalid port.

5 Avalon-ST Partial Reconfiguration Freeze Bridge

The Avalon-ST Freeze Bridge component freezes the PR region interface when the `freeze` input signal is high. The Freeze Bridge ensures that any transaction is complete before freezing the connected interface. Each interface to a PR region requires an instance of the Freeze Bridge or your own freeze logic.

Figure 16. Avalon-ST Partial Reconfiguration Freeze Bridge

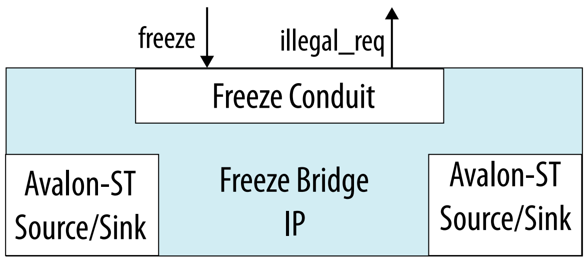


Table 35. Avalon-ST Source Freeze Bridge Interface Behavior

Interface Type	Behavior
Source bridge with packet transfer	1. When the `freeze` signal goes high, the Freeze Bridge handles the `startofpacket`, `endofpacket`, and `empty` bits and does not send transactions to the static region. 2. When the Freeze Bridge detects a `startofpacket` transaction without a corresponding `endofpacket` during the frozen state, this indicates an unfinished transaction. 3. The source bridge then completes the transaction by asserting `valid` and `endofpacket` high for one clock cycle. 4. The `channel` signal is remains constant, while data bits are set to `'hDEADBEEF` and error bit is set to `1'b1`. 5. The `illegal_req` output signal triggers update of the CSR register in the Partial Reconfiguration Region Controller.
Source bridge without packet transfer	When the `freeze` signal is high, the Freeze Bridge does not send transactions to the static region. The Freeze Bridge remains idle until it leaves the frozen state.
Source bridge with max_channel > 1	When multiple channels transfer unfinished transactions, the Freeze Bridge tracks the `channel` values to ensure that all packet transactions from different channels end by asserting the `endofpacket` bit during the frozen state.
Source bridge with ready_latency > 0	When the Freeze Bridge drives `endofpacket`, `valid`, or `channel` outputs to the static region, the Freeze Bridge reads the `ready_latency` value. The `ready_latency` value defines the actual clock cycle sink component ready for data.

Figure 17. Source Bridge Handling of Unfinished Packet Transaction During Freeze

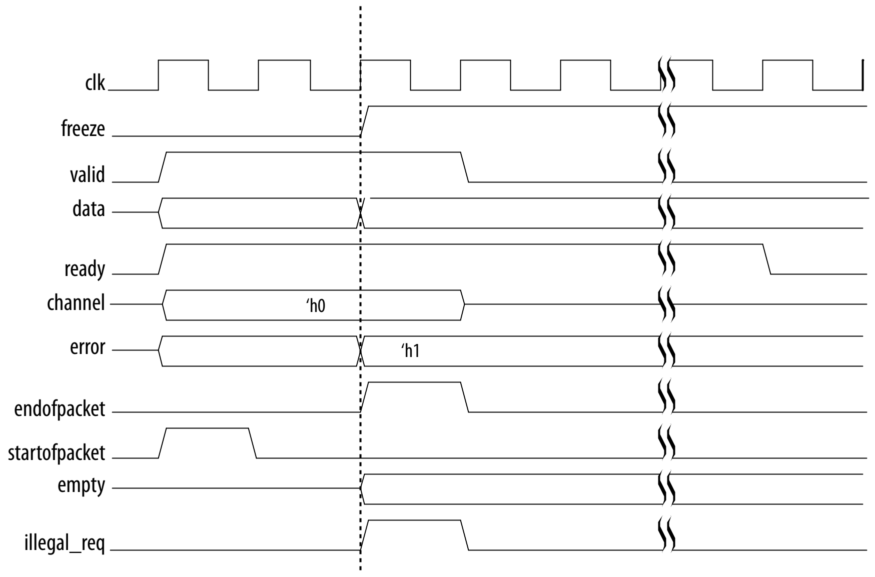


Figure 18. PR Freeze Bridge Asserting valid Signal to End Packet Transactions

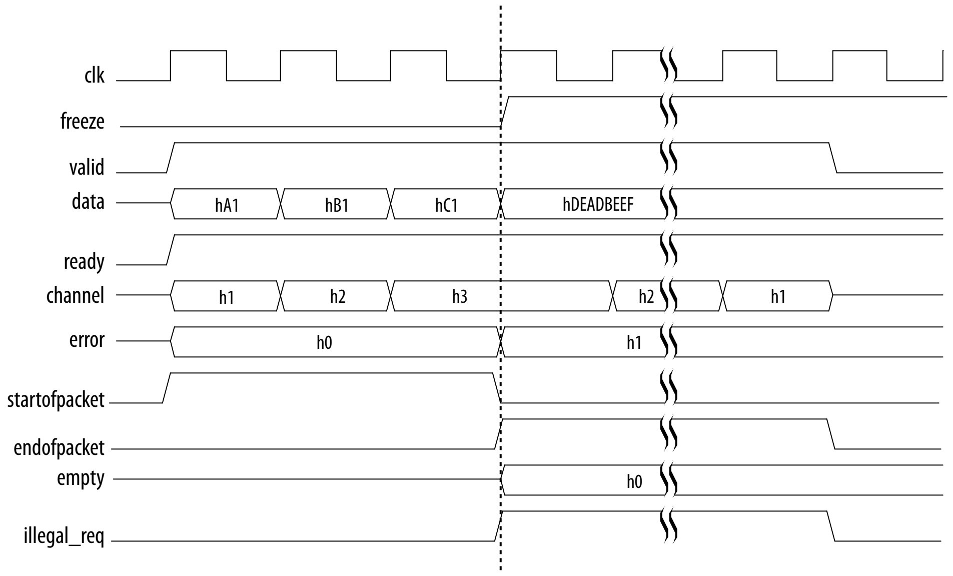


Table 36. Avalon-ST Sink Freeze Bridge Interface Behavior

Interface Type	Behavior
Sink Bridge	For transactions that includes packet transfers, when the `freeze` signal goes high, the Freeze Bridge holds the `ready` signal high until any unfinished transaction completes. For transactions that do not include packet transfers, when the `freeze` signal goes high, the Freeze Bridge holds the `ready` signal low during the freeze period.

