

Configuration via Protocol (CvP) Implementation in Altera FPGAs User Guide

Contents

Overview.....	1-1
Benefits of Using CvP.....	1-1
CvP System.....	1-1
CvP Modes.....	1-2
Prepare the Design for CvP Revision Design Flow.....	1-3
CvP Topologies.....	1-5
Additional Information about PCI Express.....	1-5
FPGA Configuration using CvP.....	2-1
CvP Configuration Images.....	2-1
CvP Modes.....	2-1
CvP Initialization Mode.....	2-1
CvP Update Mode.....	2-2
Core Image Update.....	2-2
CvP Pins.....	2-4
CvP Topologies.....	3-1
Single Endpoint.....	3-1
Multiple Endpoints.....	3-1
Mixed Chain.....	3-2
Configuring Slave FPGAs with Different Configuration Files.....	3-3
Configuring Slave FPGAs with a Single Configuration File.....	3-3
Daisy Chain.....	3-4
Design Considerations.....	4-1
Data Compression.....	4-1
Data Encryption.....	4-1
CvP Features	4-2
Core Image Update.....	4-2
Partial Reconfiguration.....	4-2
Designing CvP for an Open System.....	4-2
FPGA Power Supplies Ramp Time Requirement.....	4-2

PCIe Wake-Up Time Requirement.....	4-3
Design CvP for a Closed System.....	4-8
Clock Connections for CvP Designs Including the Transceiver Reconfiguration Controller.....	4-9
CvP Example Designs.....	5-1
Understanding the Design Steps for CvP Initialization Mode.....	5-2
Downloading and Generating the High Performance Reference Design.....	5-3
Setting up CvP Parameters in Device and Pin Options GUI for CvP Initialization Mode.....	5-5
Compiling the Design for the CvP Initialization Mode.....	5-7
Splitting the SOF File for the CvP Initialization Design Mode.....	5-7
Understanding the Design Steps for CvP Initialization Mode with the Revision Design Flow.....	5-9
Downloading and Generating the High Performance Reference Design.....	5-11
Workaround for a Known Issue with Transceiver Reconfiguration Controller IP Core.....	5-13
Creating An Alternate user_led.v File for the Reconfigurable Core Region.....	5-13
Setting up CvP Parameters in Device and Pin Options GUI for CvP Initialization Mode.....	5-14
Creating CvP Revisions of the Core Logic Region Using the CvP Revision Design Flow.....	5-15
Compiling Both the Base and cvp_app Revisions in the CvP Revision Design Flow.....	5-18
Splitting the SOF File for the CvP Initialization Design Mode.....	5-19
Splitting the SOF File for the CvP Initialization Mode with the CvP Revision Design Flow.....	5-21
Understanding the Design Steps for CvP Update Mode.....	5-23
Downloading and Generating the High Performance Reference Design.....	5-24
Workaround for a Known Issue with Transceiver Reconfiguration Controller IP Core.....	5-26
Creating An Alternate user_led.v File for the Reconfigurable Core Region.....	5-26
Setting up CvP Parameters in Device and Pin Options GUI for CvP Update Mode for CvP Revision.....	5-27
Creating CvP Revisions of the Core Logic Region Using the CvP Revision Design Flow.....	5-28
Setting up CvP Parameters in Device and Pin Options GUI for CvP Update Mode for CvP Revision.....	5-31
Compiling the Design for the CvP Update Mode.....	5-33
Splitting the SOF File for the CvP Update Design Mode.....	5-33

Splitting the SOF File for CvP Update Mode with the CvP Revision Design Flow.....	5-35
Bringing Up the Hardware	5-37
Installing Jungo WindDriver in Windows Systems.....	5-38
Installing Jungo WinDriver in Linux Systems.....	5-38
Modifying MSEL for Active Serial x4 Flash on Stratix V Dev-Kit.....	5-38
Programming CvP Images and Validating the Link.....	5-39
CvP Debugging Check List.....	5-43
Known Issues and Solutions.....	5-45
CvP Driver and Registers.....	6-1
CvP Driver Support.....	6-1
CvP Driver Flow.....	6-1
VSEC Registers for CvP.....	6-2
Altera-defined Vendor Specific Capability Header Register.....	6-3
Altera-defined Vendor Specific Header Register.....	6-3
Altera Marker Register.....	6-4
CvP Status Register.....	6-4
CvP Mode Control Register.....	6-5
CvP Data Registers.....	6-6
CvP Programming Control Register.....	6-7
Uncorrectable Internal Error Status Register.....	6-7
Uncorrectable Internal Error Mask Register.....	6-8
Correctable Internal Error Status Register.....	6-9
Correctable Internal Error Mask Register.....	6-10
Additional Information.....	7-1
Document Revision History.....	7-1
How to Contact Altera.....	7-2

2013.11.04

UG-01101

 [Subscribe](#)  [Send Feedback](#)

Configuration via Protocol (CvP) is a configuration scheme supported in Arria V[®], Cyclone V[®], and Stratix V[®] devices⁽¹⁾. The CvP configuration scheme creates separate images for the periphery and core logic. You can store the periphery image in the configuration device and the core image in host memory, reducing system costs and increasing the security for the proprietary core image. CvP is available for Endpoint variants.

Benefits of Using CvP

CvP configuration scheme has the following advantages:

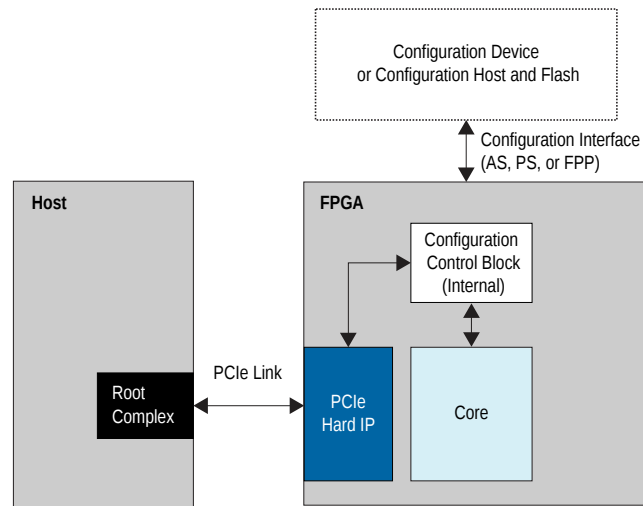
- Reduces system costs by reducing the size of the flash device to store the periphery configuration data.
- Improves security for the proprietary core bitstream. CvP ensures the PCIe host can exclusively access the FPGA core image.
- Enables dynamic core updates without requiring a system power down. CvP allows the FPGA fabric to be updated through the PCIe link without a host reboot or FPGA full chip reinitialization.
- Provides a simpler software model for configuration. A smart host can use the PCIe protocol and the application topology to initialize and update the FPGA fabric.
- Facilitates hardware acceleration.
- Reduces system size. You can use a single PCIe link to configure multiple FPGAs.

CvP System

The following figure shows the setup for the CvP system.

⁽¹⁾ CvP is an advanced feature for the Arria V and Cyclone V device families. For details, contact Altera for support.

Figure 1-1: CvP Block Diagram



A CvP system typically consists of an FPGA, a PCIe host, and a configuration device.

Most Arria V, Cyclone V, and Stratix V FPGAs include more than one Hard IP for PCI Express IP Cores. Only the bottom left PCIe Hard IP block can be used for the CvP configuration scheme. It must be configured as an Endpoint.

When you use CvP configuration scheme to program your V-series FPGA, you can program the periphery image first. Because the Hard IP for PCI Express is part of the periphery image, it can respond to configuration requests from a Root Port before the core image is programmed and the device enters user mode. This ability to respond to requests from a Root Port before the device is fully programmed is called autonomous mode. Autonomous mode allows the Hard IP for PCI Express IP Core to meet wake-up time requirements of open systems.

Prior to the V-series FPGA, the Hard IP for PCI Express IP Core did not operate in autonomous mode. The Hard IP for PCI Express IP Core was released from reset only after the FPGA core was fully configured.

The configuration device is connected to the FPGA using the conventional configuration interface. The configuration interface can be any of the supported schemes, such as active serial (AS), passive serial (PS), or fast passive parallel (FPP). The choice of the configuration device depends on your chosen configuration scheme.

CvP Modes

The CvP configuration scheme supports the following modes:

- CvP initialization mode
- CvP update mode

For Stratix V devices, both the CvP modes are supported in PCIe Gen 1. Contact Altera for PCIe Gen 2 support.

For Arria V and Cyclone V devices, both the CvP modes are supported in PCIe Gen 1. For PCIe Gen 2, only CvP update mode is supported.

CvP Initialization Mode

This mode initializes the core image of the FPGA through the PCIe link upon system power up. The PCIe link is also used for subsequent core image updates.

Choose this mode if you want to:

- Satisfy the PCIe wake-up time requirement
- Save cost by storing the core image in the external host memory
- Prevent unauthorized access to the core image
- Update the core image after the initial programming completes

CvP Update Mode

This mode assumes that you have configured the FPGA with the full configuration image from a configuration device after the initial system power up. The PCIe link is used for subsequent core image updates.

Choose this mode if you want to update the core image for any of the following reasons:

- To change core algorithms
- To perform standard updates as part of a release process
- To customize core processing for a different components that are part of a complex system
- To debug the CvP initialization mode

Related Information

- [CvP Initialization Mode](#) on page 2-1
- [CvP Update Mode](#) on page 2-2
- [CvP Example Designs](#) on page 5-1

Prepare the Design for CvP Revision Design Flow

The CvP revision design flow requires separate bitstreams for design elements implemented in the I/O ring (periphery) and FPGA core fabric. To use an I/O bitstream with multiple FPGA core fabric bitstreams, separate periphery from the reconfigurable core logic.

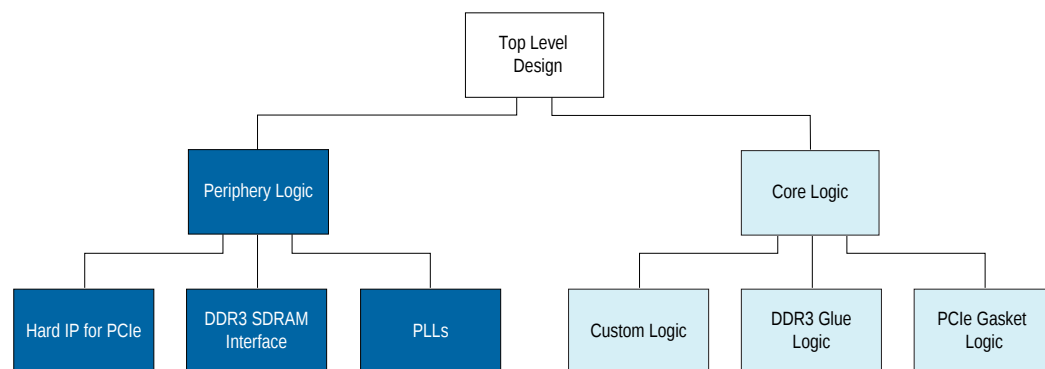
- The I/O ring (periphery) partition:
 - I/O ports
 - I/O registers
 - General-purpose I/Os (GPIOs)
 - Transceivers
 - Phase-locked loops (PLLs)
 - Hard IP for PCI Express
 - Hardened memory PHY
 - Global clocks (GCLK)
 - Regional clocks (RCLK)
- The core partition: Core logic to program the FPGA fabric. The core logic contains both the static core region and the reconfigurable core region. The core logic is stored in configuration RAM (CRAM) bits that program the logic array blocks (LABs), digital signal processing (DSP), and RAM of the core. You may create one or more partitions for the core fabric; however, only one partition can include the logic that you plan to reconfigure.

You must ensure the reconfigurable core logic does not contain any periphery components. Failure to make these connections results in the following Quartus® II compilation error:

```
Error (142040): Detected illegal nodes in reconfigurable partitions.
Only core logic is reconfigurable in this version of the Quartus II
software.
```

Figure 1-2: Recommended Design Hierarchy

The following figure shows the recommended design hierarchy for a design including the Hard IP PCI Express IP Core, an interface to DDR3 SDRAM, and core logic.



This design hierarchy represents the actual partition after the Quartus II compilation. You may have to run Quartus II compilation several times to ensure that the reconfigurable core logic does not contain any periphery elements. It may take several Quartus II compilations for you to completely separate the periphery and core logic.

CvP Topologies

You can configure the FPGA using the following topologies:

- Single endpoint—to configure a single device.
- Multiple endpoints—to configure multiple devices using a PCIe switch.
- Mixed chain—to configure multiple devices using a single configuration file or multiple configuration files for slave devices in the chain.
- Daisy chain—to configure multiple devices with two PCIe hard IP cores in the FPGA configured as an Endpoint and a Root Port.

Related Information

[CvP Topologies](#) on page 3-1

Additional Information about PCI Express

The following links provide information about the PCI Express specifications and Altera's offerings for PCI Express.

Related Information

- [PCI Express Base 2.1 Specification](#)
- [PCI Express Base 3.0 Specification](#)
- [PCI Express CEM 2.0 Specification](#)
- [Arria V Hard IP for PCI Express User Guide](#)
- [Arria V GZ Hard IP for PCI Express User Guide](#)
- [Cyclone V Hard IP for PCI Express User Guide](#)
- [Stratix V Hard IP for PCI Express User Guide](#)

2013.11.04

UG-01101



Subscribe



Send Feedback

CvP Configuration Images

In CvP, you partition your design into two images: core image and periphery image.

You use the Quartus II software to generate the images:

- Periphery image (***periph.jic**)—contains general purpose I/Os (GPIOs), I/O registers, the GCLK, QCLK, and RCLK clock networks, and logic that is implemented in hard IP such as the Hard IP for PCI Express IP Core. These components are included in the periphery image because they are controlled by I/O periphery register bits. The entire periphery image is static and cannot be reconfigured.
- Core image (***.core.rbf**)—contains logic that is programmed by configuration RAM (CRAM). This image includes LABs, DSP, and embedded memory. The core image consists of a single reconfigurable region or both static and reconfigurable regions.

Related Information

- [I/O Features in Arria V Devices](#)
Provides more information about the location of the transceiver banks and I/O banks.
- [I/O Features in Cyclone V Devices](#)
Provides more information about the location of the transceiver banks and I/O banks.
- [I/O Features in Stratix V Devices](#)
Provides more information about the location of the transceiver banks and I/O banks.

CvP Modes

CvP Initialization Mode

In this mode, you use the PCIe link to initialize your device after the initial system power up. You need to load the periphery image and the core image into the FPGA device. The periphery image is stored in the external configuration device and is loaded into the FPGA through the conventional configuration scheme. The core image is stored in a memory that is accessible by the PCIe host and is loaded into the FPGA through the PCIe link.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO
9001:2008
Registered



After the periphery image configuration is complete, the `CONF_DONE` signal goes high and allows the FPGA to start the PCIe link training. When the PCIe link training is complete, the PCIe link transitions to L0 state. The PCIe host then initiates the core image configuration through the PCIe link.

After the core image configuration is complete, the `CvP_CONFDONE` pin goes high, indicating the FPGA is fully configured.

After the FPGA is fully configured, the FPGA enters user mode. If you enabled the `INIT_DONE` signal, the `INIT_DONE` signal goes high after initialization is complete and the FPGA enters user mode.

In user mode, the PCIe links are available for normal PCIe applications. You can also use the PCIe link to perform an FPGA core image update. To perform the FPGA core image update, you create multiple FPGA core images in the Quartus II software that have identical connections to the periphery image. If the core image contains the reconfigurable core region, you must implement the CvP revision design flow.

CvP Update Mode

In this mode, the FPGA device is initialized after initial system power up by loading the full configuration image from the external configuration device to the FPGA device through the conventional configuration scheme.

After the full FPGA configuration image is complete, the `CONF_DONE` signal goes high.

After the FPGA is fully configured, the FPGA enters initialization and user mode. If you enable the `INIT_DONE` signal, the `INIT_DONE` signal goes high after initialization is completed and the FPGA enters user mode.

In user mode, the PCIe links are available for normal PCIe applications. You can use the PCIe link to perform an FPGA core image update. To perform the FPGA core image update, you create multiple FPGA core images in the Quartus II software that have identical connections to the periphery image. If the core image contains the reconfigurable core region, you must implement the CvP revision design flow.

Note: You cannot combine the features of CvP Update Mode and CvP Initialization Mode in a single design. For example, you cannot create a CvP Update image for your Quartus II project and then specify a CvP Initialization periphery image in your configuration scheme.

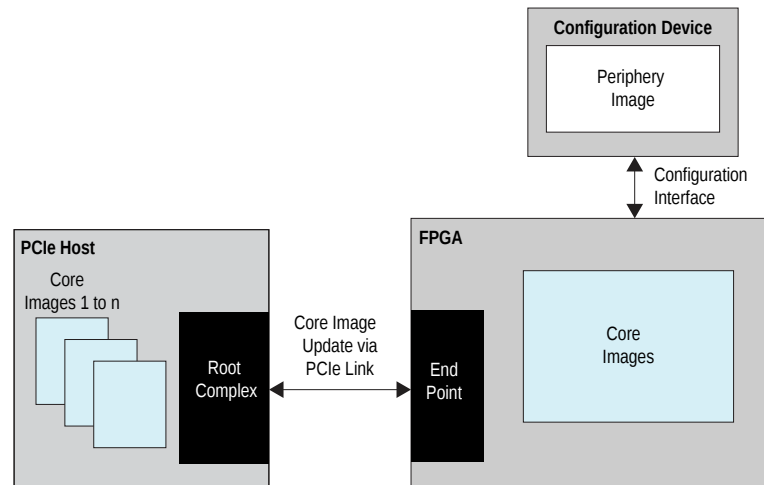
Core Image Update

After the FPGA enters user mode, the PCIe host can trigger an FPGA core image update through the PCIe link. Both CvP initialization mode and CvP update mode support core images updates.

You must choose the same bitstream settings for all core images. For example, if you have selected either encryption, compression, or both encryption and compression features for the first core image, you must ensure you turned on the same features for the other core images that you will use for core image update using CvP.

Figure 2-1: Periphery and Core Images Storage Arrangement for CvP Core Image Update

The periphery image remains the same for different core image updates. If you change the periphery image, you must reprogram the configuration device with the new periphery image.



You can use CvP revision design flow to create multiple reconfigurable core images that connect to the same periphery image.

The core logic consists of a single reconfigurable region or both static and reconfigurable regions.

- Reconfigured core logic—This region can be programmed in user mode while the PCIe link is up and fully enumerated. It must contain only resources that are controlled by CRAM such as LABs, embedded RAM blocks, and DSP blocks in the FPGA core image. It cannot contain any periphery components such as GPIOs, transceivers, PLL, I/O blocks, the Hard IP for PCI Express IP Core, or other components included in the periphery image.
- Static logic—This region cannot be modified.

When you initiate a core image update, the `CvP_CONFDONE` pin is pulled low, indicating a core image update has started. The FPGA fabric is reinitialized and reconfigured with the new core image. During the core image update through a PCIe link, the `nCONFIG` and `nSTATUS` pins of the FPGA remain at logic high. When the core image update completes, the `CvP_CONFDONE` pin is released high, indicating the FPGA has entered user mode.

Related Information

- [Understanding the Design Steps for CvP Initialization Mode with the Revision Design Flow](#) on page 5-9
Provides more information about the CvP revision design flow.
- [Setting up CvP Parameters in Device and Pin Options GUI for CvP Initialization Mode](#) on page 5-5
- [Setting up CvP Parameters in Device and Pin Options GUI for CvP Update Mode for CvP Revision](#) on page 5-27

CvP Pins

The following table lists the CvP pin descriptions and connection guidelines.

Pin Name	Pin Type	Pin Description	Pin Connection
CvP_CONF_DONE	Output	<p>Driven low during core image configuration and released or driven high after the completion of the core image configuration.</p> <p>During FPGA configuration in CvP initialization mode, you must observe this pin after the CONF_DONE pin goes high to determine if the FPGA is successfully configured.</p> <p>If you are not using the CvP modes, you can use this pin as a user I/O pin.</p>	<p>If this pin is set as dedicated output, the V_{CCPGM} power supply must meet the input voltage specification of the receiving side.</p> <p>If this pin is set as an open-drain output, connect the pin to an external 10-kΩ pull-up resistor to the V_{CCPGM} power supply or a different pull-up voltage that meets the input voltage specification of the receiving side. This gives an advantage on the voltage leveling.</p>

Pin Name	Pin Type	Pin Description	Pin Connection
nPERSTL0 ⁽²⁾	Input	<p>This pin is connected to the Hard IP for PCI Express IP Core as a dedicated fundamental reset pin for PCIe usage. If the signal is low, the transceivers and dedicated PCIe hard IP block that you use for CvP operation are released from the reset mode.</p> <p>If the Hard IP for PCI Express IP Core is not in use, you can use this pin as a user I/O pin.</p>	<p>Connect the nPERSTL0/nPERSTL1 to the PERST# pin of the PCIe slot.</p> <p>This pin may be driven by 3.3V regardless of the VCCIO voltage level of the bank without a level translator as long as the input signal meets the LVTTTL VIH/VIL specification, and as long as it meets the overshoot specifications for 100% operation as defined in Table 1-2 in the "DC and Switching Characteristics for Stratix V Devices." chapter of the Stratix V handbook.</p> <p>Only one nPERST pin is used per PCIe HIP. The Stratix V components always have all four pins listed, even when the specific component might have 1 or 2 PCIe Hard IPs.</p> <ul style="list-style-type: none"> • nPERSTL0 = Bottom Left PCIe HIP & CvP • nPERSTL1 = Top Left PCIe Hard IP (when available) • nPERSTR0 = Bottom Right PCIe Hard IP (when available) • nPERSTR1 = Top Right PCIe Hard IP (When available) <p>For maximum compatibility we recommend to use the bottom left PCIe Hard IP first, as this is the only location that supports CvP.</p>
nPERSTL1 ⁽³⁾			

Related Information

- [Pin Connection Guidelines](#)
Provides more information about related configuration pins. Refer to the respective device family pin connection guidelines.
- [PCI Express Base Specification](#)
Provides more information about PCIe link signals.
- [Arria V Hard IP for PCI Express User Guide](#)
- [Arria V GZ Hard IP for PCI Express User Guide](#)

⁽²⁾ This pin is used in Arria V and Stratix V device families.

⁽³⁾ This pin is used in Cyclone V device family.

- [Cyclone V Hard IP for PCI Express User Guide](#)
- [Stratix V Hard IP for PCI Express User Guide](#)
- [Arria V Device Datasheet](#)
Provides more information about the overshoot specification.
- [Cyclone V Device Datasheet](#)
Provides more information about the overshoot specification.
- [Stratix V Device Datasheet](#)
Provides more information about the overshoot specification.

2013.11.04

UG-01101

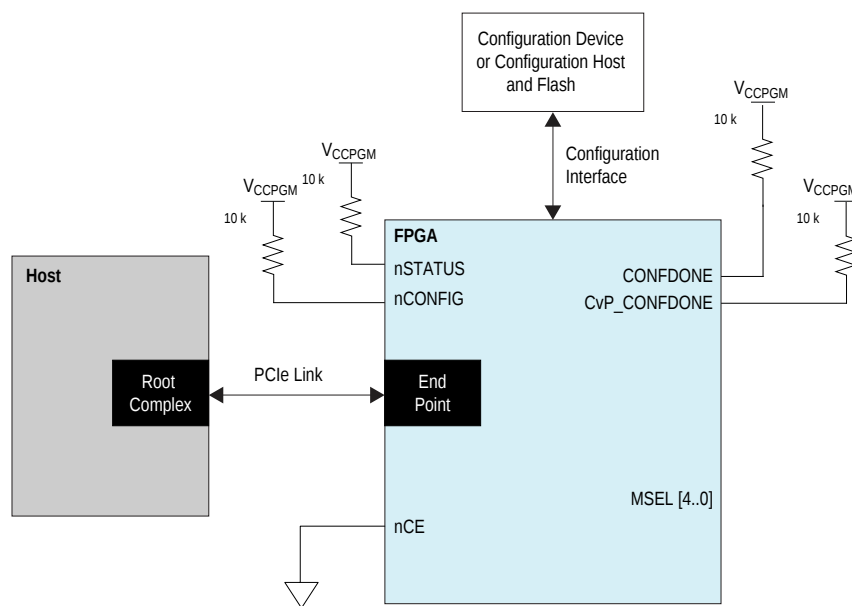
 [Subscribe](#)  [Send Feedback](#)

CvP supports several types of topologies that allow you to configure single or multiple FPGAs.

Single Endpoint

Use the single Endpoint topology to configure a single FPGA. In this topology, the PCIe link connects one PCIe Endpoint in the FPGA device to one PCIe Root Port in the host.

Figure 3-1: Single Endpoint Topology



Multiple Endpoints

Use the multiple Endpoints topology to configure multiple FPGAs through a PCIe switch. This topology provides you with the flexibility to select the device to configure or update through the PCIe link. You can connect any number of FPGAs to the host in this topology.

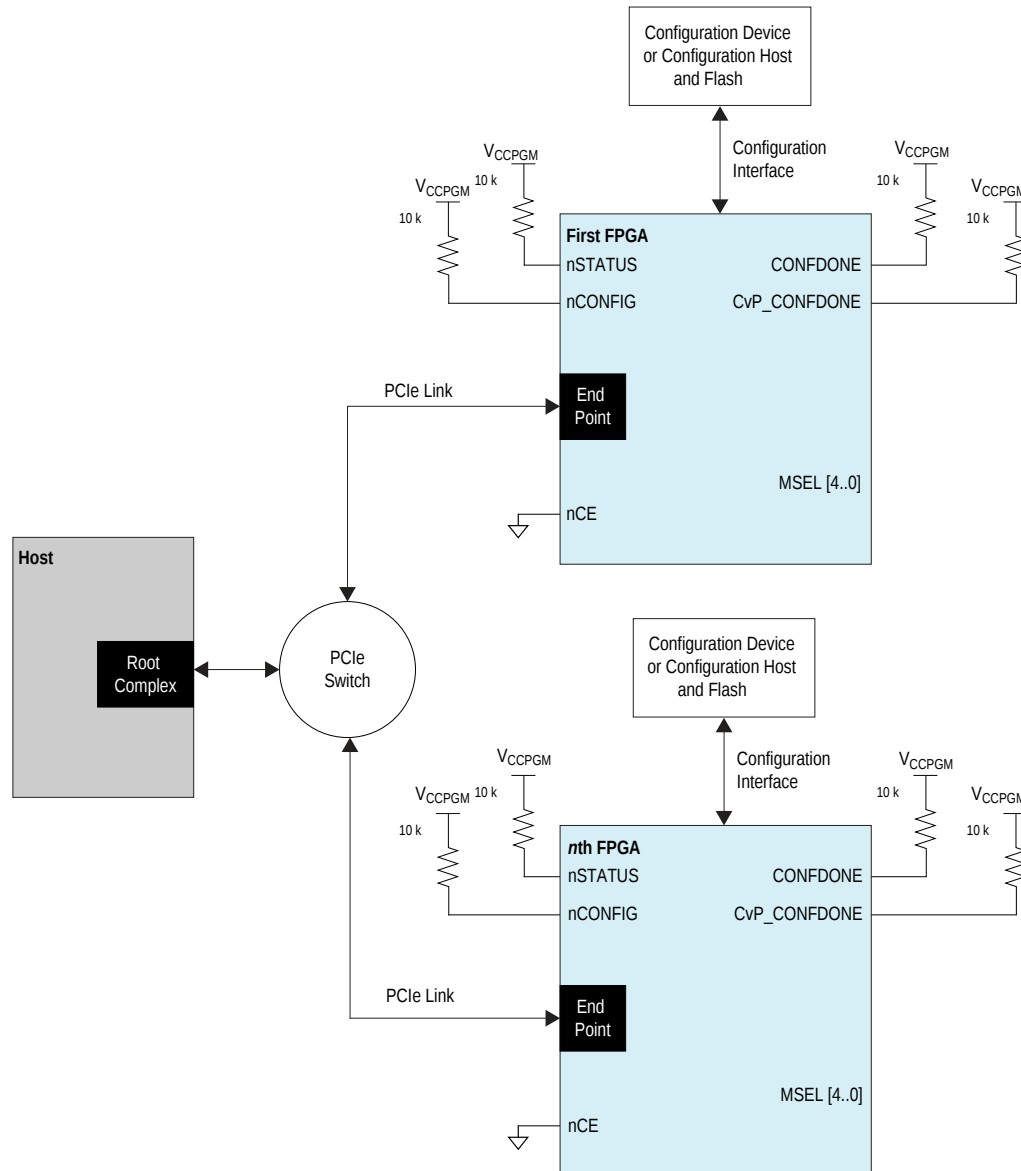
© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO
9001:2008
Registered



The PCIe switch controls the core image configuration through the PCIe link to the targeted PCIe Endpoint in the FPGA. You must ensure that the Root Port can respond to the PCIe switch and direct the configuration transaction to the designated Endpoint based on the media access control (MAC) address of the Endpoint specified by the PCIe switch.

Figure 3-2: Multiple Endpoints Topology



Mixed Chain

Use the mixed chain topology to configure multiple FPGAs that are connected in a chain using both the PCIe link and conventional configuration scheme. In this topology, the PCIe link connects the Endpoint of the master FPGA (the first FPGA in the chain) to the PCIe Root Port in the host. The slave FPGAs are connected in the chain using the PS or FPP configuration scheme. The configuration device, which you use

to store the periphery image in the CvP initialization mode and the full configuration image in the CvP update mode, is only connected to the master FPGA. The master FPGA is configured first, followed by the slave FPGAs.

You must design a user IP for the master FPGA to fetch the configuration data from the Root Port to the slave FPGAs in the chain. The data is latched out from the master device through the GPIOs and latched into the slave devices through the PS or FPP configuration pins—DCLK, DATA line, or DATA bus.

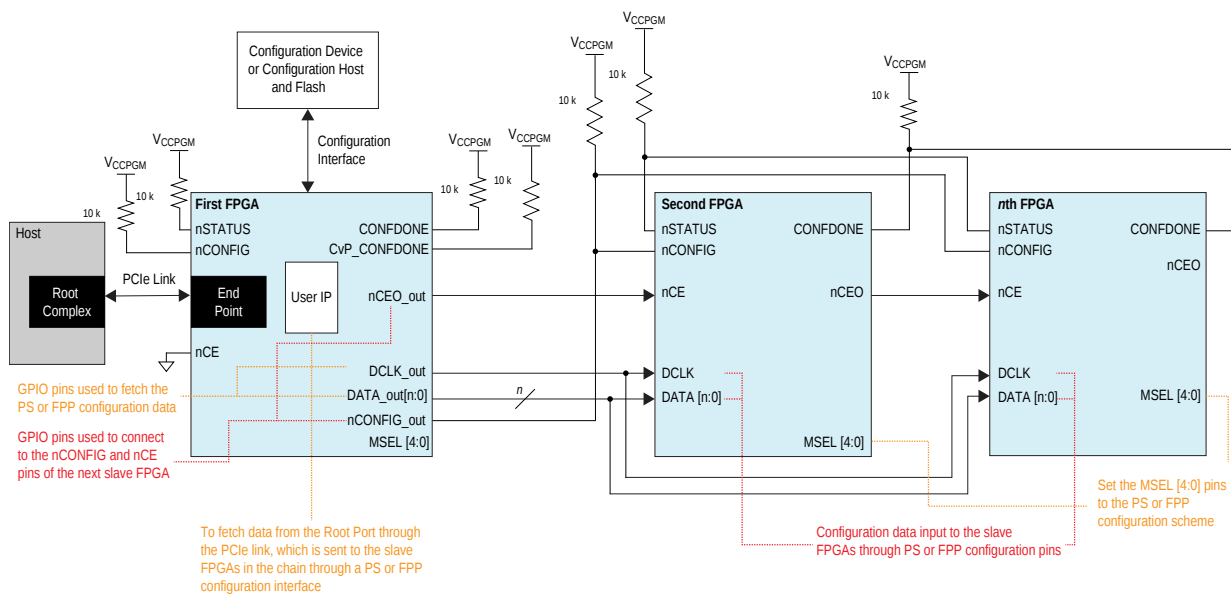
By tying DCLK, nCONFIG, nSTATUS, CONF_DONE pins, and DATA bus of the slave devices together, the slave devices enter user mode at the same time. If any device in the chain detects an error, the slave device chain reinitializes and reconfigures by pulling its nSTATUS pin low. You must ensure there is a suitable line buffering on the DCLK and DATA bus if you are configuring more than four slave devices in the chain.

Configuring Slave FPGAs with Different Configuration Files

To configure the slave FPGAs with different configuration files, connect the nCEO pin of one slave FPGA to the nCE pin of the next slave FPGA in the chain. When the first slave FPGA completes configuration, the slave FPGA pulls the nCEO pin low to enable configuration for the next slave FPGA. This process continues until the last slave FPGA in the chain is configured. You can leave the nCEO pin of the last device unconnected or use the pin as a user I/O pin.

Figure 3-3: Mixed Chain Topology with Different Configuration Files

The following figure shows the connections required to configure slave FPGAs with different configuration files using the FPP scheme.

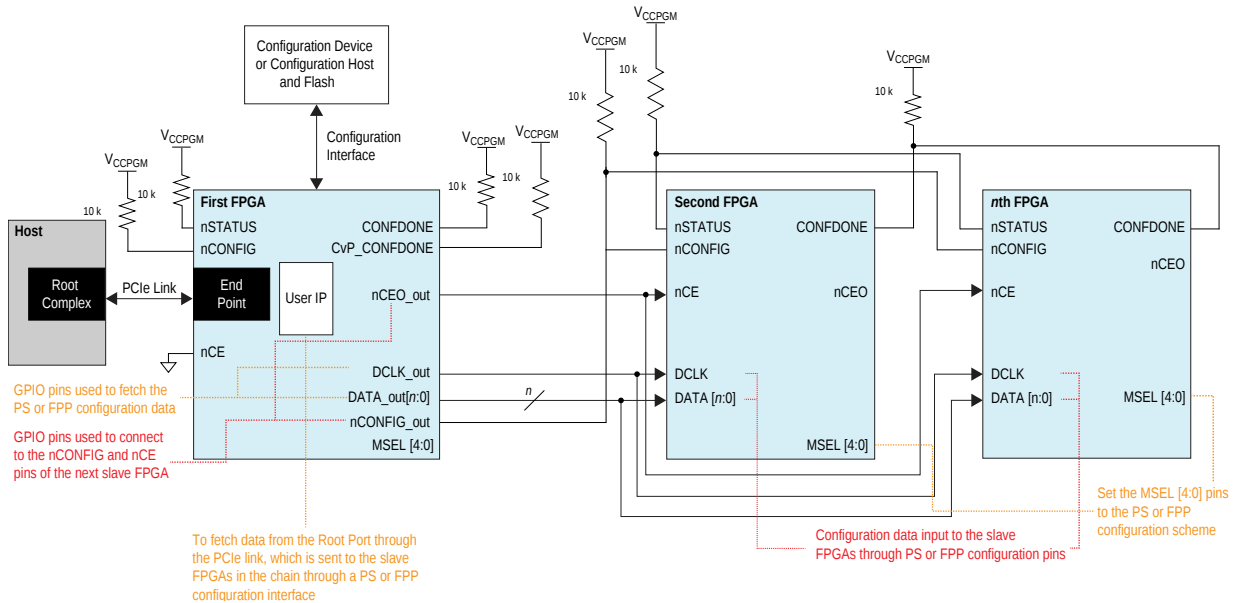


Configuring Slave FPGAs with a Single Configuration File

To configure slave FPGAs with the same configuration file, connect the nCEO pin of the master FPGA to the nCE pins of all slave FPGAs in the chain. In this topology, all the slave devices are configured at the same time. You can leave the nCEO pin of the slave FPGAs in the chain unconnected or use it as a GPIO pin.

Figure 3-4: Mixed Chain Topology with Single Configuration File

The following figure shows the connections required to configure slave FPGAs with the same configuration file using the FPP scheme.



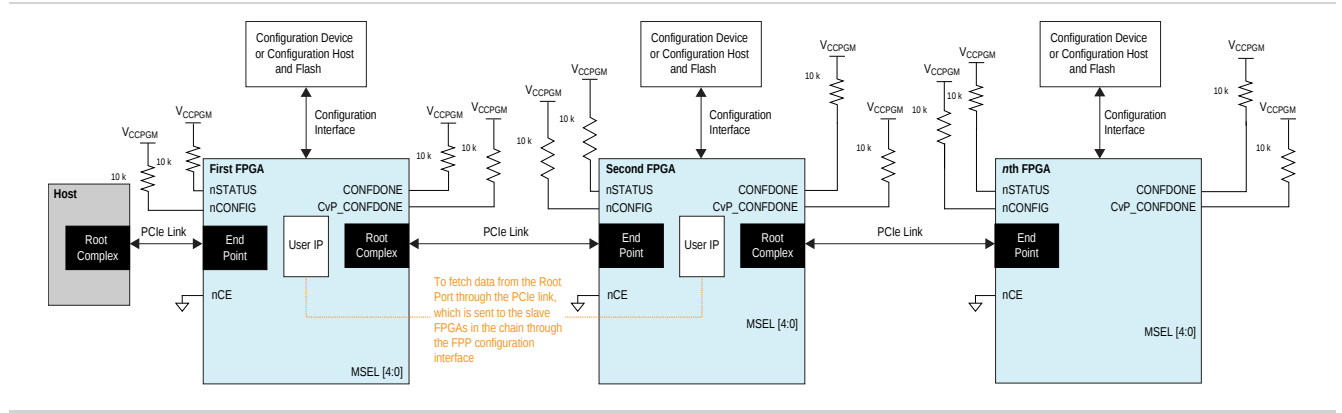
Daisy Chain

Use the daisy chain topology to configure multiple FPGAs that are connected in a PCIe chain. Each FPGA in the daisy chain has an Endpoint and a Root Port. Connect the Root Port in the host to the Endpoint of the first FPGA in the chain; the Root Port of the first FPGA to the Endpoint of the second FPGA, and so forth.

Each FPGA in this chain is connected to a configuration device, which you use to store the periphery image in the CvP initialization mode and the full configuration image in the CvP update mode. In this topology, the nCE, nSTATUS, nCONFIG, CONF_DONE, and CvP_CONF_DONE pins are not tied together, allowing each FPGA to receive the configuration data from their respective configuration devices.

You must design a user IP as part of the core image. The user IP routes the configuration data from the host to the FPGA Root Port applications. You initialize or update the core image of the first FPGA in the chain through the PCIe link between the Root Port in the host and the Endpoint of the FPGA device. When the initialization or update completes, the first FPGA enters user mode. The user IP in the first FPGA then initiates the core image initialization or update of the subsequent device. The process continues until the core image of the last FPGA in the chain is initialized or updated.

Figure 3-5: Daisy Chain Topology



2013.11.04

UG-01101

 [Subscribe](#)
 [Send Feedback](#)

Data Compression

You can choose to compress the core image by turning on the **Generate compressed bitstream** option in the **Configuration** page of the **Device and Pin Options** dialog box in the Quartus II software. The periphery image cannot be compressed. Compressing the core image reduces the storage requirement.

If you configure the FPGA using a compressed core image, you must use a compressed image when updating the core image of the FPGA.

Data Encryption

You can choose to encrypt the core image. The periphery image cannot be encrypted. To configure the FPGA with an encrypted core image, you must pre-program the FPGA with a security key. This key is then used to decrypt the incoming configuration bitstream.

A key-programmed FPGA can accept both encrypted and unencrypted bitstreams if you configure the FPGA using the AS, PS, or FPP scheme. However, if you use CvP, a key-programmed FPGA can only accept encrypted bitstreams. Use the same key to encrypt all revisions of the core image.

Table 4-1: Supported Clock Source for Encrypted Configuration Data

The following table lists the supported clock source for each conventional scheme used in a CvP system.

Key Types	Active Serial		Passive Serial	Fast Passive Parallel
	External Clock	Internal Clock	External Clock	External Clock
Volatile key	Yes	Yes	Yes	Yes
Non-volatile key	No	12.5 MHz	Yes	Yes

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO
9001:2008
Registered



CvP Features

The following features are supported in CvP Initialization and in CvP Update mode:

- Compression / Decompression
- Encryption / Decryption
- Partial Reconfiguration
- Remote System Upgrade

Note: Contact Altera support to use the Remote System Upgrade feature.

Core Image Update

If you are not using CvP, you can update the FPGA image using the remote system upgrade feature that is supported in conventional configuration schemes.

Partial Reconfiguration

You can design your system for CvP and partial reconfiguration. Partial reconfiguration is an advanced feature. If you are interested in using partial reconfiguration, contact Altera for support.

Designing CvP for an Open System

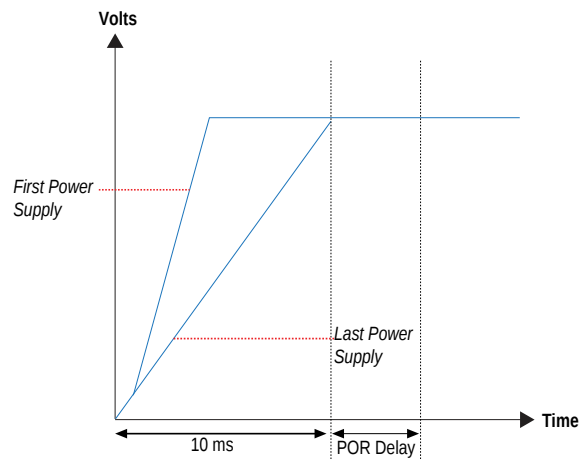
When designing a CvP system for an Open System, ensure that you observe the guidelines provided in this section.

FPGA Power Supplies Ramp Time Requirement

For an open system, you must ensure that your design adheres to the FPGA power supplies ramp-up time requirement.

The power-on reset (POR) circuitry keeps the FPGA in the reset state until the power supply outputs are within the recommended operating range. A POR event occurs when you power up the FPGA until the power supplies reach the recommended operating range within the maximum power supply ramp time, t_{RAMP} .

For CvP, the total t_{RAMP} must be less than 10 ms, from the first power supply ramp-up to the last power supply ramp-up. You must select fast POR by setting the PORSEL pin to high. The fast POR delay time is in the range of 4–12 ms, allowing sufficient time after POR for the PCIe link to start initialization and configuration.

Figure 4-1: Power Supplies Ramp-Up Time and POR**Related Information**

- [Power Management in Arria V Devices](#)
- [Power Management in Cyclone V Devices](#)
- [Power Management in Stratix V Devices](#)

PCIe Wake-Up Time Requirement

For an open system, you must ensure that the PCIe link meets the PCIe wake-up time requirement as defined in the *PCI Express CARD Electromechanical Specification*. The transition from power-on to the link active (L0) state for the PCIe wake-up timing specification must be within 200 ms. The timing from FPGA power-up until the Hard IP for PCI Express IP Core in the FPGA is ready for link training must be within 120 ms.

Related Information

- [PCI Express Card Electomechanical 3.0 Specification](#)

PCIe Wake-Up Time Requirement for CvP Initialization Mode

For CvP initialization mode, the Hard IP for PCI Express IP Core is guaranteed to meet the 120 ms requirement because the periphery image configuration time is significantly less than the full FPGA configuration time. Therefore, you can choose any of the conventional configuration schemes for the periphery image configuration.