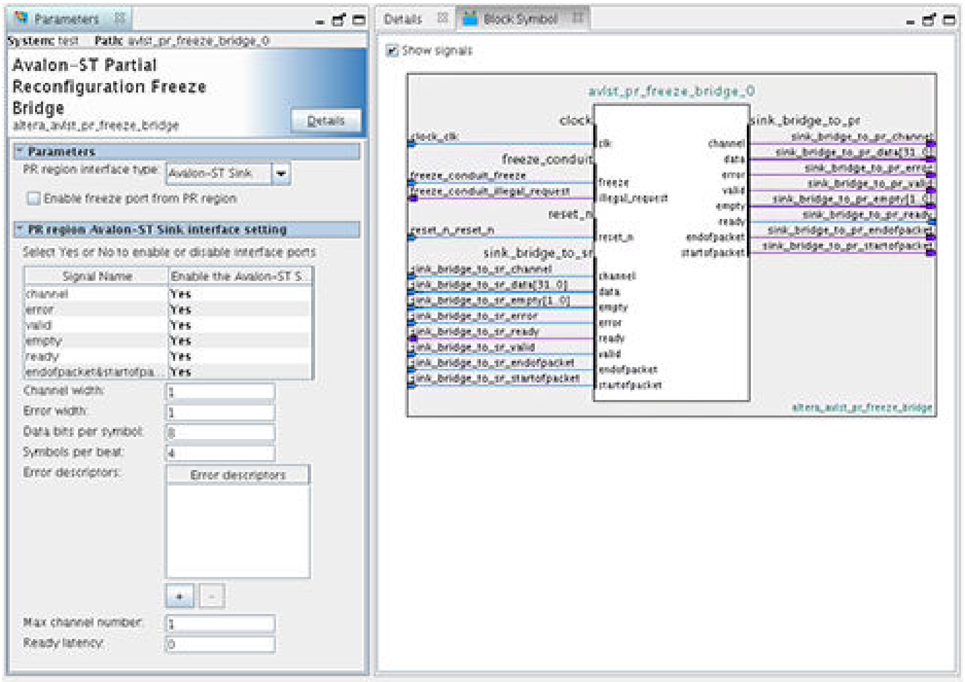
continued...

Interface Type	Behavior
	The `illegal_req` signal asserts high to indicate that the current transaction is an error. Configure the design to stop sending transactions to the PR region after the `illegal_req` signal is high.
Sink bridge with ready_latency > 0	When the Freeze Bridge drives `endofpacket`, `valid`, or `channel` outputs to the static region, the Freeze Bridge must observe the `ready_latency` value. The `ready_latency` value defines the actual clock cycle sink component ready for data.

5.1 Avalon-ST Partial Reconfiguration Freeze Bridge Parameters

The Avalon-ST Partial Reconfiguration Freeze Bridge IP core supports customization of the following parameters.

Figure 19. Avalon-ST Partial Reconfiguration Freeze Bridge Parameter Editor



System: test Path: avlst_pr_freeze_bridge_0

Avalon-ST Partial Reconfiguration Freeze Bridge
altera_avlst_pr_freeze_bridge

Parameters

PR region interface type: Avalon-ST Sink

☐ Enable freeze port from PR region

PR region Avalon-ST Sink interface setting

Select Yes or No to enable or disable interface ports

Signal Name	Enable the Avalon-ST S...
channel	Yes
error	Yes
valid	Yes
empty	Yes
ready	Yes
endofpacket&startofpa...	Yes

Channel width: 1

Error width: 1

Data bits per symbol: 8

Symbols per beat: 4

Error descriptors:

Max channel number: 1

Ready latency: 0

Table 37. Parameters for the Avalon-ST Partial Reconfiguration Freeze Bridge

Parameter	Values	Description
Parameters		
PR region Interface Type	**Avalon-ST Source/Avalon-ST Sink**	Specifies the interface type for interfacing with the freeze bridge.
Enable Freeze port from PR region	**On/Off**	Enables the freeze port to freeze all the outputs of each PR region to a known constant value. Freezing prevents the signal receivers in the static region from receiving undefined signals during the partial reconfiguration process.
PR region Avalon ST Source/Sink Interface Setting		

continued...

Parameter	Values	Description
Select Yes or No to enable or disable interface ports	**Yes/No**	Enables or disables specific optional freeze bridge interface ports.
Channel width	*<1-128>*	Specifies the width of the channel signal.
Error width	*<1-256>*	Specifies the width of the error signal.
Data bits per symbol	*<1-512>*	Specifies the number of bits per symbol.
Symbols per beat	*<1-512>*	Specifies the number of symbols that transfer on every valid clock cycle.
Error descriptors	*<text>*	Specifies one or more strings to describe the error condition for each bit of the error port on the sink interface connected to the source interface. Click the plus or minus buttons to add or remove descriptors.
Max channel number	*<0-255>*	Specifies the maximum number of output channels.
Ready latency	*<0-8>*	Specifies what ready latency to expect from the source interface connected to the sink interface. The ready latency is the number of cycles from the time `ready` asserts until valid data is driven.