To ensure successful configuration, all POR-monitored power supplies must ramp up monotonically to the operating range within the 10 ms ramp-up time. $\overline{\text{PERST\#}}$ reset input signal indicates whether the power supplies of the FPGA are within their specified voltage tolerances and are stable. The embedded hard reset controller triggers after the internal status signal, indicates that periphery image is loaded. This reset does not trigger off of $\overline{\text{PERST\#}}$. For CvP initialization mode, the PCIe link supports the FPGA core image configuration, FPGA core image update, and PCIe applications in user mode.

Note: For Gen 2 capable Endpoints, after loading the core .sof, Altera recommends that software verify that the link has trained to the expected Gen 2 rate. If the link is not operating at Gen 2, software can trigger the Endpoint to retrain.

Figure 4-2: PCIe Timing Sequence in CvP Initialization Mode

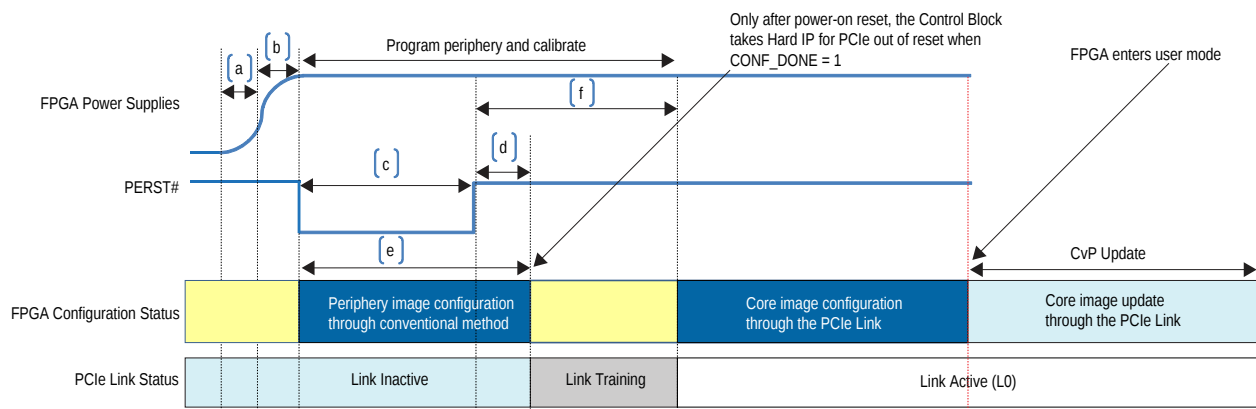


Table 4-3: Power-Up Sequence Timing in CvP Initialization Mode

Timing Sequence	Timing Range (ms)	Description
a	10	Maximum ramp-up time requirement for all POR-monitored power supplies in the FPGA to reach their respective operating range.
b	4–12	FPGA POR delay time.
c	100	Minimum PERST# signal active time from the host.
d	20	Minimum PERST# signal inactive time from the host before the PCIe link enters training state.
e	120	Maximum time from the FPGA power up to the end of periphery configuration in CvP initialization mode.
f	100	Maximum time PCIe device must enter L0 after PERST# is deasserted.

PCIe Wake-Up Time Requirement for CvP Update Mode

For CvP update mode, you initialize the FPGA by configuring it using one of the conventional configuration schemes upon device power-up. An open system requires that the FPGA initialization completes within 120 ms. To ensure that this requirement is met, choose the right conventional configuration scheme for your system.

To ensure successful configuration, all POR-monitored power supplies must ramp up monotonically to the operating range within the 10 ms ramp-up time. PERST# is one of the auxiliary signals specified in the PCIe electromechanical specification. The PERST# signal is sent from the PCIe host to the FPGA. The PERST# signal indicates whether the power supplies of the FPGA are within their specified voltage tolerances and are stable. The PERST# signal also initializes the FPGA state machines and other logic after power supplies

are stabilized. The PCIe link supports PCIe applications in user mode for CvP update mode, therefore, you can use the PCIe link for core image update.

Note: For Gen 2 capable Endpoints, after loading the core .sof, Altera recommends that software verify that the link has trained to the expected Gen 2 rate. If the link is not operating at Gen 2, software can trigger the Endpoint to retrain.

Figure 4-3: PCIe Timing Sequence in CvP Update Mode

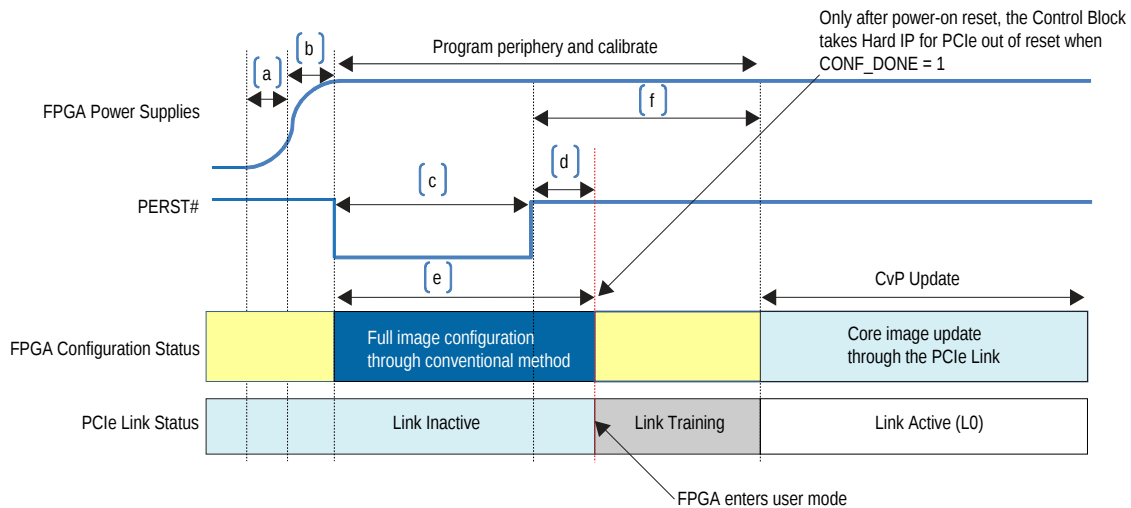


Table 4-4: Power-Up Sequence Timing in CvP Update Mode

Timing Sequence	Timing Range (ms)	Description
a	10	Maximum ramp-up time requirement for all POR-monitored power supplies in the FPGA to reach their respective operating range.
b	4–12	FPGA POR delay time.
c	100	Minimum PERST# signal active time from the host.
d	20	Minimum PERST# signal inactive time from the host before the PCIe link enters training state.
e	120	Maximum time from the FPGA power up to the end of the full FPGA configuration in CvP update mode.
f	100	Maximum time PCIe device must enter L0 after PERST# is deasserted.

Recommended Configuration Schemes

For CvP initialization mode, you can configure the FPGA with the periphery image using the AS, PS, or FPP configuration scheme.

For CvP update mode, you can configure the FPGA fully using one of the configuration schemes listed in the table below. The table lists the configuration schemes based on the fastest DCLK frequency with data compression and encryption features disabled in the CvP update mode. These features require different data

to clock ratios, which prolongs total configuration time. Consequently, total configuration time does not meet the 200-ms PCIe wake-up timing specification.

Table 4-5: Recommended Configuration Schemes for CvP Update Mode

Variant	Member Code	Configuration Scheme
Arria V GX	A1	FPP x8
	A3	FPP x16
	A5	FPP x16
	A7	
	B1	
	B3	
	B5	
	B7	
Arria V GT	C3	FPP x8 FPP x16
	C7	FPP x16
	D3	
	D7	
Arria V GZ	E1	FPP x16
	E3	FPP x32
	E5	FPP x32
	E7	
Arria V SX	B3	FPP x16
	B5	
Arria V ST	D3	FPP x16
	D5	
Cyclone V GX	C3	AS x4 FPP x8 FPP x16
	C4	FPP x8
	C5	FPP x16
	C7	
	C9	FPP x16

Variant	Member Code	Configuration Scheme
Cyclone V GT	D5	FPP x8
	D7	FPP x16
	D9	FPP x16
Cyclone V SX	C2	TBD
	C4	
	C5	FPP x8
	C6	FPP x16
Cyclone V ST	D5	FPP x8
	D6	FPP x16
Stratix V GX	A3	FPP x16
	A4	FPP x32
	A5	FPP x32
	A7	
	A9	—
	AB	
	B5	FPP x32
	B6	
	B9	—
	BB	
Stratix V GT	C5	FPP x32
	C7	
Stratix V GS	D3	FPP x8 FPP x16 FPP x32
	D4	FPP x16
	D5	FPP x32
	D6	FPP x32
	D8	

Estimating PCIe Wake-Up Time Requirement

Figure 4-4: Estimating PCIe Wake-Up Time Requirement Equation

$$\left(\frac{\text{Full configuration file size}}{\text{Number of data line}} \times \frac{1}{\text{DCLK frequency}} \right) + (\text{Power ramp up} + \text{POR delay time})$$

Conventions used for the equation:

- Full configuration file size—refer to uncompressed **.rbf** sizes.
- Number of data line—refer to the width of data bus. For example, the width of data bus for FPP x16 is 16.
- DCLK frequency—refer to f_{MAX} for the DCLK frequency.
- Power ramp up—must be within 10 ms.
- POR delay—use fast POR, maximum time is 12 ms.

You can use the equation above to make estimation if your device could meet the PCIe wake-up time requirement. The following figure shows the example of calculating the PCIe wake-up time requirement for Arria V GX A5 device.

Figure 4-5: Example of Calculating PCIe Wake-Up Time Requirement

$$\begin{aligned} & \left(\frac{101,740,640}{16} \times \frac{1}{125,000,000} \right) + (10 + 12) \\ & = 50 \text{ ms} + 22 \text{ ms} \\ & = 72 \text{ ms} \end{aligned}$$

The estimation for Arria V GX A5 device is 72 ms, which is able to meet the PCIe wake-up time requirement of 120 ms.

Related Information

- [Arria V Device Datasheet](#)
Provides more information about uncompressed **.rbf** sizes for entire FPGA and IOCSR, DCLK frequency, and POR delay.
- [Cyclone V Device Datasheet](#)
Provides more information about uncompressed **.rbf** sizes for entire FPGA and IOCSR, DCLK frequency, and POR delay.
- [Stratix V Device Datasheet](#)
Provides more information about uncompressed **.rbf** sizes for entire FPGA and IOCSR, DCLK frequency, and POR delay.

Design CvP for a Closed System

When designing CvP for a closed system, estimate the periphery configuration time for CvP initialization mode or full FPGA configuration time for CvP update mode. You must ensure that the estimated configuration time is within the time allowed by the PCIe host.

Clock Connections for CvP Designs Including the Transceiver Reconfiguration Controller

If your design includes the following components:

- An Arria V, Cyclone V, or Stratix V device with CvP enabled
- Any additional transceiver PHY connected to the same Transceiver Reconfiguration Controller

then you must connect the PLL reference clock which is called `refclk` to the `mgmt_clk_clk` signal of the Transceiver Reconfiguration Controller and the additional transceiver PHY. In addition, if your design includes more than one Transceiver Reconfiguration Controllers on the same side of the FPGA, they all must share the `mgmt_clk_clk` signal.

2013.11.04

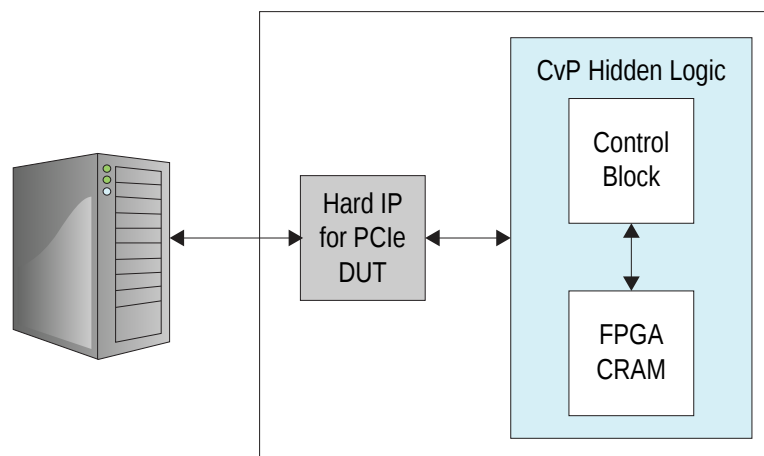
UG-01101

 [Subscribe](#)  [Send Feedback](#)

The example designs in this chapter illustrate the steps required for CvP initialization mode, CvP initialization with subsequent updates of the core logic, and CvP update mode. All start with the PCI Express High Performance Reference Design that you can download from the Altera website. The example designs show how to use the CvP revision design flow to prepare the design for reconfigurable core logic.

The CvP process involves the interactions between the PCI Express host, the FPGA Control Block, the Stratix V Hard IP for PCI Express IP Core, and the CRAM in FPGA as indicated in the following figure. The Control Block and FPGA CRAM are hidden. You cannot access them. Consequently, you cannot simulate the CvP functionality.

Figure 5-1: Key Components in a CvP Design



The following table describes some of the key files included in the example design.

Table 5-1: Key Files for the CvP Qsys Example Design

Name	Description
altpcied_sv.sdc	Synopsys Design Constraints (.sdc) for the Hard IP for PCI Express IP Core.
top_hw.sdc	Top-level timing constraint file .sdc for the complete design.
top_hw.v	Top-level wrapper for the PCI Express High Performance Reference Design.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO
9001:2008
Registered



Name	Description
top.cof	CvP conversion file for CvP initialization mode. This file specifies the input and output files that Quartus II software requires to split the original .sof or .pof file into periphery and core images.
pcie_lib	Design files that are used by synthesis tools.

Understanding the Design Steps for CvP Initialization Mode

CvP initialization mode divides the design into periphery and core images. The periphery image is stored in a flash device on the PCB. You program the periphery via JTAG. The core image is stored in host memory. You download the core image to the FPGA using the PCI Express link.

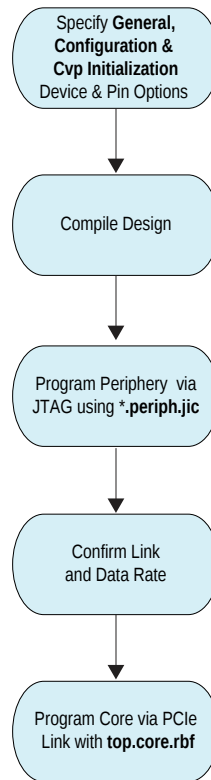
Note: If you plan to create multiple versions of the core logic for the same periphery I/O, the new core images might not work with the previous periphery image. You can use the [Understanding the Design Steps for CvP Initialization Mode with the Revision Design Flow](#) on page 5-9 to create reconfigurable images that connect to the same periphery image.

You specify CvP initialization mode in the Quartus II software by selecting the CvP Settings **Power up and subsequent core configuration**. You might choose CvP initialization mode for any of the following reasons:

- To satisfy the PCIe initial power up requirement for plug-in cards if FPGA programming time exceeds this limit
- To save cost by storing the core image in external host memory
- To prevent unauthorized access to the core image by using encryption

The following figure provides the high-level steps for CvP initialization mode.

Figure 5-2: Design Flow for CvP Initialization Mode



Note: When you select CvP initialization mode, you must use the CMU PLL and the hard reset controller for the PCI Express Hard IP.

In the walkthrough, CvP initialization mode includes the following steps:

1. [Downloading and Generating the High Performance Reference Design](#) on page 5-3
2. [Setting up CvP Parameters in Device and Pin Options GUI for CvP Initialization Mode](#) on page 5-5
3. [Compiling the Design for the CvP Initialization Mode](#) on page 5-7
4. [Splitting the SOF File for the CvP Initialization Design Mode](#) on page 5-7
5. [Bringing Up the Hardware](#) on page 5-37

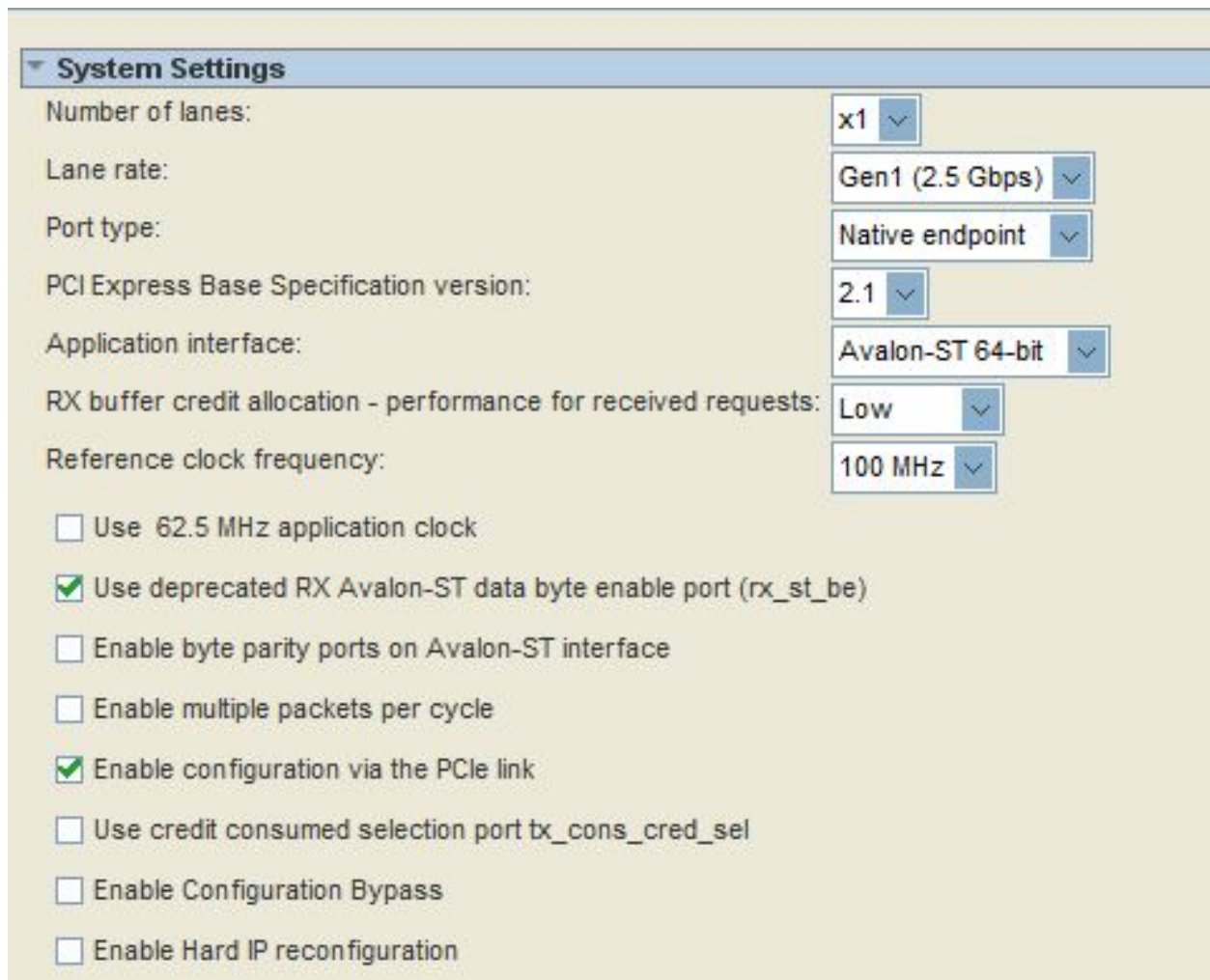
Downloading and Generating the High Performance Reference Design

Follow these steps to regenerate the PCI Express High Performance Reference Design with CvP enabled:

1. Download the **PCIe_hiperf_s5gx.zip** file from the PCI Express Avalon-ST High Performance Reference Design web page. This design includes the correct pin assignments and project settings to target the Stratix V GX FPGA Development Kit.
2. Unzip **PCIe_hiperf_s5gx.zip**.
3. Copy **hip_s5gx_x1_g1_ast64_5SGXEA7K2F40.qar** to your working directory.
4. Start the Quartus II software and restore **hip_s5gx_x1_g1_ast64_5SGXEA7K2F40.qar**.
5. On the Tools menu, select **Qsys**.
6. Open **top.qsys**.
7. On the **System Contents** tab, right-click **DUT** and select **Edit**.

8. Under **System Settings**, turn on **Enable configuration via the PCIe link** as shown in the following figure.

Figure 5-3: Hard IP for PCI Express GUI



System Settings

Number of lanes: x1

Lane rate: Gen1 (2.5 Gbps)

Port type: Native endpoint

PCI Express Base Specification version: 2.1

Application interface: Avalon-ST 64-bit

RX buffer credit allocation - performance for received requests: Low

Reference clock frequency: 100 MHz

Use 62.5 MHz application clock

Use deprecated RX Avalon-ST data byte enable port (rx_st_be)

Enable byte parity ports on Avalon-ST interface

Enable multiple packets per cycle

Enable configuration via the PCIe link

Use credit consumed selection port tx_cons_cred_sel

Enable Configuration Bypass

Enable Hard IP reconfiguration

9. Click **Finish**.

10. On the **Generation** tab, specify the settings in the following table. Then click **Generate** at the bottom of the window.