5.2 Avalon-ST Partial Reconfiguration Freeze Bridge Interface Ports

Table 38. Avalon-ST Partial Reconfiguration Freeze Bridge Interface Ports

Port	Width	Direction	Description
`clock_clk`	1	Input	Input clock for the IP.
`freeze_conduit_freeze`	1	Input	When this signal is high, the bridge handles any current transaction properly then freezes the PR interfaces.
`freeze_conduit_illegal_request`	1	Output	High on this bus indicate that an illegal request is issued to the bridge during freezing state. *n* – number of freeze bridge
`pr_freeze_pr_freeze`	1	Input	Enabled freeze port from the PR region.
`reset_n_reset_n`	1	Input	Synchronous reset for the IP.

Table 39. Avalon-ST Sink to Static Region Interface Ports

Same setting as Avalon-ST sink to PR region interface.

Port	Width	Direction	Description
`sink_bridge_to_sr_channel`	1	Input	Avalon-ST sink bridge to static region `channel` port.
`sink_bridge_to_sr_data`	32	Input	Avalon-ST sink bridge to static region `data` port.
`sink_bridge_to_sr_empty`	2	Input	Avalon-ST sink bridge to static region `empty` port.
`sink_bridge_to_sr_error`	1	Input	Avalon-ST sink bridge to static region `error` port.

continued...

Port	Width	Direction	Description
sink_bridge_to_sr_ready	1	Output	Avalon-ST sink bridge to static region ready port.
sink_bridge_to_sr_valid	1	Input	Avalon-ST sink bridge to static region valid port.
sink_bridge_to_sr_endofpacket	1	Input	Avalon-ST sink bridge to static region endofpacket port.
sink_bridge_to_sr_startofpacket	1	Input	Avalon-ST sink bridge to static region startofpacket port.

Table 40. Avalon-ST Sink to PR Region Interface Ports

Port	Width	Direction	Description
sink_bridge_to_pr_channel	1	Output	Optional Avalon-ST sink bridge to PR region channel port.
sink_bridge_to_pr_data	32	Output	Optional Avalon-ST sink bridge to PR region data port.
sink_bridge_to_pr_empty	2	Output	Optional Avalon-ST sink bridge to PR region empty port.
sink_bridge_to_pr_error	1	Output	Optional Avalon-ST sink bridge to PR region error port.
sink_bridge_to_pr_ready	1	Input	Optional Avalon-ST sink bridge to PR region ready port.
sink_bridge_to_pr_valid	1	Output	Optional Avalon-ST sink bridge to PR region valid port.
sink_bridge_to_pr_endofpacket	1	Output	Optional Avalon-ST sink bridge to PR region endofpacket port.
sink_bridge_to_pr_startofpacket	1	Output	Optional Avalon-ST sink bridge to PR region startofpacket port.

Table 41. Avalon-ST Source to Static Region Interface Ports

Same setting as Avalon-ST source to PR region interface.

Port	Width	Direction	Description
source_bridge_to_sr_channel	1	Output	Avalon-ST source bridge to static region channel port.
source_bridge_to_sr_data	32	Output	Avalon-ST source bridge to static region data port.
source_bridge_to_sr_empty	2	Output	Avalon-ST source bridge to static region empty port.
source_bridge_to_sr_error	1	Output	Avalon-ST source bridge to static region error port.
source_bridge_to_sr_ready	1	Input	Avalon-ST source bridge to static region ready port.

continued...

Port	Width	Direction	Description
`source_bridge_to_sr_valid`	1	Output	Avalon-ST source bridge to static region `valid` port.
`source_bridge_to_sr_endofpacket`	1	Output	Avalon-ST source bridge to static region `endofpacket` port.
`source_bridge_to_sr_startofpacket`	1	Output	Avalon-ST source bridge to static region `startofpacket` port.

Table 42. Avalon-ST Source to PR Region Interface Ports

Port	Width	Direction	Description
`source_bridge_to_pr_channel`	1	Input	Optional Avalon-ST source bridge to PR region `channel` port.
`source_bridge_to_pr_data`	32	Input	Optional Avalon-ST source bridge to PR region `data` port.
`source_bridge_to_pr_empty`	2	Input	Optional Avalon-ST source bridge to PR region `empty` port.
`source_bridge_to_pr_error`	1	Input	Optional Avalon-ST source bridge to PR region `error` port.
`source_bridge_to_pr_ready`	1	Output	Optional Avalon-ST source bridge to PR region `ready` port.
`source_bridge_to_pr_valid`	1	Input	Optional Avalon-ST source bridge to PR region `valid` port.
`source_bridge_to_pr_endofpacket`	1	Input	Optional Avalon-ST source bridge to PR region `endofpacket` port.
`source_bridge_to_pr_startofpacket`	1	Input	Optional Avalon-ST source bridge to PR region `startofpacket` port.

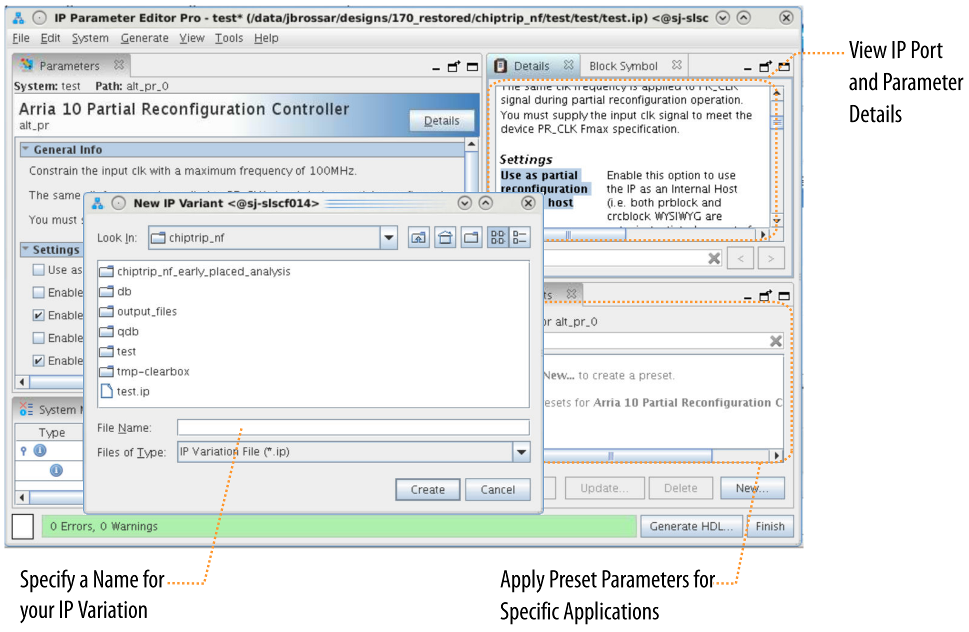
6 Generating and Simulating the IP

Use the following information to generate and simulate an IP core variation.