Table 5-2: Qsys Generation Tab Settings

Parameter	Value
Create simulation model	None
Create testbench Qsys system	None
Create testbench simulation model	None
Create HDL design files for synthesis	Verilog

Parameter	Value
Create block symbol file (.bsf)	Leave this entry off
Path	< working_dir > top
Simulation	Leave this entry blank
Testbench	<working_dir> /top /synthesis

11. After successful compilation, close Qsys.

Related Information

[PCI Express Avalon-ST High Performance Reference Design](#)

Setting up CvP Parameters in Device and Pin Options GUI for CvP Initialization Mode

Follow these steps to specify CvP parameters using the Quartus II software:

1. On the Quartus II Assignments menu, select **Device**, and then click **Device and Pin Options**
2. Under **Category** select **General**, and then enable following options:
 - a. **Auto-restart configuration after error.** If this option is enabled, CvP restarts after an error is detected.
 - b. **Enable autonomous PCIe HIP mode.**

Checking this box has no affect if you have enabled CvP by turning on **Enable Configuration via the PCIe link** in the Hard IP for PCI Express GUI. The Quartus II software automatically enables autonomous mode by default. In autonomous mode, the control block takes the Hard IP for PCI Express out of reset after periphery image is loaded. The Hard IP for PCI Express responds to configuration requests and memory requests with the normal successful status. The core image is loaded using PCIe link in both CvP initialization and CvP update mode.

The **Enable autonomous PCIe HIP mode** option only has effect if your design has the following two characteristics:

- You are using something other than the PCIe link to load the core image, for example a flash device or Ethernet controller.
- You have not checked **Enable Configuration via the PCIe link** in the Hard IP for PCI Express GUI.

If both of these conditions are true, the following two options are available:

- If you checked **Enable autonomous PCIe HIP mode**, the control block takes the Hard IP for PCI Express out of reset after the periphery image is loaded. The Hard IP responds to configuration requests with Configuration Retry Status (CRS) and memory requests with UR status until the FPGA enters user mode.
- If you did not check **Enable autonomous PCIe HIP mode**, the Hard IP remains in reset until FPGA enters user mode. Link training only occurs after the FPGA enters user mode.

Note:

This parameter only controls functionality for the initial configuration. It allows open PCI Express systems to meet the configuration time requirement defined in the *PCI Express Base Specification*. After the initial configuration, it has no significance because the core image has already been configured.

- c. Leave all other options disabled.
3. Under **Category**, select **Configuration** to specify the configuration scheme and device. Specify the settings in the following table:

Table 5-3: CvP Initialization Mode Configuration Settings

Parameter	Value
Configuration scheme	Active Serial x4.
Configuration mode	Standard .
Configuration device	EPCQ256.
Configuration device I/O voltage	Auto.
Force VCCIO to be compatible with configuration I/O voltage	Leave this option off.
Generate compressed bitstreams	Turn this option off. Because this is a small example design, it does not use a compressed bitstream. For larger designs, using a compressed bitstream significantly reduces configuration time. In addition, a compressed bitstream requires a smaller flash device.
Active serial clock source	100 MHz Internal Oscillator.
Enable input tri-state on active configuration pins in user mode	Leave this option off.

Under **Category** select **CvP Settings**. For **CvP Initialization** mode, specify the following settings in the following table:

Table 5-4: CvP Initialization Category Settings

Parameter	Value
CvP Initialization	Power up and subsequent core configuration
Enable CvP_CONFDONE pin	Turn this option on
Enable open drain on CvP_CONFDONE pin	Turn this option on

These **Configuration** settings use the configuration devices available on the Stratix V GX FPGA Development Board. The **EPCQ256** flash device is far larger than required to load a periphery image.

4. Click **OK** to close the **Device and Pin Options** dialog box.
5. Click **OK** to close the **Device** dialog box.
6. Save your project.

Compiling the Design for the CvP Initialization Mode

1. To compile the design, on the Processing menu, select **Start compilation**. Compilation creates a **.sof** file in the **pcie_quartus_files** subdirectory.

Splitting the SOF File for the CvP Initialization Design Mode

Follow these steps to split your **.sof** file into separate images for the periphery and core logic.

1. On the File menu, select **Convert Programming File**.
2. Under **Output programming files to convert**, specify the options in the following table.

Table 5-5: CvP Initialization Output Programming Files Settings

Parameter	Value
Programming file type	JTAG Indirect Configuration File (. jic).
Configuration device	EPCQ256.
Mode	Active Serial x4.
File name	Browse to and select the ./pcie_quartus_files/ directory. Type the file name top. jic . Then click Save .
Create Memory Map File	Turn this option on.
Create CvP files	Turn this option on. This box is greyed out until you specify the SOF Data file under Input files to convert .

3. Under **Input files to convert**, specify the options in the following table:

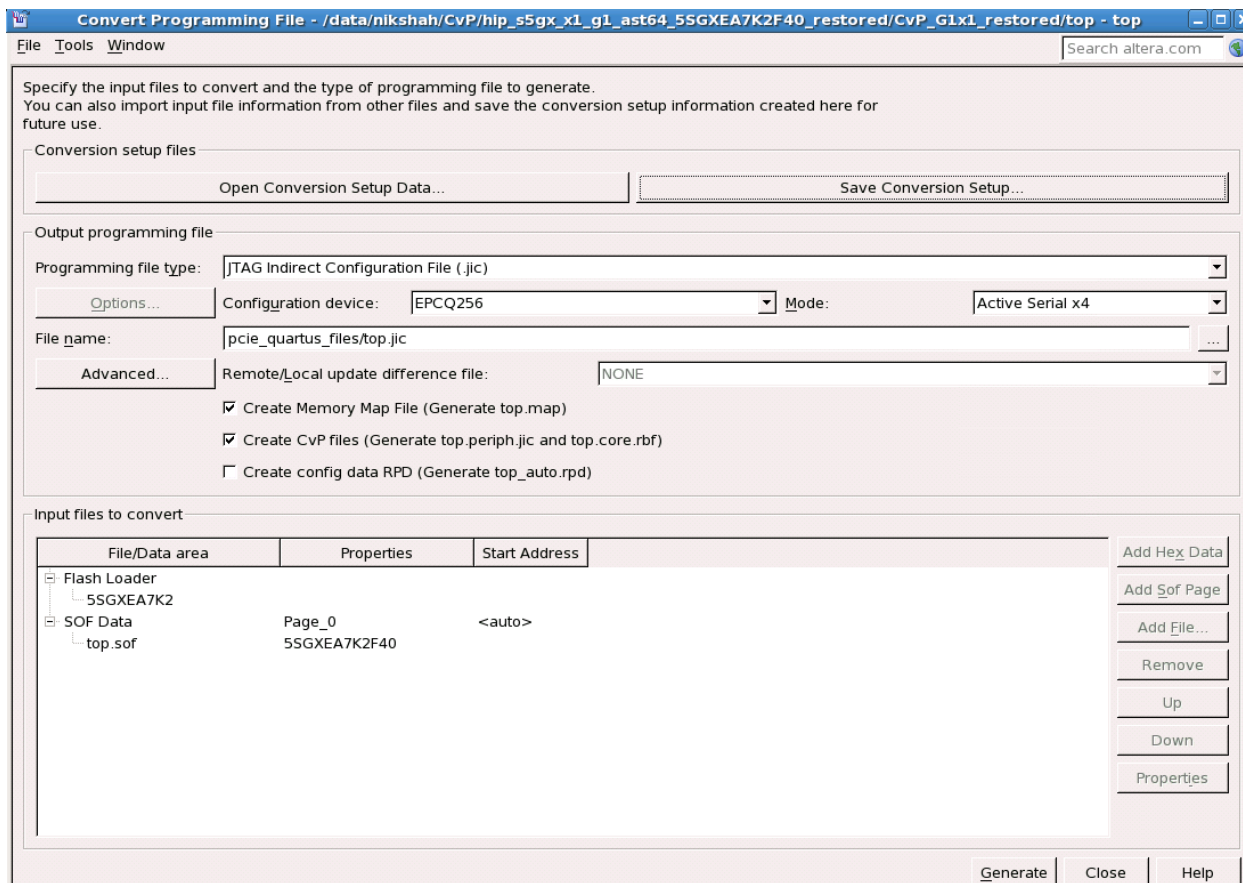
Table 5-6: CvP Initialization Input Files to Convert Settings

Parameter	Value
Click Flash Loader	Click Add Device and select Stratix V and then 5SGXEA7K2 , and click OK .
Click SOF Data	Click Add File and navigate to ./pcie_quartus_files/ top.sof . If you specified a compressed or encrypted bitstream in the Device and Pin Options dialog box, you must specify the same options for Conversion Programming File window. To enable these settings, click top.sof . Then click Properties and check the appropriate boxes.

Parameter	Value
Mode	Active Serial x4.

The following figure illustrates the options that you specified.

Figure 5-4: CvP Initialization Mode: Convert Programming File Settings



- Now turn on the **Create CvP files (Generate top.periph.jic and top.core.rbf)** parameter in the **Output Programming Files** section.

Note: If you do not check this box, the Quartus II software does not create separate files for the periphery and core images.

5. Click **Save Conversion Setup** to save these settings. For this exercise, call your settings **cvp_base.cof**. The Quartus II software does not automatically save your choices.
6. Click **Generate** to create **top.periph.jic** and **top.core.rbf**.

Understanding the Design Steps for CvP Initialization Mode with the Revision Design Flow

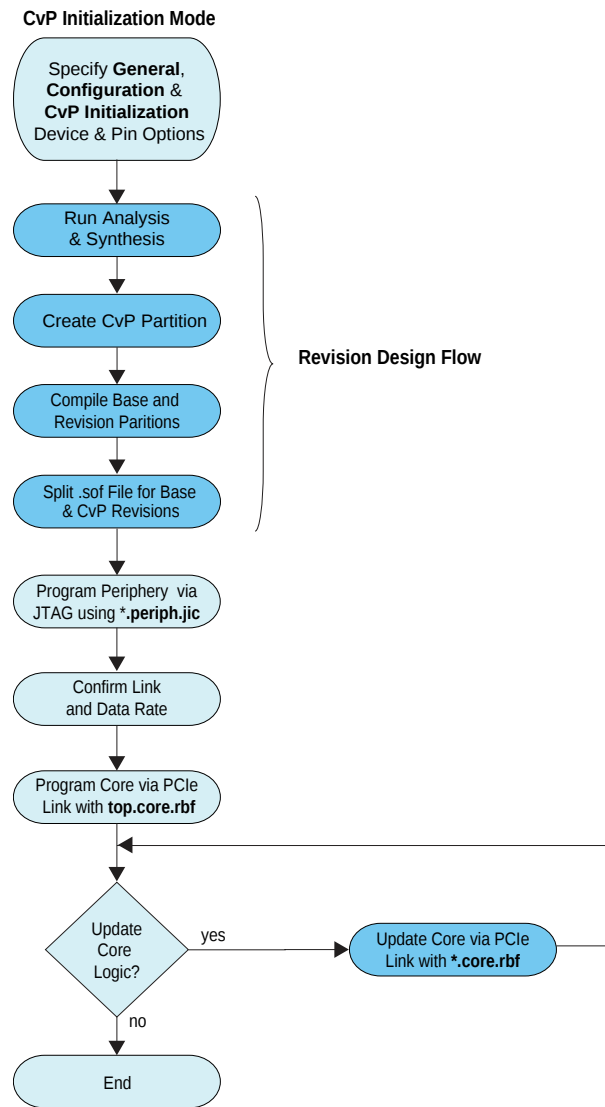
The CvP initialization mode with the revision design flow allows you to create a reconfigurable core image that work with the single periphery image. The core image is stored in host memory. You download the core image to the FPGA using the PCI Express link. By using the Revision Design Flow, you can update the core image after the initial download to run alternate versions of the core logic.

You specify this mode in the Quartus II software by selecting the CvP Settings **Power up and subsequent core configuration**. When the FPGA is fully programmed, the FPGA enters user mode. In user mode, you can reprogram the original static core image. The following are typical reasons to choose CvP initialization mode:

- To satisfy the PCIe initial power up requirement for plug-in cards if FPGA programming time exceeds this limit
- To save cost by storing the core image in external host memory
- To prevent unauthorized access to the core image by using encryption
- To update the core logic the following reasons:
 - To customize the core logic for different tasks
 - To provide periodic updates for routine maintenance of the core logic

If you plan to create multiple versions of the core logic for the same periphery I/O, the new core images might not work with the previous periphery image. You can use the **CvP Revision Design Flow** to create reconfigurable images that connect to the same periphery image. The following figure provides the high-level steps for CvP initialization mode with the CvP Revision Design Flow.

Figure 5-5: Design Flow for the CvP Initialization Mode with the Revision Design Flow



Note: When you select CvP initialization mode, you must use the CMU PLL and the hard reset controller for the PCI Express Hard IP.

In the following walkthrough, the CvP initialization mode includes the following steps:

1. [Downloading and Generating the High Performance Reference Design](#) on page 5-3
2. [Workaround for a Known Issue with Transceiver Reconfiguration Controller IP Core](#) on page 5-13
3. [Creating An Alternate user_led.v File for the Reconfigurable Core Region](#) on page 5-13
4. [Setting up CvP Parameters in Device and Pin Options GUI for CvP Initialization Mode](#) on page 5-5
5. [Creating CvP Revisions of the Core Logic Region Using the CvP Revision Design Flow](#) on page 5-15
6. [Compiling Both the Base and cvp_app Revisions in the CvP Revision Design Flow](#) on page 5-18

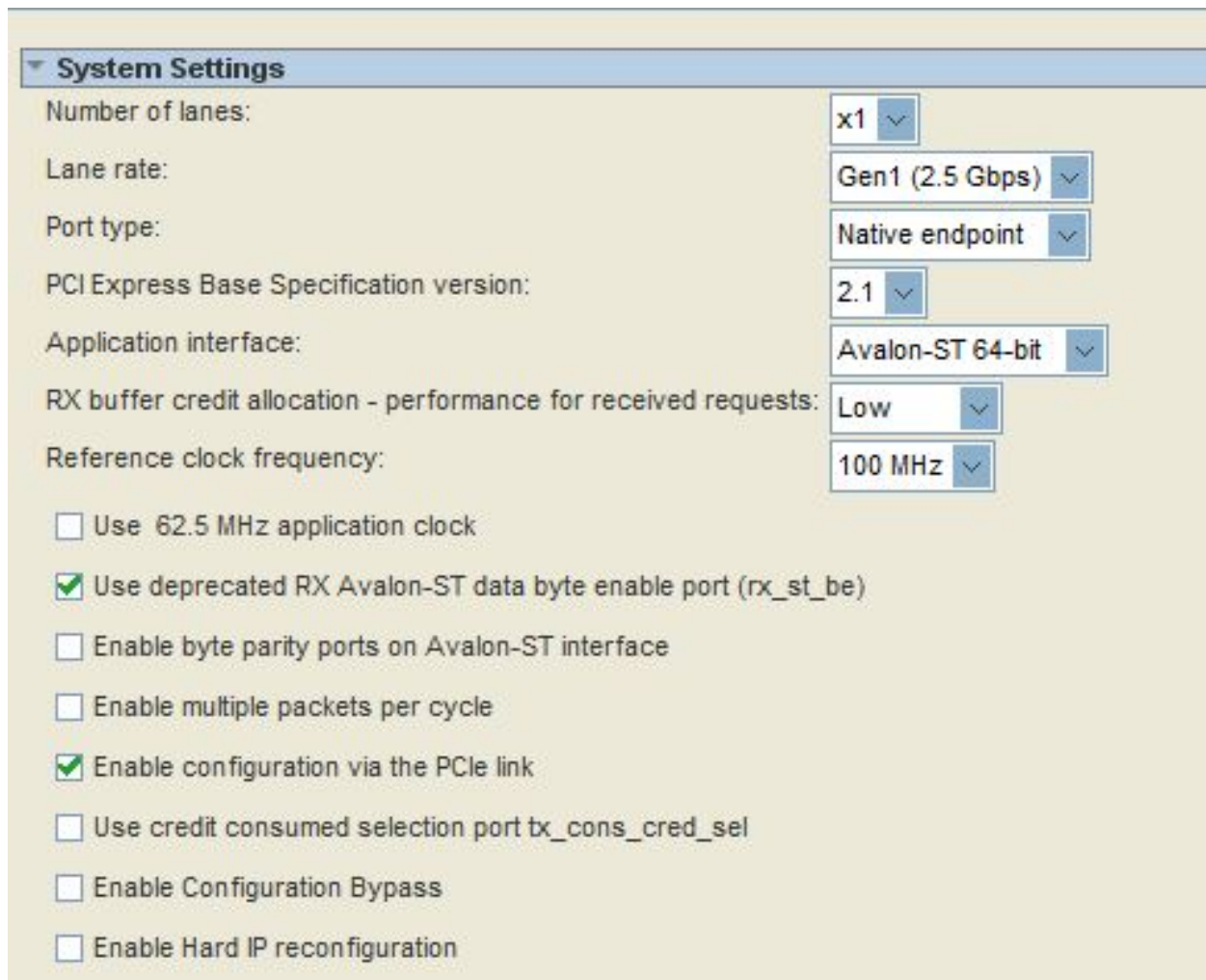
7. [Splitting the SOF File for the CvP Initialization Design Mode](#) on page 5-7
8. [Splitting the SOF File for the CvP Initialization Mode with the CvP Revision Design Flow](#) on page 5-21
9. [Bringing Up the Hardware](#) on page 5-37

Downloading and Generating the High Performance Reference Design

Follow these steps to regenerate the PCI Express High Performance Reference Design with CvP enabled:

1. Download the **PCIe_hiperf_s5gx.zip** file from the PCI Express Avalon-ST High Performance Reference Design web page. This design includes the correct pin assignments and project settings to target the Stratix V GX FPGA Development Kit.
2. Unzip **PCIe_hiperf_s5gx.zip**.
3. Copy **hip_s5gx_x1_g1_ast64_5SGXEA7K2F40.qar** to your working directory.
4. Start the Quartus II software and restore **hip_s5gx_x1_g1_ast64_5SGXEA7K2F40.qar**.
5. On the Tools menu, select **Qsys**.
6. Open **top.qsys**.
7. On the **System Contents** tab, right-click **DUT** and select **Edit**.
8. Under **System Settings**, turn on **Enable configuration via the PCIe link** as shown in the following figure.

Figure 5-6: Hard IP for PCI Express GUI



System Settings

Number of lanes: x1

Lane rate: Gen1 (2.5 Gbps)

Port type: Native endpoint

PCI Express Base Specification version: 2.1

Application interface: Avalon-ST 64-bit

RX buffer credit allocation - performance for received requests: Low

Reference clock frequency: 100 MHz

Use 62.5 MHz application clock

Use deprecated RX Avalon-ST data byte enable port (rx_st_be)

Enable byte parity ports on Avalon-ST interface

Enable multiple packets per cycle

Enable configuration via the PCIe link

Use credit consumed selection port tx_cons_cred_sel

Enable Configuration Bypass

Enable Hard IP reconfiguration

9. Click **Finish**.

10. On the **Generation** tab, specify the settings in the following table. Then click **Generate** at the bottom of the window.

Table 5-7: Qsys Generation Tab Settings

Parameter	Value
Create simulation model	None
Create testbench Qsys system	None
Create testbench simulation model	None
Create HDL design files for synthesis	Verilog
Create block symbol file (.bsf)	Leave this entry off

Parameter	Value
Path	< working_dir > top
Simulation	Leave this entry blank
Testbench	< working_dir > /top /synthesis

11. After successful compilation, close Qsys.

Related Information

[PCI Express Avalon-ST High Performance Reference Design](#)

Workaround for a Known Issue with Transceiver Reconfiguration Controller IP Core

If you plan to perform almost continuous updates of the reconfigurable core logic in stress tests or in your actual system, you may encounter an issue with the Transceiver Reconfiguration Controller. This issue might cause the PCIe link to downtrain in Quartus II 13.0 release and earlier versions of the Quartus II software. If you are using Quartus II 13.0 SP1 or later versions of the Quartus II software, then you may not encounter this issue. Complete the following steps to avoid this issue:

1. Open **pcie_lib/top.v**.
2. Search for the Reconfiguration Controller instance named `alt_xcvr_reconfig` and comment out the entire `reconfig_controller` in **top.v**. (The Transceiver Reconfiguration Controller instance includes 32 lines of Verilog HDL code.)
3. Add the 5 lines of Verilog HDL shown in below after the commented out instance,

```
alt_xcvr_reconfig:
```

```
wire [69:0] reconfig_to_xcvr_bus = {24'h0, 2'b11, 44'h0};
```

```
assign pcie_reconfig_driver_0_reconfig_mgmt_waitrequest = 1'b0;
```

```
assign pcie_reconfig_driver_0_reconfig_mgmt_readdata = 32'h0;
```

```
assign alt_xcvr_reconfig_0_reconfig_busy_reconfig_busy = 1'b0;
```

```
assign alt_xcvr_reconfig_0_reconfig_to_xcvr_reconfig_to_xcvr = { 2 {reconfig_to_xcvr_bus}};
```

In this example, the first statement hardwires the `reconfig_to_xcvr_bus` to the correct values per channel. The first three assignment statements specify the correct values for the `waitrequest`, `readdata`, `reconfig_busy` signals. The final assignment statement for `alt_xcvr_reconfig_0_reconfig_to_xcvr_reconfig_to_xcvr` represents the full reconfiguration bus for all active transceiver channels. This bus is replicated 2 times because 2 channels are active in the Gen1 x1 instance.

Creating An Alternate `user_led.v` File for the Reconfigurable Core Region

This example design creates a new version of the PCI Express High Performance Reference Design. The original version of this reference design includes an LED which turns on whenever the Link Training and Status and State Machine (LTSSM) enters the Polling.Compliance state (0x3). The alternate version of **user_led.v** turns on the LED whenever bit[23] of a counter is one. The LED is instantiated as a separate module in the High Performance Reference Design to demonstrate the steps necessary to create a design with multiple versions of the core logic.

Complete the following steps to create the alternate version of the High Performance Reference Design:

1. Download **user_led.zip** from http://www.altera.com/literature/ug/user_led.zip and save it to your desktop.
2. Open and unzip **user_led.zip**.
3. Copy **user_led.v** and **top_hw.v** to your working directory.

This version of **user_led.v** turns on when the Link Training and Status and State Machine (LTSSM) enters the Polling.Compliance state (0x3). **top_hw.v** is the top-level wrapper for the PCI Express High Performance Reference Design. It instantiates **user_led.v** as a separate module.

4. Move or copy the **cvp_app_src** to a subdirectory of your working directory.

This alternate version of **user_led.v** turns on the LED whenever bit[23] of a counter is one.

Setting up CvP Parameters in Device and Pin Options GUI for CvP Initialization Mode

Follow these steps to specify CvP parameters using the Quartus II software:

1. On the Quartus II Assignments menu, select **Device**, and then click **Device and Pin Options**
2. Under **Category** select **General**, and then enable following options:
 - a. **Auto-restart configuration after error**. If this option is enabled, CvP restarts after an error is detected.
 - b. **Enable autonomous PCIe HIP mode**.

Checking this box has no affect if you have enabled CvP by turning on **Enable Configuration via the PCIe link** in the Hard IP for PCI Express GUI. The Quartus II software automatically enables autonomous mode by default. In autonomous mode, the control block takes the Hard IP for PCI Express out of reset after periphery image is loaded. The Hard IP for PCI Express responds to configuration requests and memory requests with the normal successful status. The core image is loaded using PCIe link in both CvP initialization and CvP update mode.

The **Enable autonomous PCIe HIP mode** option only has effect if your design has the following two characteristics:

- You are using something other than the PCIe link to load the core image, for example a flash device or Ethernet controller.
- You have not checked **Enable Configuration via the PCIe link** in the Hard IP for PCI Express GUI.

If both of these conditions are true, the following two options are available:

- If you checked **Enable autonomous PCIe HIP mode**, the control block takes the Hard IP for PCI Express out of reset after the periphery image is loaded. The Hard IP responds to configuration requests with Configuration Retry Status (CRS) and memory requests with UR status until the FPGA enters user mode.
- If you did not check **Enable autonomous PCIe HIP mode**, the Hard IP remains in reset until FPGA enters user mode. Link training only occurs after the FPGA enters user mode.

Note:

This parameter only controls functionality for the initial configuration. It allows open PCI Express systems to meet the configuration time requirement defined in the *PCI Express Base Specification*. After the initial configuration, it has no significance because the core image has already been configured.

- c. Leave all other options disabled.
3. Under **Category**, select **Configuration** to specify the configuration scheme and device. Specify the settings in the following table:

Table 5-8: CvP Initialization Mode Configuration Settings

Parameter	Value
Configuration scheme	Active Serial x4.
Configuration mode	Standard .
Configuration device	EPCQ256.
Configuration device I/O voltage	Auto.
Force VCCIO to be compatible with configuration I/O voltage	Leave this option off.
Generate compressed bitstreams	Turn this option off. Because this is a small example design, it does not use a compressed bitstream. For larger designs, using a compressed bitstream significantly reduces configuration time. In addition, a compressed bitstream requires a smaller flash device.
Active serial clock source	100 MHz Internal Oscillator.
Enable input tri-state on active configuration pins in user mode	Leave this option off.

Under **Category** select **CvP Settings**. For **CvP Initialization** mode, specify the following settings in the following table:

Table 5-9: CvP Initialization Category Settings

Parameter	Value
CvP Initialization	Power up and subsequent core configuration
Enable CvP_CONFDONE pin	Turn this option on
Enable open drain on CvP_CONFDONE pin	Turn this option on

These **Configuration** settings use the configuration devices available on the Stratix V GX FPGA Development Board. The **EPCQ256** flash device is far larger than required to load a periphery image.

4. Click **OK** to close the **Device and Pin Options** dialog box.
5. Click **OK** to close the **Device** dialog box.
6. Save your project.

Creating CvP Revisions of the Core Logic Region Using the CvP Revision Design Flow

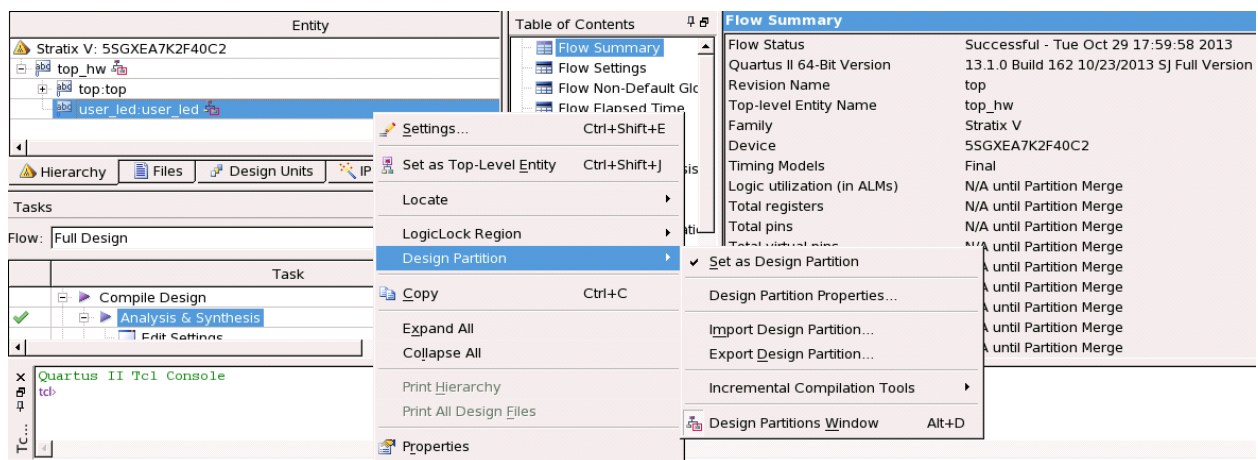
This section provides the instructions to create CvP revisions for the reconfigured core logic region that can be updated. The remainder of the design is treated as a static core region.

Follow these steps to create the base version of the core logic:

1. On the **Assignments** menu, select **Settings** and then select **Files**.
2. In the **File name** box, **Browse** and select **user_led.v**, then click **Add**.

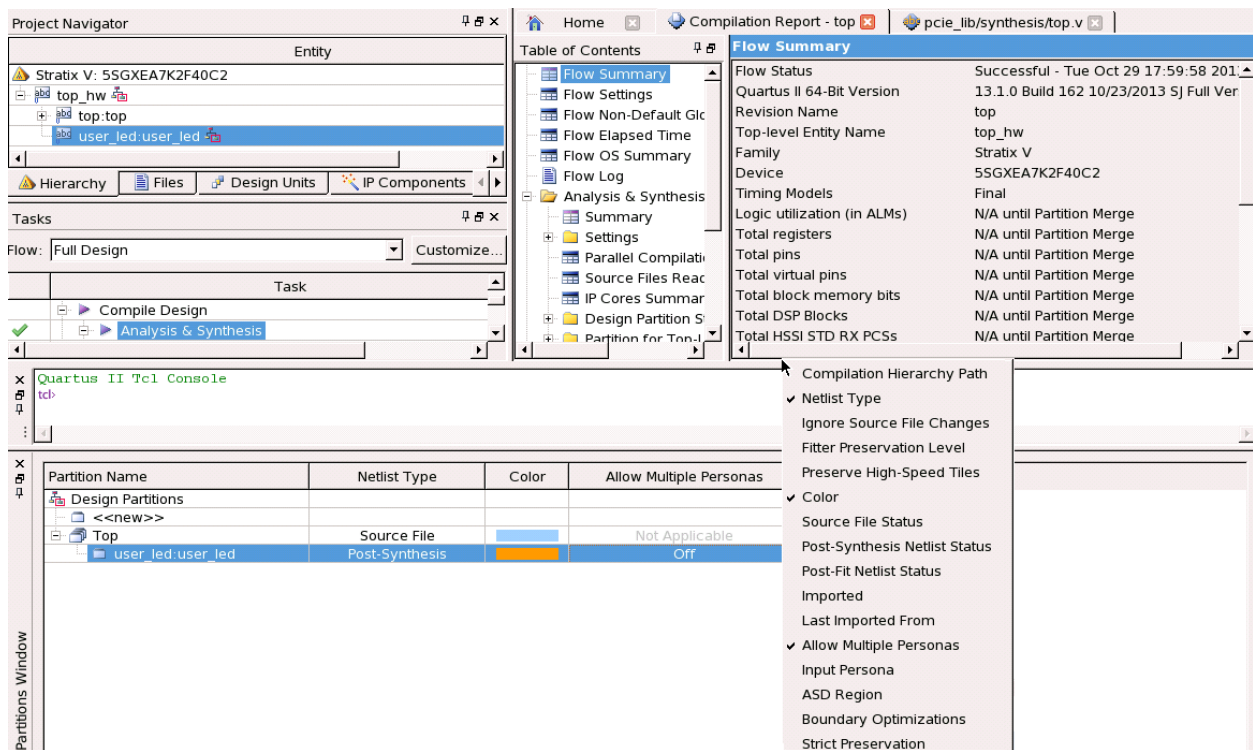
3. Click **OK**.
4. Run **Analysis & Synthesis** so that the Quartus II software parses the design to create a design hierarchy that includes the `user_led` instance.
5. To set `user_led` as a design partition, right click `user_led:user_led` in the design hierarchy and select **Design Partition**. A small read box appears next to `user_led:user_led` indicating that it is a separate partition. (If you perform the same steps again, you remove the separate design partition from `user_led:user_led`.) The following image illustrates this step.

Figure 5-7: Setting a Design Partition



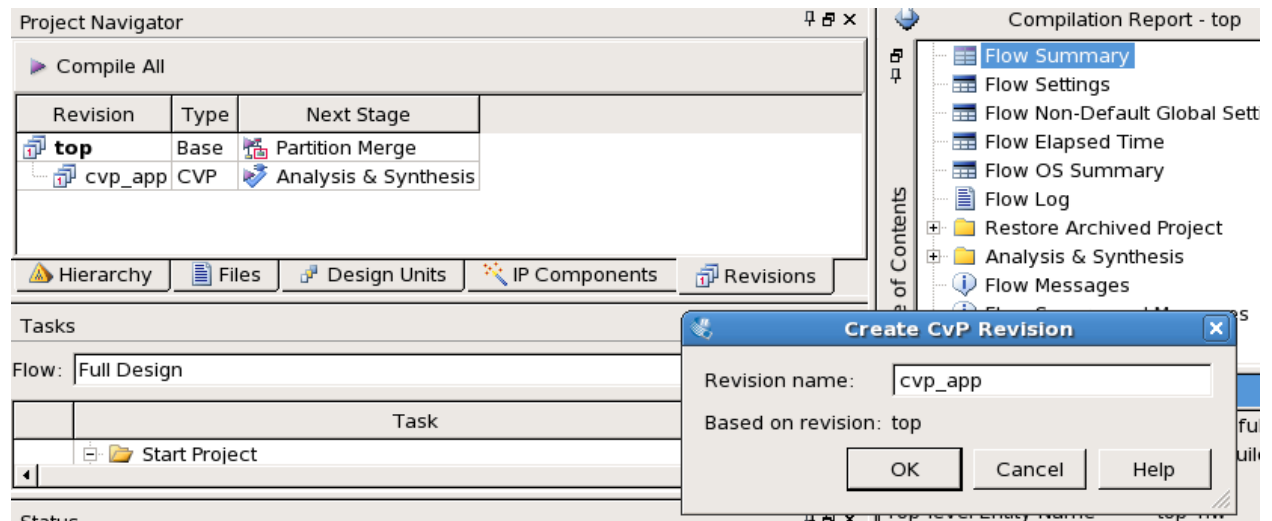
6. Click the **Design Partitions Window** at the bottom of the menu cascade shown in the figure above. The **Design Partitions Window** appears.
7. To add the **Allow Multiple Personas** column to the **Design Partitions Window**, right click on the top bar of **Design Partition Window** next to the **Color** heading and select **Allow Multiple Personas** from the list as shown in the following figure.

Figure 5-8: Allowing Multiple Personas



8. Click the core instance **user_led:user_led** and set **Allow Multiple Personas** to **On**.
9. Click in the **Netlist Type** column and set the **user_led:user_led** Netlist Type to **Source File**.
10. Follow these steps to create a CvP revision for the modified project.
 - a. Under the **Revisions** tab, right click on the Revision **top** and select **Create CvP Revision**. The **Create CvP Revision** dialog box appears.
 - b. For the **Revision name** type `cvp_app` and click **OK** to create a CvP revision as illustrated in the following figure.

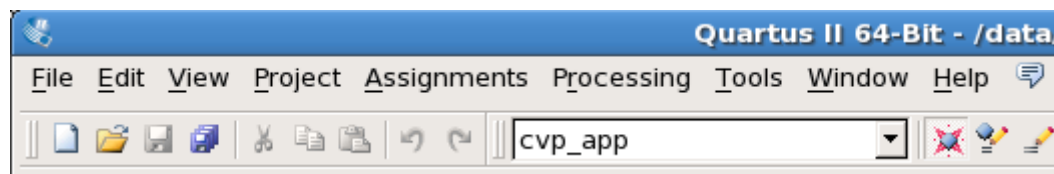
Figure 5-9: Specifying Revision Name



c. Save the Quartus II project.

11. Now change to the CvP revision in the Quartus II software design revision list as shown in the following figure.

Figure 5-10: Changing the CvP Revision



12. To remove **user_led.v** from the **cvp_app** revision, on the **Assignments** menu, select **Settings**, then select **Files**.

This is the original **user_led.v** file that turns on the LED when the LTSSM enters the Polling.Compliance state.

13. In the **Files** list, click **user_led.v**, then click **Remove**.

14. To add **cvp_app_src/user_led.v** for the **cvp_app** revision, in the **File name** box, click **Browse** and browse to **cvp_app_src/user_led.v**, then click **Add**.

This is the modified **user_led.v** file that turns on the LED when the bit[23] of a counter is one.

15. In the **File name** box, click **Browse** and browse to **cvp_app_src/user_led.v**, then click **Add**.

16. Click **OK**.

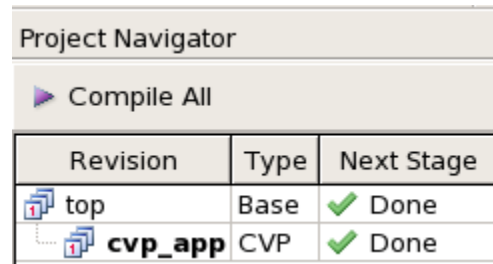
17. In the **Partition Name** window, select **user_led:user_led** and change the **Netlist Type** to **Source File** and turn On **Allow Multiple Personas**.

18. Change back to the **top** revision in the Quartus II software design revision list.

Compiling Both the Base and cvp_app Revisions in the CvP Revision Design Flow

To compile your project, click **Compile All**. Using **Compile All** guarantees that the Quartus II software compiles the **top** revision and **cvp_app** revision in the correct order.

Figure 5-11: Compile Both Base and cvp_app Revisions Using the Compile All Option in Quartus 13.0



Revision	Type	Next Stage
top	Base	✓ Done
cvp_app	CVP	✓ Done

You might need to go through a few iterations of compiling your designs to separate all of the periphery components from the core logic. As a result, the final design might not maintain the functional relationships between logic blocks that you originally planned. Adding extra comments in your design will help you to trace the HDL.

You must compile your project to update the reconfigured core image if any of the following conditions are true:

- The CvP revision has never been compiled.
- You have changed the periphery logic.
- You have changed the wrapper file for any of the core revisions.
- You have migrated to a new version of the Quartus II software.
- You have changed any project settings in the Quartus II Settings File (*.qsf).

Splitting the SOF File for the CvP Initialization Design Mode

Follow these steps to split your .sof file into separate images for the periphery and core logic.

1. On the File menu, select **Convert Programming File**.
2. Under **Output programming files to convert**, specify the options in the following table.

Table 5-10: CvP Initialization Output Programming Files Settings

Parameter	Value
Programming file type	JTAG Indirect Configuration File (.jic).
Configuration device	EPCQ256.
Mode	Active Serial x4.
File name	Browse to and select the <code>./pcie_quartus_files/</code> directory. Type the file name <code>top.jic</code> . Then click Save .
Create Memory Map File	Turn this option on.
Create CvP files	Turn this option on. This box is greyed out until you specify the SOE Data file under Input files to convert .

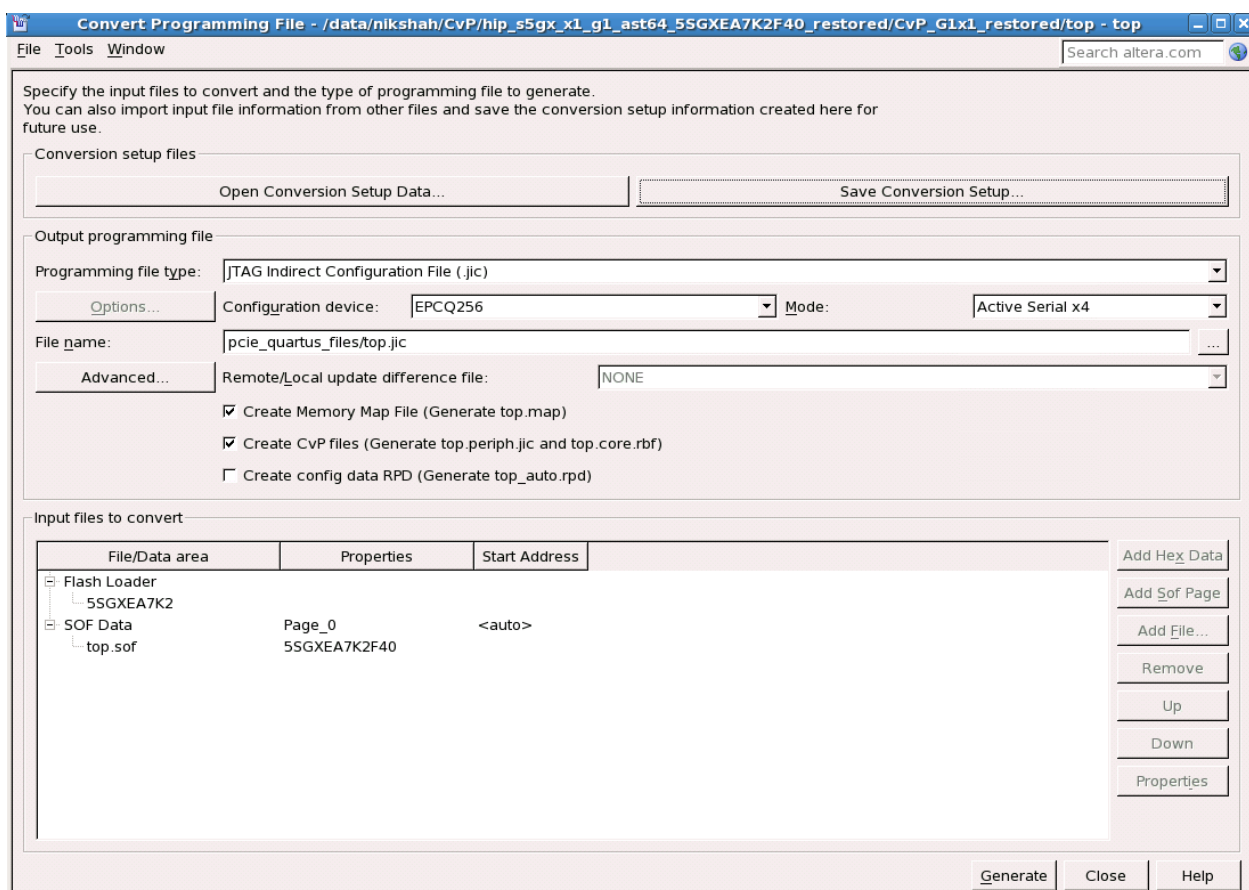
3. Under **Input files to convert**, specify the options in the following table:

Table 5-11: CvP Initialization Input Files to Convert Settings

Parameter	Value
Click Flash Loader	Click Add Device and select Stratix V and then 5SGXEA7K2 , and click OK .
Click SOF Data	Click Add File and navigate to <code>./pcie_quartus_files/top.sof</code> . If you specified a compressed or encrypted bitstream in the Device and Pin Options dialog box, you must specify the same options for Conversion Programming File window. To enable these settings, click <code>top.sof</code> . Then click Properties and check the appropriate boxes.
Mode	Active Serial x4 .

The following figure illustrates the options that you specified.

Figure 5-12: CvP Initialization Mode: Convert Programming File Settings



- Now turn on the **Create CvP files (Generate top.periph.jic and top.core.rbf)** parameter in the **Output Programming Files** section.

Note: If you do not check this box, the Quartus II software does not create separate files for the periphery and core images.

5. Click **Save Conversion Setup** to save these settings. For this exercise, call your settings **cvp_base.cof**. The Quartus II software does not automatically save your choices.
6. Click **Generate** to create **top.periph.jic** and **top.core.rbf**.

Splitting the SOF File for the CvP Initialization Mode with the CvP Revision Design Flow

To implement the **CvP Revision Design Flow** in CvP initialization mode, you must replace the base **.sof** (**top.sof**) with the revision **.sof** (**cvp_app1.sof**). You must also specify a different file name for the CvP revision. This example uses **cvp_app.jic**. The periphery and the core images created for the CvP revision are **cvp_app.periph.jic** and **cvp_app.core.rbf**, respectively. Follow these steps to create the periphery and core images for the CvP revision:

1. On the File menu, select **Convert Programming File**. Under **Output programming file** specify the options in the following table. These settings are illustrated in the figure below.

Table 5-12: CvP Revision Design Flow: Output Programming File Options

Parameter	Value
Programming file type	JTAG Indirect Configuration File (.jic)
Configuration device	EPCQ256
Mode	Active Serial x4
File name	Click browse and specify pcie_quartus_files/cvp_app.jic
Create Memory Map File	Turn this option on.
Create CvP files	Turn this option on. This box is greyed out until you specify the SOF Data file under Input files to convert .