6.1 Generating IP Cores (Quartus Prime Pro Edition)

Quickly configure a custom IP variation in the Quartus Prime parameter editor. Double-click any component in the IP Catalog to launch the parameter editor. The parameter editor allows you to define a custom variation of the selected IP core. The parameter editor generates the IP variation synthesis and optional simulation files, and adds the `.ip` file representing the variation to your project automatically.

Figure 20. IP Parameter Editor (Quartus Prime Pro Edition)


IP Parameter Editor Pro - test* (/data/jbrossar/designs/170_restored/chiptrip_nf/test/test/test.ip) <@sj-slsc
File Edit System Generate View Tools Help
Parameters
Details
Block Symbol
Arria 10 Partial Reconfiguration Controller
alt_pr
General Info
Settings
New IP Variant <@sj-slscf014>
Look In: chiptrip_nf
chiptrip_nf_early_placed_analysis
db
output_files
qdb
test
tmp-clearbox
test.ip
File Name:
Files of Type: IP Variation File (*.ip)
Create
Cancel
Update...
Delete
New...
0 Errors, 0 Warnings
Generate HDL...
Finish
View IP Port and Parameter Details
Specify a Name for your IP Variation
Apply Preset Parameters for Specific Applications

System: test Path: alt_pr_0

Constrain the input clk with a maximum frequency of 100MHz.

The same clk frequency is applied to PR_CLK signal during partial reconfiguration operation. You must supply the input clk signal to meet the device PR_CLK Fmax specification.

Settings

Use as partial reconfiguration host	Enable this option to use the IP as an Internal Host (i.e. both prblock and crcblock WYSIWYG are

Follow these steps to locate, instantiate, and customize an IP core in the parameter editor:

1. Create or open a Quartus Prime project (`.qpf`) to contain the instantiated IP variation.
2. In the IP Catalog (**Tools ➤ IP Catalog**), locate and double-click the name of the IP core to customize. To locate a specific component, type some or all of the component's name in the IP Catalog search box. The New IP Variation window appears.
3. Specify a top-level name for your custom IP variation. Do not include spaces in IP variation names or paths. The parameter editor saves the IP variation settings in a file named <*your_ip*>`.ip`. Click **OK**. The parameter editor appears.
4. Set the parameter values in the parameter editor and view the block diagram for the component. The **Parameterization Messages** tab at the bottom displays any errors in IP parameters:
   - Optionally, select preset parameter values if provided for your IP core. Presets specify initial parameter values for specific applications.
   - Specify parameters defining the IP core functionality, port configurations, and device-specific features.
   - Specify options for processing the IP core files in other EDA tools.

   Note: Refer to your IP core user guide for information about specific IP core parameters.
5. Click **Generate HDL**. The **Generation** dialog box appears.
6. Specify output file generation options, and then click **Generate**. The synthesis and/or simulation files generate according to your specifications.
7. To generate a simulation testbench, click **Generate ➤ Generate Testbench System**. Specify testbench generation options, and then click **Generate**.
8. To generate an HDL instantiation template that you can copy and paste into your text editor, click **Generate ➤ Show Instantiation Template**.
9. Click **Finish**. Click **Yes** if prompted to add files representing the IP variation to your project.
10. After generating and instantiating your IP variation, make appropriate pin assignments to connect ports.

    Note: Some IP cores generate different HDL implementations according to the IP core parameters. The underlying RTL of these IP cores contains a unique hash code that prevents module name collisions between different variations of the IP core. This unique code remains consistent, given the same IP settings and software version during IP generation. This unique code can change if you edit the IP core's parameters or upgrade the IP core version. To avoid dependency on these unique codes in your simulation environment, refer to *Generating a Combined Simulator Setup Script*.

Related Links

- Introduction to Intel FPGA IP Cores
- Generating a Combined Simulator Setup Script

6.1.1 Running the Freeze Bridge Update script

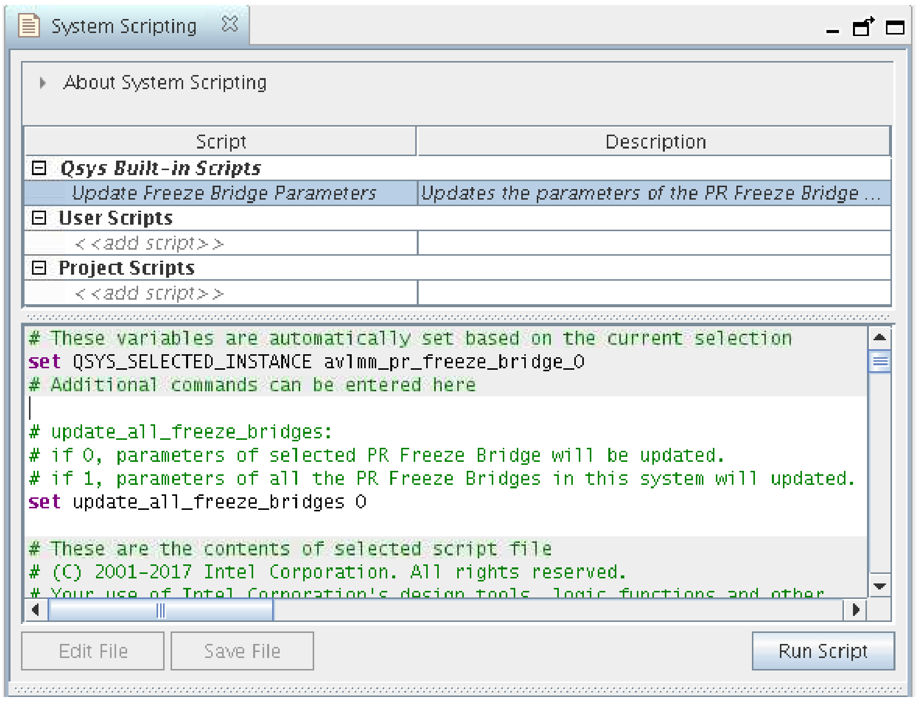
When instantiating the Freeze Bridge as a Qsys Pro system component, the interface connections between the Freeze Bridge and the PR region must match, so that Qsys Pro inserts no extra interconnect during system generation. Rather than manually matching the Avalon interface properties individually in the parameter editor, you can run the provided Update Freeze Bridge Parameters script to update Freeze Bridge Avalon interface properties automatically.