2. Under **Input files to convert** specify the options in the following table:

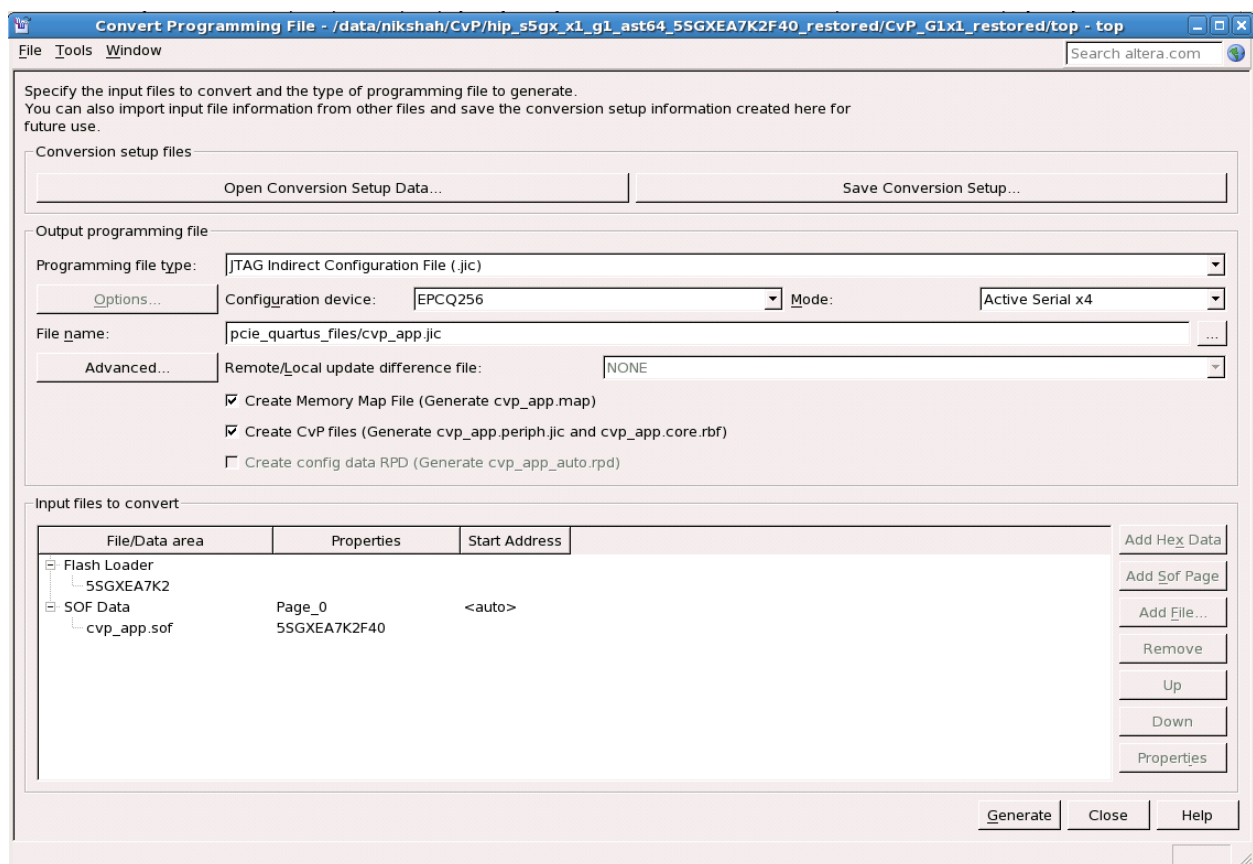
Table 5-13: CvP Revision Design Flow: Input Files to Convert

Parameter	Value
Flash Loader	Click Add Device and select Stratix V and then 5SGXEA7K2

Parameter	Value
Click SOF Data	Click Add File and navigate to <code>./pcie_quartus_files/cvp_app.sof</code> . If you specified a compressed or encrypted bitstream in the Device and Pin Options dialog box, you must specify the same options in the Conversion Programming File window. To enable these settings, click <code>cvp_app.sof</code> , then click Properties and check the appropriate boxes.

- Now turn on the **Create CvP files (Generate cvp_app1.periph.jic and cvp_app1.core.rbf)** parameter in the **Output Programming Files** section.
- To save the **Conversion Programming File** parameters, click **Save Conversion Setup** and type an output file name. Saving your conversion setup saves time on subsequent conversions. For this exercise, call your settings `cvp_revision.cof`. You can reload the conversion parameters by opening the Conversion Setup File (`*.cof`) using **Open Conversion Setup Data** in the **Convert Programming File** window. The following figure illustrates these options.
- Click **Generate**. This conversion process creates the alternate core logic, `cvp_app.core.rbf`, and the periphery logic, `cvp_app.periph.jic`. The periphery logic, `cvp_app.periph.jic`, should be identical to the periphery logic created when splitting the base revision.

Figure 5-13: CvP Revision Design Flow: Convert Programming Files



- Now proceed to **Bringing Up the Hardware** on page 5-37.

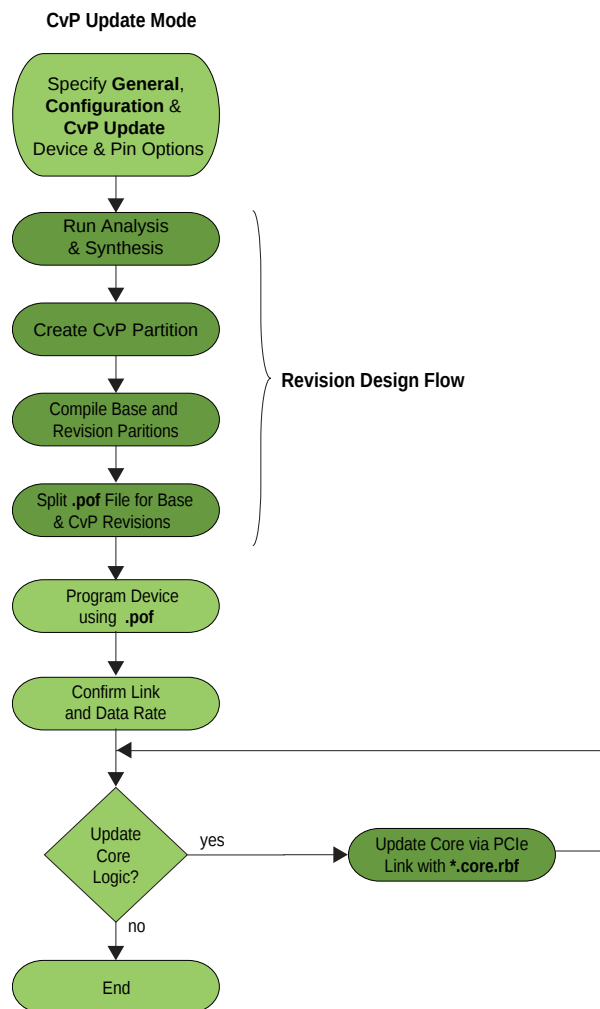
The file **top.cof** provided in *_cvp* designs is a template for CvP initialization mode. You can open this file in the **Open Conversion Setup Data** of **Convert Programming File** window to retrieve the parameters shown in figure above.

Understanding the Design Steps for CvP Update Mode

CvP update mode divides the design into periphery and core images. Initially, you program the entire image using conventional programming options. Subsequently, you can download alternative versions of the core image using the PCI Express link.

You specify this mode in the Quartus II software by selecting the CvP Setting **Subsequent Core Reconfiguration**. The following figure provides the high-level steps for CvP update mode.

Figure 5-14: Design Flow for CvP Update Mode



Note: When you select CvP initialization mode, you must use the CMU PLL and the hard reset controller for the PCI Express Hard IP.

In the walkthrough, CvP update mode includes the following steps:

1. [Downloading and Generating the High Performance Reference Design](#) on page 5-3
2. [Workaround for a Known Issue with Transceiver Reconfiguration Controller IP Core](#) on page 5-13
3. [Creating An Alternate user_led.v File for the Reconfigurable Core Region](#) on page 5-13
4. [Setting up CvP Parameters in Device and Pin Options GUI for CvP Update Mode for CvP Revision](#) on page 5-27
5. [Creating CvP Revisions of the Core Logic Region Using the CvP Revision Design Flow](#) on page 5-15
6. [Compiling the Design for the CvP Update Mode](#) on page 5-33
7. [Splitting the SOF File for the CvP Update Design Mode](#) on page 5-33
8. [Bringing Up the Hardware](#) on page 5-37

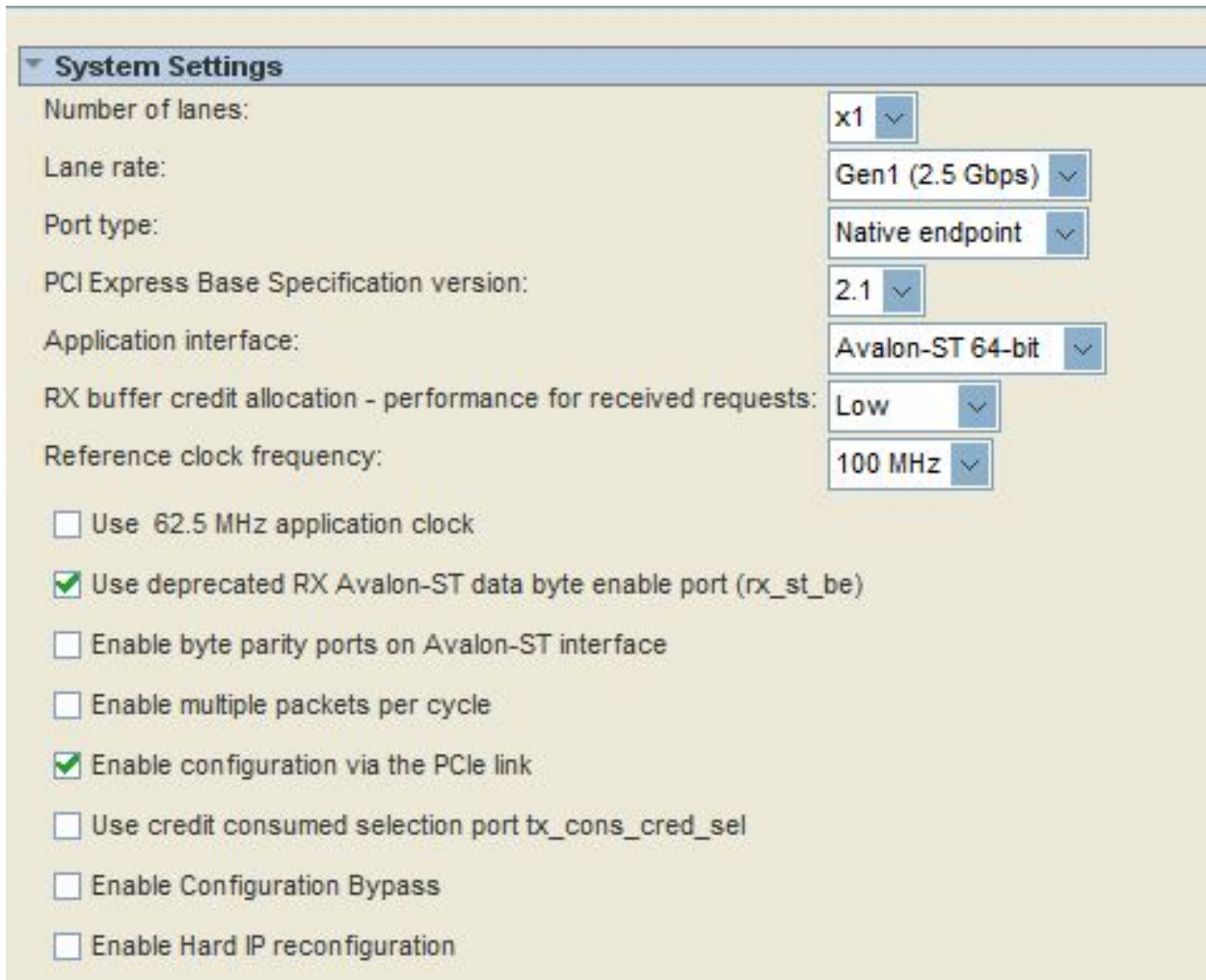
By default, once the FPGA enters user mode, you can reprogram the original *static* core image. If you want to have multiple core images in user mode, you can use the **CvP Revision Design Flow** to create multiple core images that connect to the same periphery image.

Downloading and Generating the High Performance Reference Design

Follow these steps to regenerate the PCI Express High Performance Reference Design with CvP enabled:

1. Download the **PCIe_hiperf_s5gx.zip** file from the PCI Express Avalon-ST High Performance Reference Design web page. This design includes the correct pin assignments and project settings to target the Stratix V GX FPGA Development Kit.
2. Unzip **PCIe_hiperf_s5gx.zip**.
3. Copy **hip_s5gx_x1_g1_ast64_5SGXEA7K2F40.qar** to your working directory.
4. Start the Quartus II software and restore **hip_s5gx_x1_g1_ast64_5SGXEA7K2F40.qar**.
5. On the Tools menu, select **Qsys**.
6. Open **top.qsys**.
7. On the **System Contents** tab, right-click **DUT** and select **Edit**.
8. Under **System Settings**, turn on **Enable configuration via the PCIe link** as shown in the following figure.

Figure 5-15: Hard IP for PCI Express GUI



System Settings

Number of lanes: x1

Lane rate: Gen1 (2.5 Gbps)

Port type: Native endpoint

PCI Express Base Specification version: 2.1

Application interface: Avalon-ST 64-bit

RX buffer credit allocation - performance for received requests: Low

Reference clock frequency: 100 MHz

Use 62.5 MHz application clock

Use deprecated RX Avalon-ST data byte enable port (rx_st_be)

Enable byte parity ports on Avalon-ST interface

Enable multiple packets per cycle

Enable configuration via the PCIe link

Use credit consumed selection port tx_cons_cred_sel

Enable Configuration Bypass

Enable Hard IP reconfiguration

9. Click **Finish**.

10. On the **Generation** tab, specify the settings in the following table. Then click **Generate** at the bottom of the window.

Table 5-14: Qsys Generation Tab Settings

Parameter	Value
Create simulation model	None
Create testbench Qsys system	None
Create testbench simulation model	None
Create HDL design files for synthesis	Verilog
Create block symbol file (.bsf)	Leave this entry off

Parameter	Value
Path	< working_dir > top
Simulation	Leave this entry blank
Testbench	<working_dir> /top /synthesis

11. After successful compilation, close Qsys.

Related Information

[PCI Express Avalon-ST High Performance Reference Design](#)

Workaround for a Known Issue with Transceiver Reconfiguration Controller IP Core

If you plan to perform almost continuous updates of the reconfigurable core logic in stress tests or in your actual system, you may encounter an issue with the Transceiver Reconfiguration Controller. This issue might cause the PCIe link to downtrain in Quartus II 13.0 release and earlier versions of the Quartus II software. If you are using Quartus II 13.0 SP1 or later versions of the Quartus II software, then you may not encounter this issue. Complete the following steps to avoid this issue:

1. Open **pcie_lib/top.v**.
2. Search for the Reconfiguration Controller instance named `alt_xcvr_reconfig` and comment out the entire `reconfig_controller` in **top.v**. (The Transceiver Reconfiguration Controller instance includes 32 lines of Verilog HDL code.)
3. Add the 5 lines of Verilog HDL shown in below after the commented out instance,

```
alt_xcvr_reconfig:
```

```
wire [69:0] reconfig_to_xcvr_bus = {24'h0, 2'b11, 44'h0};
```

```
assign pcie_reconfig_driver_0_reconfig_mgmt_waitrequest = 1'b0;
```

```
assign pcie_reconfig_driver_0_reconfig_mgmt_readdata = 32'h0;
```

```
assign alt_xcvr_reconfig_0_reconfig_busy_reconfig_busy = 1'b0;
```

```
assign alt_xcvr_reconfig_0_reconfig_to_xcvr_reconfig_to_xcvr = { 2 {reconfig_to_xcvr_bus}};
```

In this example, the first statement hardwires the `reconfig_to_xcvr_bus` to the correct values per channel. The first three assignment statements specify the correct values for the `waitrequest`, `readdata`, `reconfig_busy` signals. The final assignment statement for `alt_xcvr_reconfig_0_reconfig_to_xcvr_reconfig_to_xcvr` represents the full reconfiguration bus for all active transceiver channels. This bus is replicated 2 times because 2 channels are active in the Gen1 x1 instance.

Creating An Alternate `user_led.v` File for the Reconfigurable Core Region

This example design creates a new version of the PCI Express High Performance Reference Design. The original version of this reference design includes an LED which turns on whenever the Link Training and Status and State Machine (LTSSM) enters the Polling.Compliance state (0x3). The alternate version of **user_led.v** turns on the LED whenever bit[23] of a counter is one. The LED is instantiated as a separate module in the High Performance Reference Design to demonstrate the steps necessary to create a design with multiple versions of the core logic.

Complete the following steps to create the alternate version of the High Performance Reference Design:

1. Download **user_led.zip** from http://www.altera.com/literature/ug/user_led.zip and save it to your desktop.
2. Open and unzip **user_led.zip**.
3. Copy **user_led.v** and **top_hw.v** to your working directory.

This version of **user_led.v** turns on when the Link Training and Status and State Machine (LTSSM) enters the Polling.Compliance state (0x3). **top_hw.v** is the top-level wrapper for the PCI Express High Performance Reference Design. It instantiates **user_led.v** as a separate module.

4. Move or copy the **cvp_app_src** to a subdirectory of your working directory.

This alternate version of **user_led.v** turns on the LED whenever bit[23] of a counter is one.

Setting up CvP Parameters in Device and Pin Options GUI for CvP Update Mode for CvP Revision

Follow these steps to specify CvP parameters using the Quartus II software:

1. On the Assignments menu, select **Device**, and then click **Device and Pin Options . . .**
2. Under **Category** first select **General**, and then enable following options:
 - a. **Auto-restart configuration after error.** If this option is enabled, CvP restarts after an error is detected.
 - b. **Enable autonomous PCIe HIP mode.**

Checking this box has no affect if you have enabled CvP by turning on **Enable Configuration via the PCIe link** in the Hard IP for PCI Express GUI. The Quartus II software automatically enables autonomous mode by default. In autonomous mode, the control block takes the Hard IP for PCI Express out of reset after periphery image is loaded. The Hard IP for PCI Express responds to configuration requests and memory requests with the normal successful status. The core image is loaded using PCIe link in both CvP initialization and CvP update mode.

The **Enable autonomous PCIe HIP mode** option only has effect if your design has the following two characteristics:

- You are using something other than the PCIe link to load the core image, for example a flash device or Ethernet controller.
- You have not checked **Enable Configuration via the PCIe link** in the Hard IP for PCI Express GUI.

If both of these conditions are true, the following two options are available:

- If you checked **Enable autonomous PCIe HIP mode**, the control block takes the Hard IP for PCI Express out of reset after the periphery image is loaded. The Hard IP responds to configuration requests with Configuration Retry Status (CRS) and memory requests with UR status until the FPGA enters user mode.
- If you did not check **Enable autonomous PCIe HIP mode**, the Hard IP remains in reset until FPGA enters user mode. Link training only occurs after the FPGA enters user mode.

Note:

This parameter only controls functionality for the initial configuration. It allows open PCI Express systems to meet the configuration time requirement defined in the *PCI Express Base Specification*. After the initial configuration, it has no significance because the core image has already been configured.

- c. Leave all other options disabled.
3. Under **Category** select **Configuration** to specify the configuration scheme and device. Specify the settings in the following table:

Table 5-15: CvP Update Mode Configuration Settings

Parameter	Value
Configuration scheme	Passive Serial
Configuration mode	Standard
Configuration device	Auto
Configuration device I/O voltage	Auto
Force VCCIO to be compatible with configuration I/O voltage	Leave this option off
Generate compressed bitstreams	Leave this option on
Active serial clock source	100 MHz Internal Oscillator
Enable input tri-state on active configuration pins in user mode	Leave this option off

Under **Category** select **CvP Settings**. Specify the settings in the following table:

Table 5-16: CvP Update Category Settings

Parameter	Value
CvP via Protocol	Subsequent core configuration
Enable CvP_CONFDONE pin	Turn this option on
Enable open drain on CvP_CONFDONE pin	Turn this option on

4. Click **OK** to close the **Device and Pin Options** dialog box.
5. Click **OK** to close the **Device** dialog box.
6. Save your project.

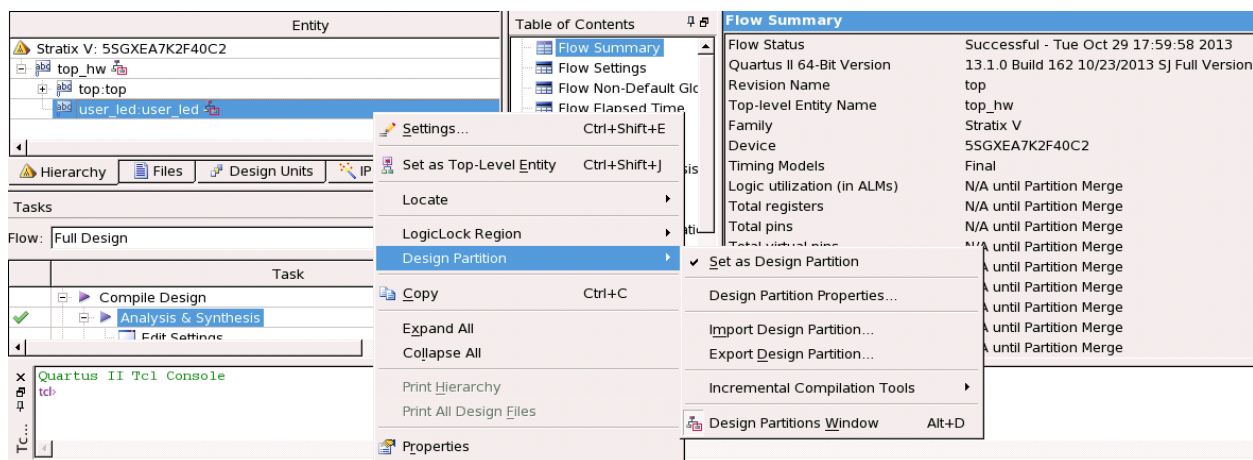
Creating CvP Revisions of the Core Logic Region Using the CvP Revision Design Flow

This section provides the instructions to create CvP revisions for the reconfigured core logic region that can be updated. The remainder of the design is treated as a static core region.

Follow these steps to create the base version of the core logic:

1. On the **Assignments** menu, select **Settings** and then select **Files**.
2. In the **File name** box, **Browse** and select **user_led.v**, then click **Add**.

3. Click **OK**.
4. Run **Analysis & Synthesis** so that the Quartus II software parses the design to create a design hierarchy that includes the **user_led** instance.
5. To set **user_led** as a design partition, right click **user_led:user_led** in the design hierarchy and select **Design Partition**. A small read box appears next to **user_led :user:led** indicating that it is a separate partition. (If you perform the same steps again, you remove the separate design partition from **user_led:user_led**.) The following image illustrates this step.

Figure 5-16: Setting a Design Partition

6. Click the **Design Partitions Window** at the bottom of the menu cascade shown in the figure above. The **Design Partitions Window** appears.
7. To add the **Allow Multiple Personas** column to the **Design Partitions Window**, right click on the top bar of **Design Partition Window** next to the **Color** heading and select **Allow Multiple Personas** from the list as shown in the following figure.

Figure 5-17: Allowing Multiple Personas

The screenshot displays the Quartus II IDE interface. The 'Partitions Window' at the bottom shows a table with the following data:

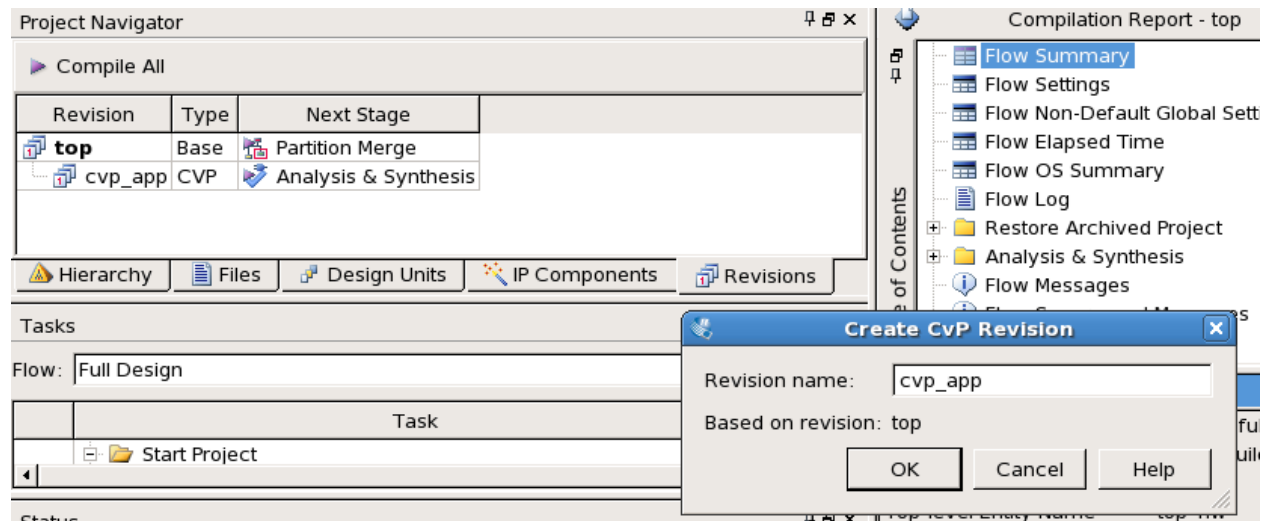
Partition Name	Netlist Type	Color	Allow Multiple Personas
Design Partitions			
<<new>>			
Top	Source File		Not Applicable
user_led:user_led	Post-Synthesis		Off

A context menu is open over the 'user_led:user_led' row, showing the following options:

- Compilation Hierarchy Path
- Netlist Type
- Ignore Source File Changes
- Fitter Preservation Level
- Preserve High-Speed Tiles
- Color
- Source File Status
- Post-Synthesis Netlist Status
- Post-Fit Netlist Status
- Imported
- Last Imported From
- Allow Multiple Personas
- Input Persona
- ASD Region
- Boundary Optimizations
- Strict Preservation

8. Click the core instance **user_led:user_led** and set **Allow Multiple Personas** to **On**.
9. Click in the **Netlist Type** column and set the **user_led:user_led** Netlist Type to **Source File**.
10. Follow these steps to create a CvP revision for the modified project.
 - a. Under the **Revisions** tab, right click on the Revision **top** and select **Create CvP Revision**. The **Create CvP Revision** dialog box appears.
 - b. For the **Revision name** type **cvp_app** and click **OK** to create a CvP revision as illustrated in the following figure.

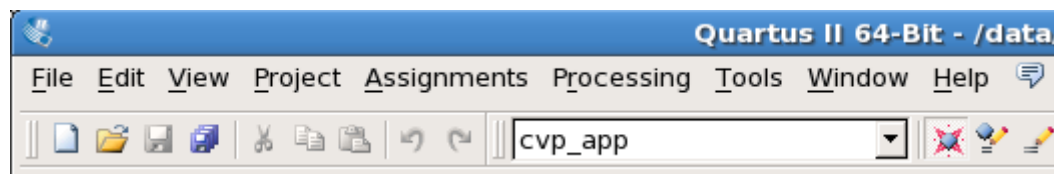
Figure 5-18: Specifying Revision Name



c. Save the Quartus II project.

11. Now change to the CvP revision in the Quartus II software design revision list as shown in the following figure.

Figure 5-19: Changing the CvP Revision



12. To remove **user_led.v** from the **cvp_app** revision, on the **Assignments** menu, select **Settings**, then select **Files**.

This is the original **user_led.v** file that turns on the LED when the LTSSM enters the Polling.Compliance state.

13. In the **Files** list, click **user_led.v**, then click **Remove**.

14. To add **cvp_app_src/user_led.v** for the **cvp_app** revision, in the **File name** box, click **Browse** and browse to **cvp_app_src/user_led.v**, then click **Add**.

This is the modified **user_led.v** file that turns on the LED when the bit[23] of a counter is one.

15. In the **File name** box, click **Browse** and browse to **cvp_app_src/user_led.v**, then click **Add**.

16. Click **OK**.

17. In the **Partition Name** window, select **user_led:user_led** and change the **Netlist Type** to **Source File** and turn On **Allow Multiple Personas**.

18. Change back to the **top** revision in the Quartus II software design revision list.

Setting up CvP Parameters in Device and Pin Options GUI for CvP Update Mode for CvP Revision

Follow these steps to specify CvP parameters using the Quartus II software:

1. On the Assignments menu, select **Device**, and then click **Device and Pin Options . . .**
2. Under **Category** first select **General**, and then enable following options:
 - a. **Auto-restart configuration after error.** If this option is enabled, CvP restarts after an error is detected.
 - b. **Enable autonomous PCIe HIP mode.**

Checking this box has no affect if you have enabled CvP by turning on **Enable Configuration via the PCIe link** in the Hard IP for PCI Express GUI. The Quartus II software automatically enables autonomous mode by default. In autonomous mode, the control block takes the Hard IP for PCI Express out of reset after periphery image is loaded. The Hard IP for PCI Express responds to configuration requests and memory requests with the normal successful status. The core image is loaded using PCIe link in both CvP initialization and CvP update mode.

The **Enable autonomous PCIe HIP mode** option only has effect if your design has the following two characteristics:

- You are using something other than the PCIe link to load the core image, for example a flash device or Ethernet controller.
- You have not checked **Enable Configuration via the PCIe link** in the Hard IP for PCI Express GUI.

If both of these conditions are true, the following two options are available:

- If you checked **Enable autonomous PCIe HIP mode**, the control block takes the Hard IP for PCI Express out of reset after the periphery image is loaded. The Hard IP responds to configuration requests with Configuration Retry Status (CRS) and memory requests with UR status until the FPGA enters user mode.
- If you did not check **Enable autonomous PCIe HIP mode**, the Hard IP remains in reset until FPGA enters user mode. Link training only occurs after the FPGA enters user mode.

Note:

This parameter only controls functionality for the initial configuration. It allows open PCI Express systems to meet the configuration time requirement defined in the *PCI Express Base Specification*. After the initial configuration, it has no significance because the core image has already been configured.

- c. Leave all other options disabled.
3. Under **Category** select **Configuration** to specify the configuration scheme and device. Specify the settings in the following table:

Table 5-17: CvP Update Mode Configuration Settings

Parameter	Value
Configuration scheme	Passive Serial
Configuration mode	Standard
Configuration device	Auto
Configuration device I/O voltage	Auto
Force VCCIO to be compatible with configuration I/O voltage	Leave this option off
Generate compressed bitstreams	Leave this option on
Active serial clock source	100 MHz Internal Oscillator

Parameter	Value
Enable input tri-state on active configuration pins in user mode	Leave this option off

Under **Category** select **CvP Settings**. Specify the settings in the following table:

Table 5-18: CvP Update Category Settings

Parameter	Value
CvP via Protocol	Subsequent core configuration
Enable CvP_CONFDONE pin	Turn this option on
Enable open drain on CvP_CONFDONE pin	Turn this option on

4. Click **OK** to close the **Device and Pin Options** dialog box.
5. Click **OK** to close the **Device** dialog box.
6. Save your project.

Compiling the Design for the CvP Update Mode

1. To compile the design, on the **Processing** menu, select **Start compilation**. Compilation creates a **.pof** file in the **pcie_quartus_files** subdirectory.

You might need to go through a few iterations of compiling your designs to separate all of the periphery components from the core logic. As a result, the final design might not maintain the functional relationships between logic blocks that you originally planned. Adding extra comments in your design will help you to trace the HDL.

Splitting the SOF File for the CvP Update Design Mode

Follow these steps to split your to create file into periphery and core images for CvP update mode. You use the core image, **top.core.rbf**, to perform CvP updates.

1. On the **File** menu, select **Convert Programming File**.
2. Under **Output programming file** specify the options in the following table. These options are illustrated in the figure below.

Table 5-19: Output Programming File

Parameter	Value
Programming file type	Programmer Object File (.pof).
Configuration device	CFI_128Mb.
Mode	1-bit Passive Serial.
File name	Click browse and specify pcie_quartus_files/top.pof .
Create Memory Map File	Turn this option on.

Parameter	Value
Create CvP files	Turn this option on. This box is greyed out until you specify the SOF Data file under Input files to convert .

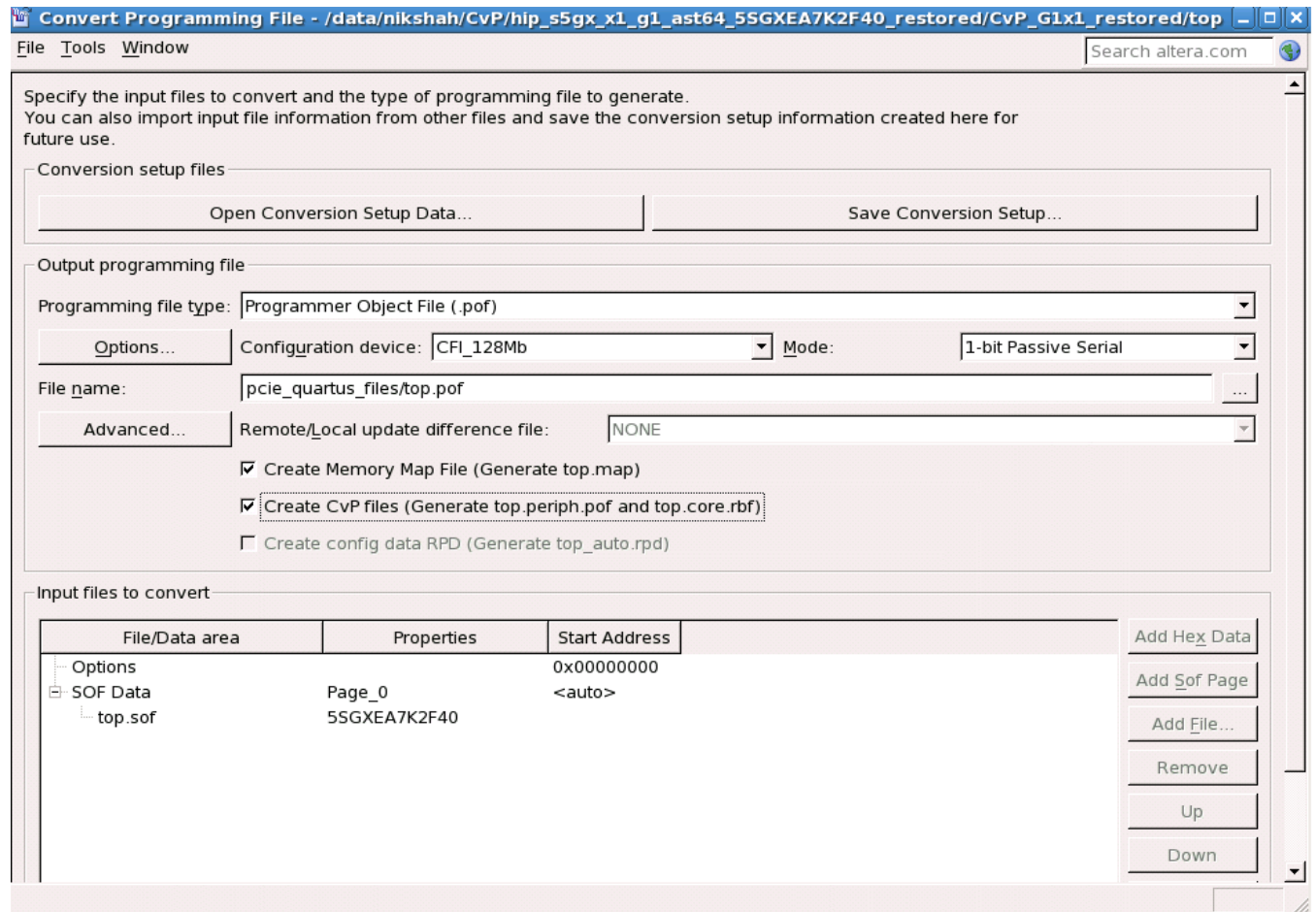
3. Under **Input files to convert** specify the options in the following table:

Table 5-20: Input files to convert

Parameter	Value
Click SOF Data	Click Add File and navigate to <code>./pcie_quartus_files/top.sof</code> . If you specified a compressed or encrypted bitstream in the Device and Pin Options dialog box, you must specify the same options in the Conversion Programming File window. To enable these settings, click <code>top.sof</code> , then click Properties and check the appropriate boxes.

4. Now turn on the **Create CvP files** (**Generate top.periph.jic** and **top.core.rbf** parameter in the **Output Programming Files** section).
5. Click **Generate** to create **top.periph.pof**, and **top.core.rbf**. The periphery file, **top.periph.pof** is generated, but it is not used.

Figure 5-20: Base Revision of CvP Update Mode: Convert Programming File Settings



Note: The **Configuration scheme** and **Configuration device** in **Device and Pin Options** must match with **Configuration Device** and **Mode** in **Convert Programming File** respectively.

Splitting the SOF File for CvP Update Mode with the CvP Revision Design Flow

To implement the **CvP Revision Design Flow** with CvP update mode, you must replace the base **.sof** (**top.sof**) with the revision **.sof** (**cvp_app.sof**). You must also specify a different file name for the CvP revision. This example uses **cvp_app**. The periphery and the core images created for the CvP revision are **cvp_app.periph.pof** and **cvp_app.core.rbf**, respectively. Follow these steps to create the periphery and core images for the CvP revision:

1. On the File menu, select **Convert Programming File**. Under **Output programming file** specify the options in the following table. These settings are illustrated in the figure below.

Table 5-21: CvP Revision Design Flow: Output Programming File Options

Parameter	Value
Programming file type	Programmer Object File (.pof)

Parameter	Value
Configuration device	CFL_128Mb
Mode	1-bit Passive Serial
File name	Click browse and specify pcie_quartus_files/cvp_app.pof
Create Memory Map File	Turn this option on.
Create CvP files	Turn this option on. This box is greyed out until you specify the SOF Data file under Input files to convert .

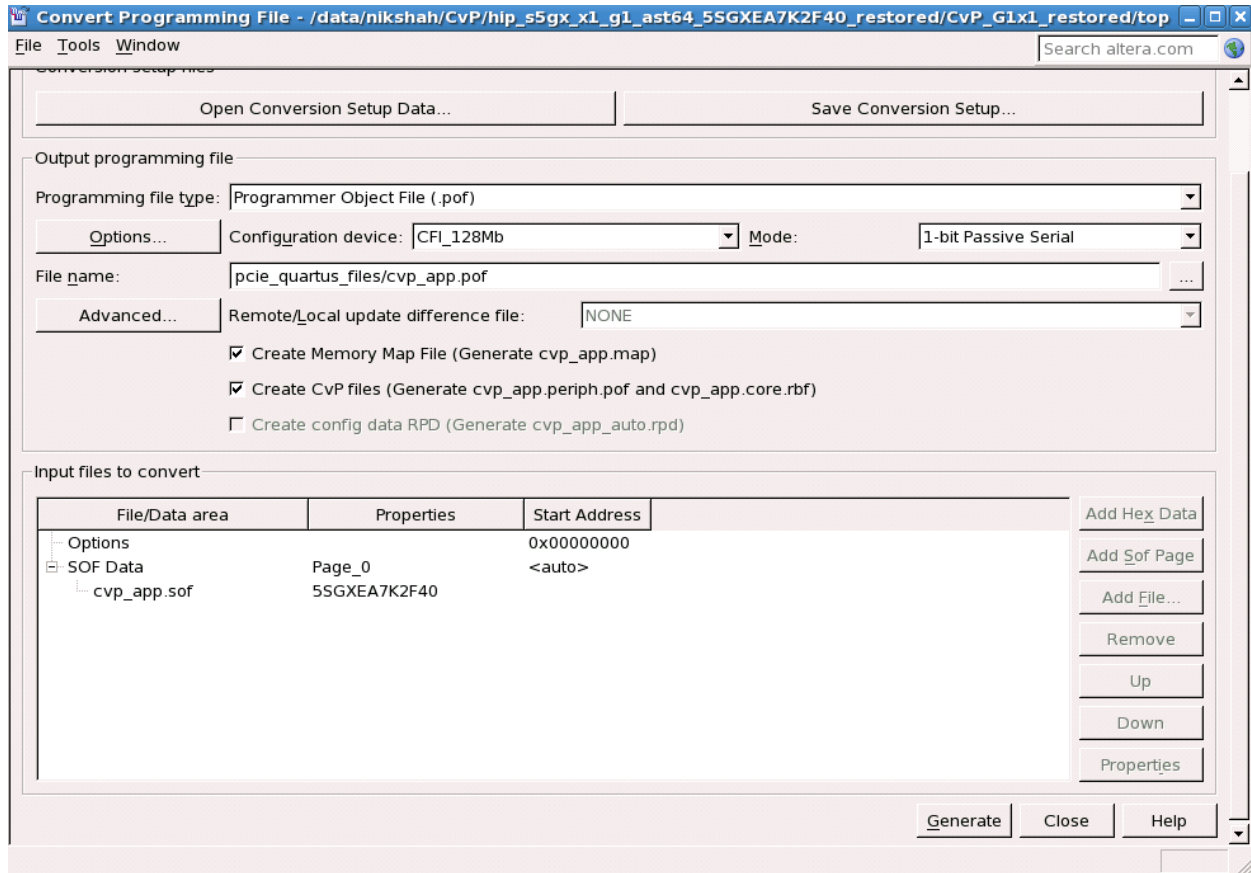
- Under **Input files to convert** specify the options in the following table:

Table 5-22: CvP Revision Design Flow: Input Files to Convert

Parameter	Value
Click SOF Data	Click Add File and navigate to ./pcie_quartus_files/cvp_app.sof . If you specified a compressed or encrypted bitstream in the Device and Pin Options dialog box, you must specify the same options in the Conversion Programming File window. To enable these settings, click cvp_app.sof , then click Properties and check the appropriate boxes.

- Now turn on the **Create CvP files (Generate cvp_app.periph.jic and cvp_app.core.rbf)** parameter in the **Output Programming Files** section.
- To save the **Conversion Programming File** parameters, click **Save Conversion Setup** and type an output file name. Saving your conversion setup saves time on subsequent conversions. For this exercise, call your settings **cvp_update_revision_setup.cof**. You can reload the conversion parameters by opening the Conversion Setup File (*.cof) using **Open Conversion Setup Data** in the **Convert Programming File** window. The following figure illustrates these options.
- Click **Generate**. This conversion process creates the alternate core logic, **cvp_app.core.rbf**, and the periphery logic, **cvp_app.periph.pof**.

Figure 5-21: CvP Revision Design Flow: Convert Programming Files



6. Now proceed to [Bringing Up the Hardware](#) on page 5-37.

The file **top.cof** provided in *_cvp* designs is a template for CvP initialization mode. You can open this file in the **Open Conversion Setup Data** of **Convert Programming File** window to retrieve the parameters shown in figure above.