Running this script updates the master and slave interfaces or the sink and source interfaces of the Freeze Bridge, according to the Avalon property settings of the connecting PR region component.

To run the Update Freeze Bridge Parameters script:

1. Open a Qsys Pro system containing one or more instances of the Freeze Bridge component.
2. In Qsys Pro, click **View ➤ System Scripting**. The **System Scripting** tab displays **Qsys Built-in Scripts**.
3. To update all freeze bridges in your Qsys Pro system, set `update_all_freeze_bridges 1` in the Additional Commands section of the script. To update only a single freeze bridge, click the freeze bridge instance.
4. Click **Run Script**. The script runs and updates the freeze bridge parameters.

Figure 21. Qsys Pro System Scripting Tab



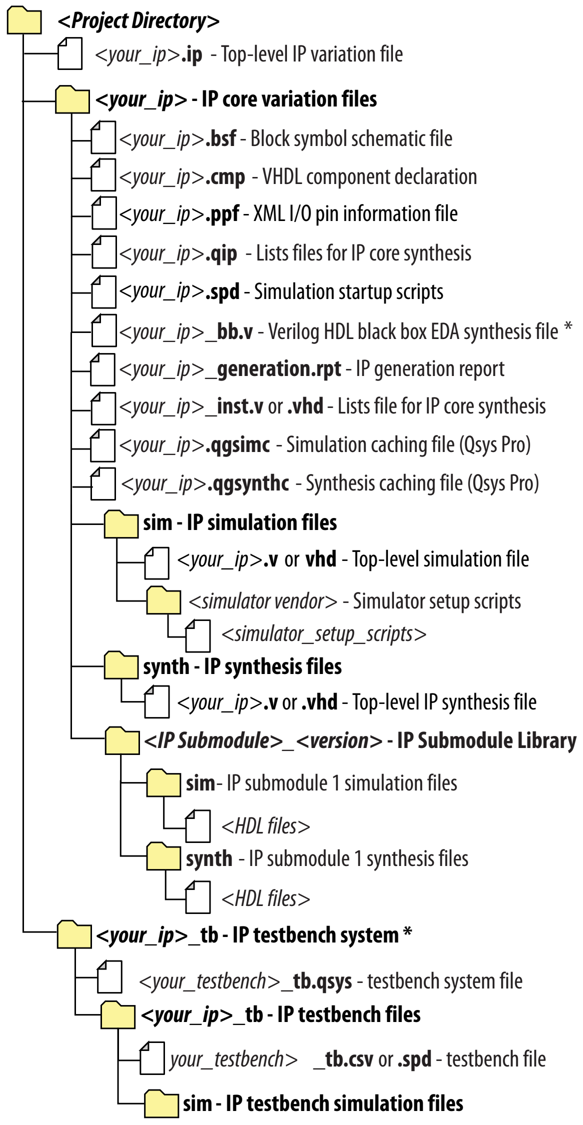
Script	Description
Qsys Built-in Scripts	
Update Freeze Bridge Parameters	Updates the parameters of the PR Freeze Bridge ...
User Scripts	
<<add script>>	
Project Scripts	
<<add script>>	

```
# These variables are automatically set based on the current selection
set QSYS_SELECTED_INSTANCE avlmm_pr_freeze_bridge_0
# Additional commands can be entered here

# update_all_freeze_bridges:
# if 0, parameters of selected PR Freeze Bridge will be updated.
# if 1, parameters of all the PR Freeze Bridges in this system will updated.
set update_all_freeze_bridges 0

# These are the contents of selected script file
# (C) 2001-2017 Intel Corporation. All rights reserved.
# Your use of Intel Corporation's design tools, logic functions and other
```

6.1.2 IP Core Generation Output (Quartus Prime Pro Edition)

The Quartus Prime software generates the following output file structure for individual IP cores that are not part of a Qsys Pro system.

Figure 22. Individual IP Core Generation Output (Quartus Prime Pro Edition)


<Project Directory>
<your_ip>.ip - Top-level IP variation file
<your_ip> - IP core variation files
<your_ip>.bsf - Block symbol schematic file
<your_ip>.cmp - VHDL component declaration
<your_ip>.ppf - XML I/O pin information file
<your_ip>.qip - Lists files for IP core synthesis
<your_ip>.spd - Simulation startup scripts
<your_ip>_bb.v - Verilog HDL black box EDA synthesis file *
<your_ip>_generation.rpt - IP generation report
<your_ip>_inst.v or .vhd - Lists file for IP core synthesis
<your_ip>.qgsimc - Simulation caching file (Qsys Pro)
<your_ip>.qgsynthc - Synthesis caching file (Qsys Pro)
sim - IP simulation files
<your_ip>.v or vhd - Top-level simulation file
<simulator vendor> - Simulator setup scripts
<simulator_setup_scripts>
synth - IP synthesis files
<your_ip>.v or .vhd - Top-level IP synthesis file
<IP Submodule>_<version> - IP Submodule Library
sim- IP submodule 1 simulation files
<HDL files>
synth - IP submodule 1 synthesis files
<HDL files>
<your_ip>_tb - IP testbench system *
<your_testbench>_tb.qsys - testbench system file
<your_ip>_tb - IP testbench files
your_testbench> _tb.csv or .spd - testbench file
sim - IP testbench simulation files

* If supported and enabled for your IP core variation.

Table 43. Files Generated for IP Cores

File Name	Description
`<your_ip>.ip`	Top-level IP variation file that contains the parameterization of an IP core in your project. If the IP variation is part of a Qsys Pro system, the parameter editor also generates a `.qsys` file.
`<your_ip>.cmp`	The VHDL Component Declaration (`.cmp`) file is a text file that contains local generic and port definitions that you use in VHDL design files.
`<your_ip>_generation.rpt`	IP or Qsys Pro generation log file. Displays a summary of the messages during IP generation.
`<your_ip>.qgsimc` (Qsys Pro systems only)	Simulation caching file that compares the `.qsys` and `.ip` files with the current parameterization of the Qsys Pro system and IP core. This comparison determines if Qsys Pro can skip regeneration of the HDL.
`<your_ip>.qgsynth` (Qsys Pro systems only)	Synthesis caching file that compares the `.qsys` and `.ip` files with the current parameterization of the Qsys Pro system and IP core. This comparison determines if Qsys Pro can skip regeneration of the HDL.
`<your_ip>.qip`	Contains all information to integrate and compile the IP component.
`<your_ip>.csv`	Contains information about the upgrade status of the IP component.
`<your_ip>.bsf`	A symbol representation of the IP variation for use in Block Diagram Files (`.bdf`).
`<your_ip>.spd`	Required input file for `ip-make-simscript` to generate simulation scripts for supported simulators. The `.spd` file contains a list of files you generate for simulation, along with information about memories that you initialize.
`<your_ip>.ppf`	The Pin Planner File (`.ppf`) stores the port and node assignments for IP components you create for use with the Pin Planner.
`<your_ip>_bb.v`	Use the Verilog blackbox (`_bb.v`) file as an empty module declaration for use as a blackbox.
`<your_ip>_inst.v` or `_inst.vhd`	HDL example instantiation template. Copy and paste the contents of this file into your HDL file to instantiate the IP variation.
`<your_ip>.regmap`	If the IP contains register information, the Quartus Prime software generates the `.regmap` file. The `.regmap` file describes the register map information of master and slave interfaces. This file complements the `.sopcinfo` file by providing more detailed register information about the system. This file enables register display views and user customizable statistics in System Console.
`<your_ip>.svd`	Allows HPS System Debug tools to view the register maps of peripherals that connect to HPS within a Qsys Pro system. During synthesis, the Quartus Prime software stores the `.svd` files for slave interface visible to the System Console masters in the `.sof` file in the debug session. System Console reads this section, which Qsys Pro queries for register map information. For system slaves, Qsys Pro accesses the registers by name.
`<your_ip>.v` `<your_ip>.vhd`	HDL files that instantiate each submodule or child IP core for synthesis or simulation.
`mentor/`	Contains a script `msim_setup.tcl` to set up and run a simulation.
`aldec/`	Contains a Riviera*-PRO script `rivierapro_setup.tcl` to setup and run a simulation.
`/synopsys/vcs` `/synopsys/vcsmx`	Contains a shell script `vcs_setup.sh` to set up and run a VCS* simulation. Contains a shell script `vcsmx_setup.sh` and `synopsys_sim.setup` file to set up and run a VCS MX* simulation.

continued...

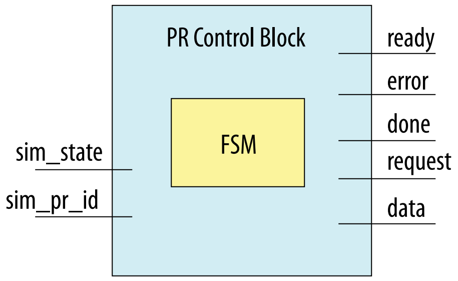
File Name	Description
/cadence	Contains a shell script ncsim_setup.sh and other setup files to set up and run an NCSIM simulation.
/submodules	Contains HDL files for the IP core submodule.
<IP submodule>/	For each generated IP submodule directory, Qsys Pro generates /synth and /sim sub-directories.

6.2 Arria 10 PR Control Block Simulation Model

The Quartus Prime Pro Edition software supports simulating the delivery of a partial reconfiguration bitstream to the PR control block. This simulation allows you to observe the resulting change and the intermediate effect in a reconfigurable partition.

The Arria 10 PR control block supports PR simulation. Sending a simulation RBF (PR bitstream) allows the PR control block to behave accordingly, to PR simulation success or PR simulation failure. To activate simulation of a specific PR persona in your PR region simulation wrapper, use a PR ID encoded in the simulation RBF, in conjunction with the PR control block. Simulate the PR control block either as standalone, or as part of the simulation file set for the Arria 10 Partial Reconfiguration Controller IP core. For complete information on the Arria 10 PR control block simulation model (twentynm_prblock), refer to the <*installation directory*>/eda/sim_lib/altera_lnsim.sv file.

Figure 23. Arria 10 PR Control Block Simulation Model



The Arria 10 PR control block simulation model contains two additional simulation-only ports—sim_state and sim_pr_id. Connect these simulation ports, and the other ports, to the twentynm_prblock_if SystemVerilog interface. This connection allows monitoring of the PR control block using your testbench's PR control block monitor. The Quartus Prime software automatically instantiates the twentynm_prblock_if interface when generating the simulation file set of the Arria 10 Partial Reconfiguration IP core. Obtain a reference to the twentynm_prblock_if instantiated in the PR Controller IP by using the alt_pr_test_pkg::twentynm_prblock_if_mgr singleton, as shown in the following example:

```
virtual twentynm_prblock_if prblock_if;

alt_pr_test_pkg::twentynm_prblock_if_mgr cb_mgr;

// Get the PR control block from the prblock manager
cb_mgr = alt_pr_test_pkg::twentynm_prblock_if_mgr::get();
prblock_if = cb_mgr.if_ref;
```

The code for the `twentynm_prblock_if` interface is as follows:

```
interface twentynm_prblock_if(input logic pr_clk, input logic clk);

    logic prrequest;
    logic [31:0] data;
    wire error;
    wire ready;
    wire done;
    logic [31:0] sim_only_state;
    wire [31:0] sim_only_pr_id;

    // All signals are async except data
    clocking cb1 @(posedge pr_clk);
        output data;
    endclocking

endinterface : twentynm_prblock_if
```

For more information on the `twentynm_prblock_if` interface, refer to the *<installation directory>*`/eda/sim_lib/altera_lnsim.sv` file.

The simulation state of the Arria 10 PR control block simulation model represents the PR_EVENT_TYPE enumeration state of the control block. The `twentynm_prblock_test_pkg` SystemVerilog package defines these enumerations. These states represent the different allowed states for the control block. The defined control block enumerations are as follows:

```
package twentynm_prblock_test_pkg;
    typedef enum logic [31:0] {
        NONE,
        IDLE,
        PR_REQUEST,
        PR_IN_PROGRESS,
        PR_COMPLETE_SUCCESS,
        PR_COMPLETE_ERROR,
        PR_INCOMPLETE_EARLY_WITHDRAWL,
        PR_INCOMPLETE_LATE_WITHDRAWL
    } PR_EVENT_TYPE;
```

When the simulation state is PR_IN_PROGRESS, the affected PR region must have its simulation output multiplexes driven to X, by asserting the `pr_activate` signal. This action simulates the unknown outputs of the PR region during partial reconfiguration. In addition, you must assert the `pr_activate` signal in the PR simulation model to load all registers in the PR model with the PR activation value.

Once the simulation state reaches PR_COMPLETE_SUCCESS, activate the appropriate PR persona using the appropriate PR region simulation wrapper mux `sel` signals. You can decode the region, as well as the specific select signal from the `sim_only_pr_id` signal of the PR control block. This ID corresponds to the encoded ID in the simulation RBF.

Table 44. Required Sequence of Words in Simulation RBF

Zero or more of the following words can be written in step 1. For all other steps, only 1 word is written.

1	zero padding blocks	0x00000000
2	PR_HEADER_WORD	0x0000A65C
3	PR_ID	32-bit user ID

continued...

4	PRDATA_COUNT_0	0x01234567
5	PRDATA_COUNT_1	0x89ABCDEF
6	PRDATA_COUNT_2	0x02468ACE
7	PRDATA_COUNT_3	0x13579BDF

Note: The `PR_ID` word is output on the `sim_only_pr_id` word, starting at `PRDATA_COUNT_0`. Using a different value for the header or data count results in PR simulation errors.

Related Links

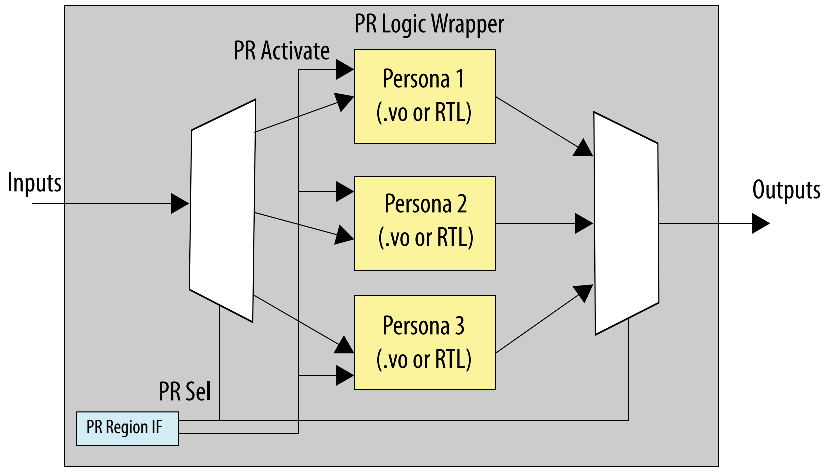
- Simulating Intel FPGA IP Cores
- Simulating Intel FPGA Designs
- Partial Reconfiguration Simulation and Verification

6.3 Simulating PR Persona Replacement

The logical operation of the PR partition changes when a new persona is loaded during the partial reconfiguration process. Simulate the replacement of personas using multiplexors on the input and output of the persona under simulation. Create RTL wrapper logic to represent the top-level of the persona. The wrapper instantiates the default persona during compilation. During simulation, the wrapper allows the replacement of the active persona with another persona. Instantiate each persona as either the behavioral RTL in the PR simulation model the Quartus Prime EDA Netlist Writer generates. The Quartus Prime software includes simulation modules to interface with your simulation testbench:

- `altera_pr_wrapper_mux_in`
- `altera_pr_wrapper_mux_out`
- `altera_pr_persona_if` (SystemVerilog interface allows you to connect the wrapper multiplexes to a testbench driver)

Figure 24. Simulation of PR Persona Switching



Example 1. RTL Wrapper for PR Persona Switching Simulation

The `pr_activate` input of the `altera_pr_wrapper_mux_out` module enables the MUX to output X. This functionality allows the simulation of unknown outputs from the PR persona, and also verifies the normal operation of design's freeze logic. The following code corresponds the simulation of PR persona switching, shown in the above figure:

```
module pr_core_wrapper
(
   input wire a,
   input wire b,
   output wire o
);

localparam ENABLE_PERSONA_1 = 1;
localparam ENABLE_PERSONA_2 = 1;
localparam ENABLE_PERSONA_3 = 1;
localparam NUM_PERSONA = 3;

logic pr_activate;
int persona_select;

altera_pr_persona_if persona_bfm();
assign pr_activate = persona_bfm.pr_activate;
assign persona_select = persona_bfm.persona_select;


wire a_mux [NUM_PERSONA-1:0];
wire b_mux [NUM_PERSONA-1:0];
wire o_mux [NUM_PERSONA-1:0];

generate
   if (ENABLE_PERSONA_1) begin
      localparam persona_id = 0;

      `ifdef ALTERA_ENABLE_PR_MODEL
         assign u_persona_0.altera_sim_pr_activate = pr_activate;
      `endif

      pr_and u_persona_0
      (
         .a(a_mux[persona_id]),
         .b(b_mux[persona_id]),
         .o(o_mux[persona_id])
      );
   end
endgenerate

generate
   if (ENABLE_PERSONA_2) begin
      localparam persona_id = 1;

      `ifdef ALTERA_ENABLE_PR_MODEL
         assign u_persona_1.altera_sim_pr_activate = pr_activate;
      `endif

      pr_or u_persona_1
      (
         .a(a_mux[persona_id]),
         .b(b_mux[persona_id]),
         .o(o_mux[persona_id])
      );

   end
endgenerate

generate
```

```
   if (ENABLE_PERSONA_3) begin
      localparam persona_id = 2;

      `ifdef ALTERA_ENABLE_PR_MODEL
         assign u_persona_2.altera_sim_pr_activate = pr_activate;
      `endif

      pr_empty u_persona_2
      (
         .a(a_mux[persona_id]),
         .b(b_mux[persona_id]),
         .o(o_mux[persona_id])
      );

   end
endgenerate


altera_pr_wrapper_mux_in #(.NUM_PERSONA(NUM_PERSONA), .WIDTH(1))
u_a_mux(.sel(persona_select), .mux_in(a), .mux_out(a_mux));

altera_pr_wrapper_mux_in #(.NUM_PERSONA(NUM_PERSONA), .WIDTH(1))
u_b_mux(.sel(persona_select), .mux_in(b), .mux_out(b_mux));

altera_pr_wrapper_mux_out #(.NUM_PERSONA(NUM_PERSONA), .WIDTH(1))
u_o_mux(.sel(persona_select), .mux_in(o_mux), .mux_out(o), .pr_activate(pr_activat
e));

endmodule
```

6.3.1 PR Simulation Wrapper Modules

6.3.1.1 `altera_pr_wrapper_mux_in` Module

The `altera_pr_wrapper_mux_in` module allows you to de-multiplex inputs to a PR partition wrapper for all PR personas.

Instantiate one multiplexor per input port. Specify the active persona using the `sel` port of the multiplexor. Parameterize the component to specify the number of persona output and the width of the multiplexor.

```
module altera_pr_wrapper_mux_in(sel, mux_in, mux_out);
    parameter NUM_PERSONA = 1;
    parameter WIDTH = 1;

    input int sel;
    input wire [WIDTH-1:0] mux_in;
    output reg [WIDTH-1 : 0] mux_out [NUM_PERSONA-1:0];

    always_comb begin
        for (int i = 0; i < NUM_PERSONA; i++)
            if (i == sel)
                mux_out[i] = mux_in;
            else
                mux_out[i] = 'x;
    end


endmodule : altera_pr_wrapper_mux_in
```

The `altera_pr_wrapper_mux_in` component is defined in the `altera_lnsim.sv` file, located in *<QUARTUS_INSTALL_DIR>*`/eda/sim_lib/altera_lnsim.sv`.

6.3.1.2 altera_pr_wrapper_mux_out Module

The altera_pr_wrapper_mux_out module allows you to multiplex the outputs of all PR personas to the outputs of the PR region wrapper.

Instantiate one multiplexor per output port. Specify the active persona using the sel port of the multiplexor. The optional pr_activate port allows you to drive the multiplexor output to "x", to emulate the unknown value of PR region outputs during a PR operation. Parameterize the component to specify the number of persona input and the width of the multiplexor.

```
module altera_pr_wrapper_mux_out(sel, mux_in, mux_out, pr_activate);
    parameter NUM_PERSONA = 1;
    parameter WIDTH = 1;

    input int sel;
    input wire [WIDTH-1 : 0] mux_in [NUM_PERSONA-1:0];
    output reg [WIDTH-1:0]   mux_out;
    input wire               pr_activate;

    always_comb begin
        if ((sel < NUM_PERSONA) && (!pr_activate))
            mux_out = mux_in[sel];
        else
            mux_out = 'x;
    end

endmodule : altera_pr_wrapper_mux_out
```

The altera_pr_wrapper_mux_out component is defined in the altera_lnsim.sv file, located in *<QUARTUS_INSTALL_DIR>*/eda/sim_lib/altera_lnsim.sv.

6.3.1.3 altera_pr_persona_if Module

Instantiate the altera_pr_persona_if SystemVerilog interface in a PR region simulation wrapper to connect to all the wrapper multiplexors. Optionally, connect pr_activate to the PR simulation model.

Connect the interface's persona_select to the sel port of all input and output multiplexes. Connect the pr_activate to the pr_activate of all the output multiplexes. Then, the PR region driver testbench component can drive the interface.

```
interface altera_pr_persona_if;
    logic pr_activate;
    int   persona_select;

    initial begin
        pr_activate <= 1'b0;
    end
endinterface : altera_pr_persona_if
```

The altera_pr_persona_if component is defined in the altera_lnsim.sv file, located in *<QUARTUS_INSTALL_DIR>*/eda/sim_lib/altera_lnsim.sv.

7 Document Revision History

Date	Version	Changes
2017.05.08	17.0.0	Initial public release.