Bringing Up the Hardware

Before testing the design in hardware, you must install Jungo WinDriver in your DUT system. For Windows, the procedure is described in [Installing Jungo WinDriver in Windows Systems](#) on page 5-38. For Linux, the instructions are [Installing Jungo WinDriver in Windows Systems](#) on page 5-38. You can also install **RW Utilities** or other system verification tools to monitor the link status of the Endpoint and to observe traffic on the link. You can download these utilities for free from many web sites.

The test setup includes the following components:

- Stratix V GX FPGA Development Kit
- USB Blaster
- A DUT PC with PCI Express slotto plug in the Stratix V GX FPGA Development Kit
- A host PC running the Quartus II software to program the periphery image, **.sof** or **.pof** file

Although a separate host PC is not strictly necessary, it makes testing less cumbersome.

Installing Jungo WinDriver in Windows Systems

1. Navigate to `<Quartus II installation path>\quartus\drivers\wdrv\windows<32 or 64>`.
2. Run the command:
 - `wdreg -inf windrvr6.inf install`
3. Copy the `wdapi1021.dll` file to the `%windir%\system32` directory.

Installing Jungo WinDriver in Linux Systems

1. Navigate to `<Quartus II installation path>/quartus/drivers/wdrv/linux<32 or 64>`.
2. Run the following commands:
 - a. `./configure --disable-usb-support`
 - b. `make`
 - c. `su`
 - d. `make install`
3. You can change the permissions for the device file. For example, `chmod 666 /dev/windrvr6`.
4. For 64-bit Linux systems, set the `Quartus_64BIT` environment variable before you run `quartus_cvp` using the following command:
 - `export QUARTUS_64BIT=1`
5. You can use the `quartus_cvp` command to download ***core.rbf** files to your FPGA. The following table lists the `quartus_cvp` commands for all modes.

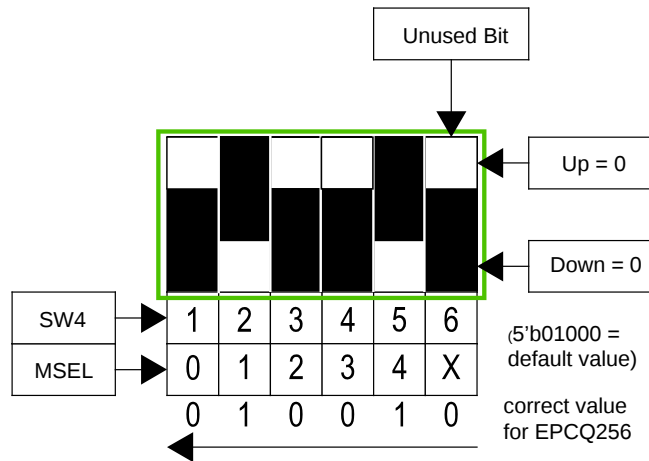
Table 5-23: Syntax for quartus_cvp Commands

Mode	quartus_cvp Command
Uncompressed	<code>quartus_cvp --vid=<Vendor ID> --did=<Device ID> <Core .rbf file path></code>
Unencrypted	
Compressed	<code>quartus_cvp -c --vid=<Vendor ID> --did=<Device ID> <Core .rbf file path></code>
Encrypted	<code>quartus_cvp -e --vid=<Vendor ID> --did=<Device ID> <Core .rbf file path></code>
Compressed and encrypted	<code>quartus_cvp -c -e --vid=<Vendor ID> --did=<Device ID> <Core .rbf file path></code>

Modifying MSEL for Active Serial x4 Flash on Stratix V Dev-Kit

The MSEL switch labeled SW4 on the back of the Stratix V GX FPGA Development Kit PCB specifies the flash type. The correct setting for an active serial x4 flash is 5'b10010 as shown in the following figure. The factory default value is 5'b01000.

Figure 5-22: Switch 4 (SW4) Configuration for MSEL[4:0] = 5'b10010 on the back view of Stratix V Device Kit



In this figure, the switch head outlined in by a green rectangle. The up position signifies logic zero and the down position signifies logic one. The MSB of the switch, SW4[6], is on the far right. This bit is unused and must be set to zero (SW4[6]=up). The MSB bit of MSEL[4] is position 5, the second bit from the right. To set the unused bit to 0 and MSEL[4:0] = 5'b10010, the SW4[6:1] sequence is up(0), down(1), up(0), up(0), down(1), up(0), reading from right to left.

Related Information

[Stratix V GX FPGA Development Kit Reference Manual](#)

Programming CvP Images and Validating the Link

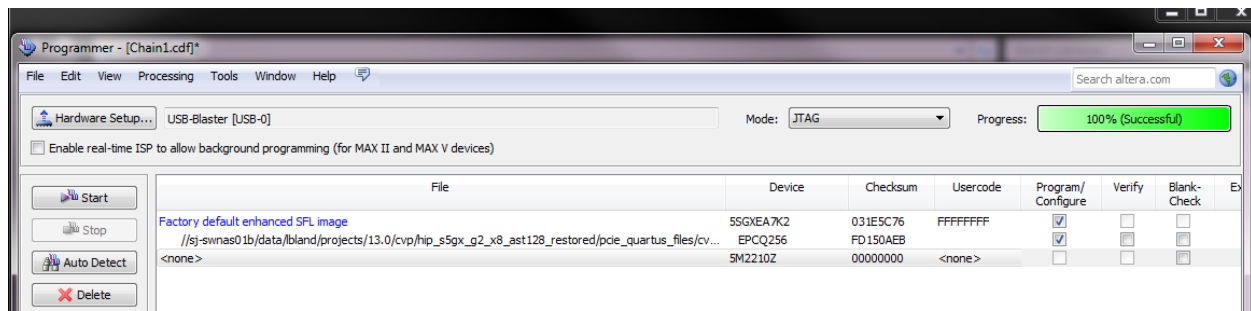
For CvP initialization mode, you must program the periphery image (**top.periph.jic**) and then download the core image core image (**top.core.rbf**) using the PCIe Link. After loading the periphery image via the JTAG port, the link should reach the expected data rate and link width. You can confirm the PCIe link status using the **RW Utilities**. Then, you can update the core image (**top.core.rbf**) using the **quartus_cvp** command.

For In CvP update mode, you first program the FPGA using the **.sof** or **.pof** image. After the programming completes the FPGA enters user mode. You can now reconfigure the core image (**.core.rbf**), the **quartus_cvp** command or your own driver.

Follow these to program and test the CvP functionality:

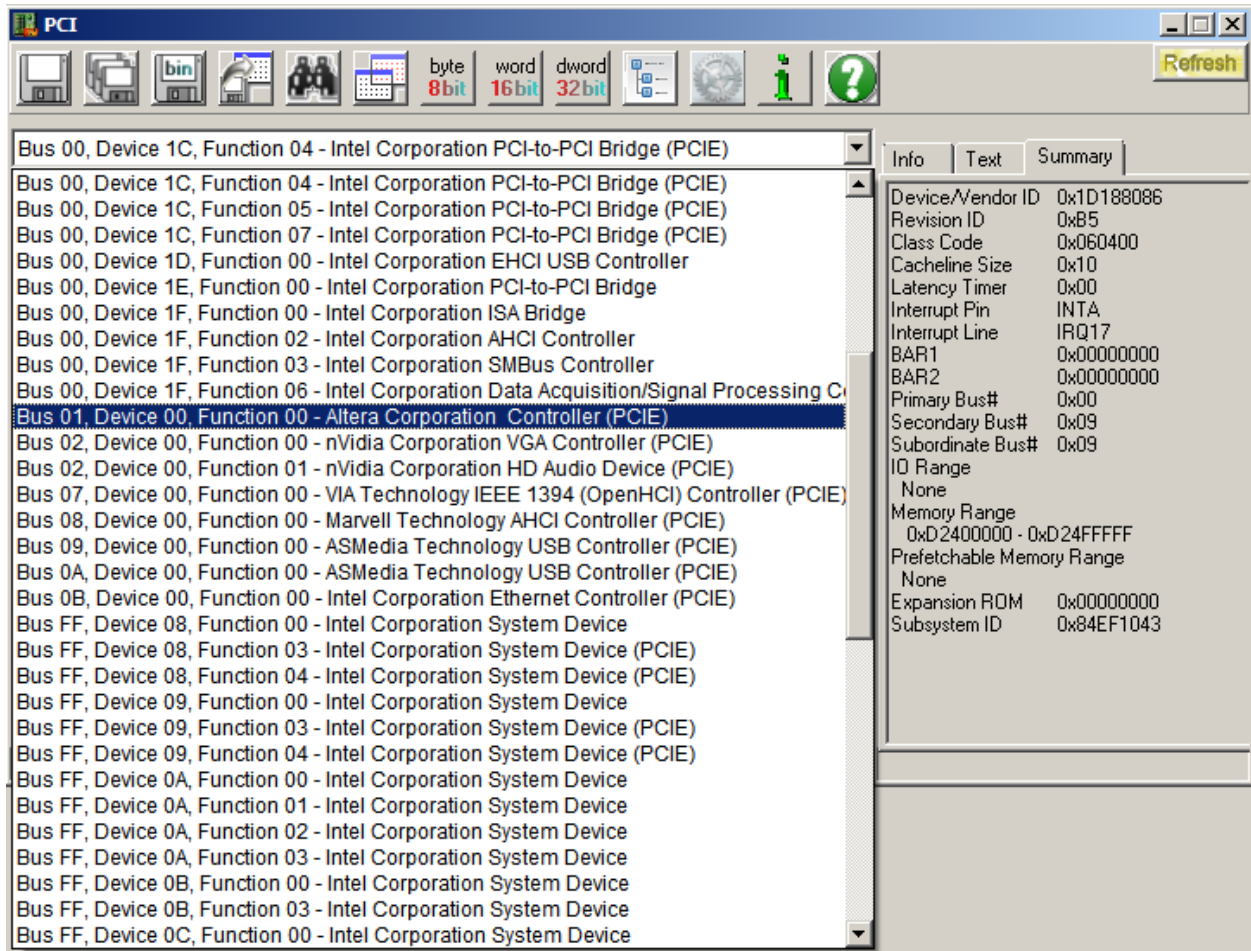
1. Plug the Stratix V GX FPGA Development Kit into the PCI Express slot of the DUT PC and power it on. Altera recommends that you use the external power supply that the development kit includes.
2. On the host PC, open the Quartus II Tools menu and select **Programmer**. The Programmer appears.

Figure 5-23: Quartus II Programmer Settings



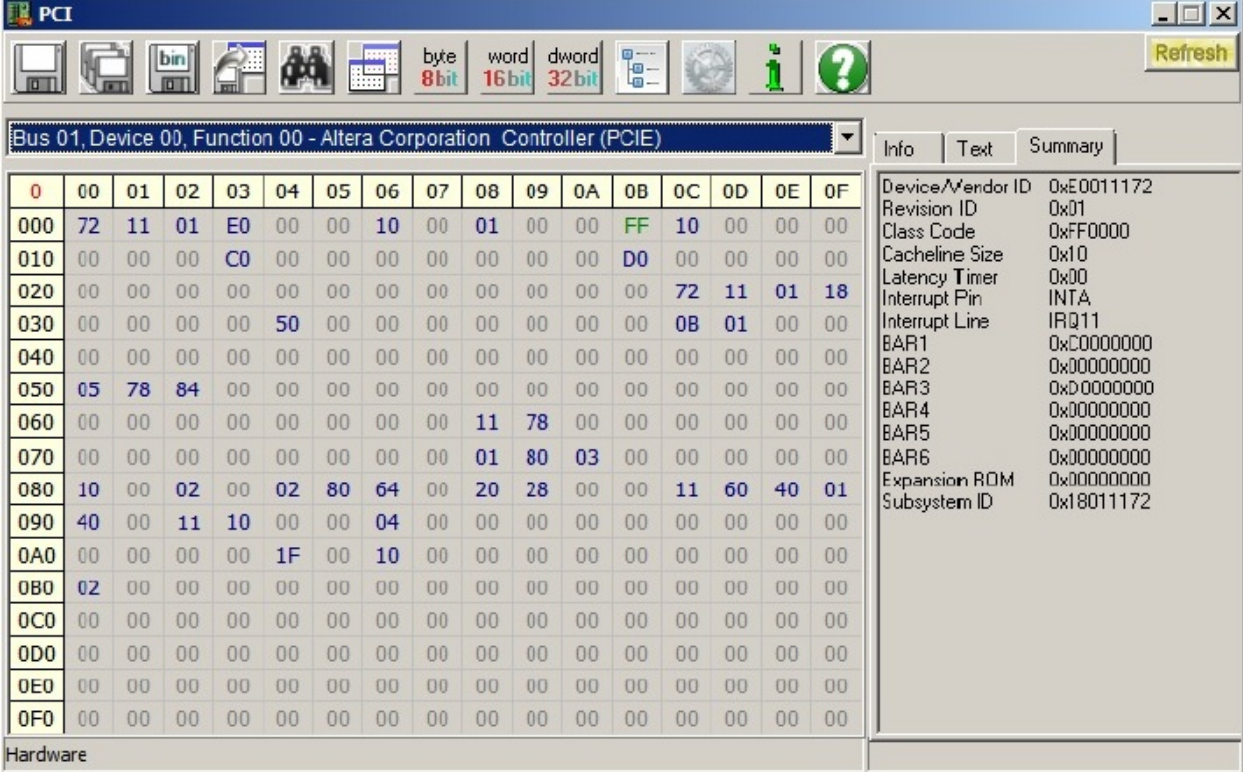
3. Click **Auto Detect** to verify that the USB Blaster recognizes the Stratix V FPGA.
4. Follow these steps to program the periphery image:
 - a. Select Stratix V device, then right click **None** under **File** column.
 - b. Navigate to **pcie_quartus_files/top.periph.jic** and click **Open**.
 - c. Under **Program/Configure** column, select **5SGXEA7K2** and **EPCQ256**.
 - d. Click **Start** to program the periphery image to EPCQ256 flash.
5. To force the host PC to re-enumerate the link with the new image, power cycle the DUT PC and the Stratix V GX High Performance FPGA Development Kit.
6. You can use RW Utilities or another system software driver to verify the link status. The following figure shows that the RW Utilities enumeration includes an Altera PCIe on Bus 01.

Figure 5-24: RW Utilities Transcript



- You can also confirm expected link speed and width. For this Gen1 x1 example design, the following figure shows that both Altera EP Link Capability Register at 0x8C and Link Status register at 0x92 have value 0x11 which confirms the link successfully comes up as Gen1 x1.

Figure 5-25: Checking Link Status



The screenshot shows the PCI configuration utility window. The title bar reads "PCI". Below the title bar are icons for file operations and a "Refresh" button. The main window displays the configuration space for "Bus 01, Device 00, Function 00 - Altera Corporation Controller (PCIe)".

0	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
000	72	11	01	E0	00	00	10	00	01	00	00	FF	10	00	00	00
010	00	00	00	C0	00	00	00	00	00	00	00	D0	00	00	00	00
020	00	00	00	00	00	00	00	00	00	00	00	00	72	11	01	18
030	00	00	00	00	50	00	00	00	00	00	00	00	0B	01	00	00
040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
050	05	78	84	00	00	00	00	00	00	00	00	00	00	00	00	00
060	00	00	00	00	00	00	00	00	11	78	00	00	00	00	00	00
070	00	00	00	00	00	00	00	00	01	80	03	00	00	00	00	00
080	10	00	02	00	02	80	64	00	20	28	00	00	11	60	40	01
090	40	00	11	10	00	00	04	00	00	00	00	00	00	00	00	00
0A0	00	00	00	00	1F	00	10	00	00	00	00	00	00	00	00	00
0B0	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
0F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

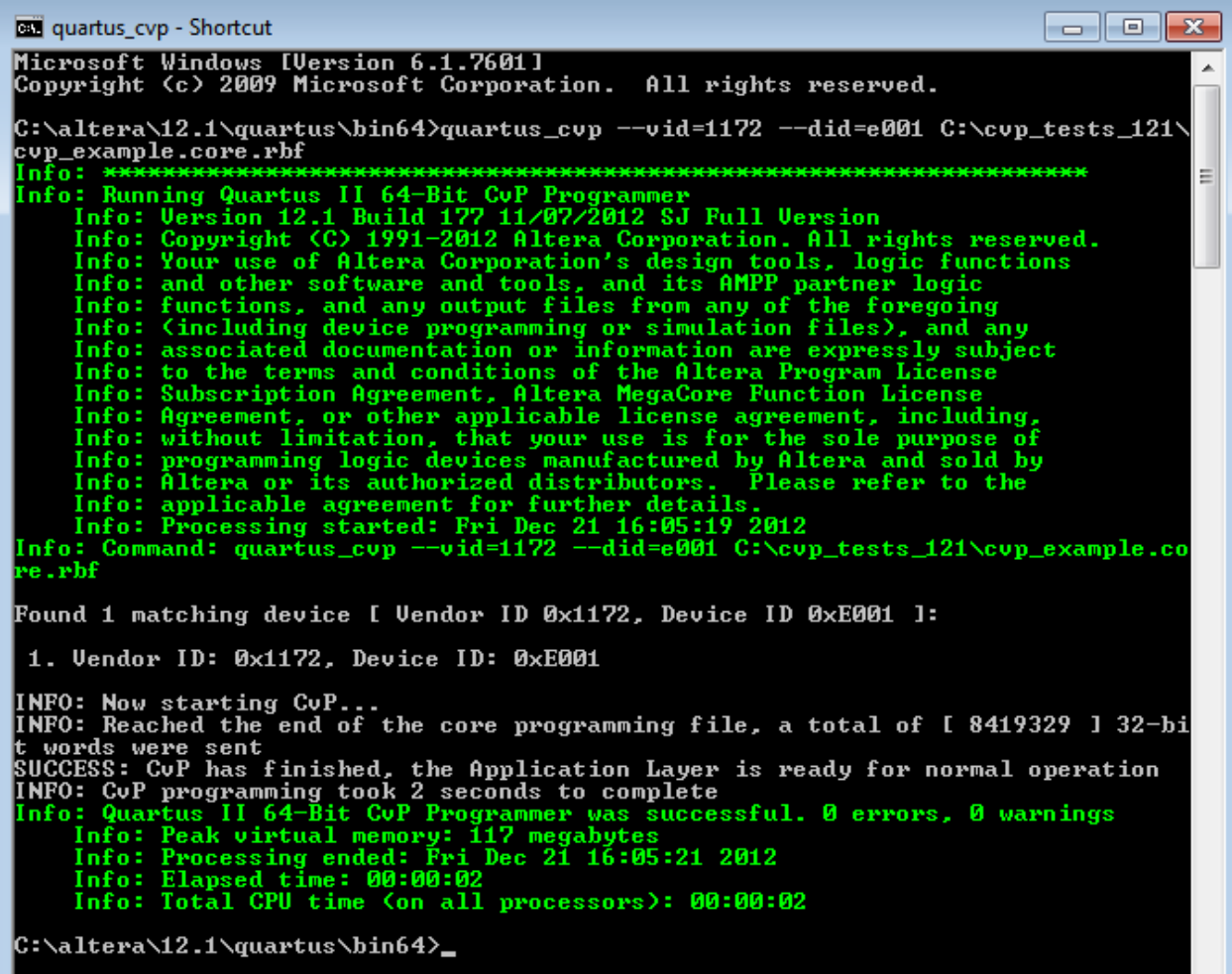
The right pane shows the following device information:

- Device/Vendor ID: 0xE001172
- Revision ID: 0x01
- Class Code: 0xFF0000
- Cacheline Size: 0x10
- Latency Timer: 0x00
- Interrupt Pin: INTA
- Interrupt Line: IRQ11
- BAR1: 0xC0000000
- BAR2: 0x00000000
- BAR3: 0xD0000000
- BAR4: 0x00000000
- BAR5: 0x00000000
- BAR6: 0x00000000
- Expansion ROM: 0x00000000
- Subsystem ID: 0x1801172

8. Follow these steps to program the core image (**top.core.rbf**) into FPGA:
 - a. Open a **DOS command** window.
 - b. Change to appropriate Quartus II **bin** install directory. Both 64-bit and 32-bit **bin** directories are available. This example uses **C:\altera\13.0\quartus\bin64**.
 - c. Type the following command to program the core image. The value of Vendor ID (vid) and Device ID (did) are in hexadecimal and must match the values you specified on the **Device Identification Registers** tab of the Stratix V Hard IP for PCI Express IP Core GUI:


```
quartus_cvp --vid=1172 --did=e001 <path>/top.core.rbf
```
 - d. The figure below shows the results of successful CvP programming.

Figure 5-26: Transcript from quartus_cvp Command



```
quartus_cvp - Shortcut
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\altera\12.1\quartus\bin64>quartus_cvp --vid=1172 --did=e001 C:\cvp_tests_121\
cvp_example.core.rbf
Info: *****
Info: Running Quartus II 64-Bit CvP Programmer
Info: Version 12.1 Build 177 11/07/2012 SJ Full Version
Info: Copyright (C) 1991-2012 Altera Corporation. All rights reserved.
Info: Your use of Altera Corporation's design tools, logic functions
Info: and other software and tools, and its AMPP partner logic
Info: functions, and any output files from any of the foregoing
Info: (including device programming or simulation files), and any
Info: associated documentation or information are expressly subject
Info: to the terms and conditions of the Altera Program License
Info: Subscription Agreement, Altera MegaCore Function License
Info: Agreement, or other applicable license agreement, including,
Info: without limitation, that your use is for the sole purpose of
Info: programming logic devices manufactured by Altera and sold by
Info: Altera or its authorized distributors. Please refer to the
Info: applicable agreement for further details.
Info: Processing started: Fri Dec 21 16:05:19 2012
Info: Command: quartus_cvp --vid=1172 --did=e001 C:\cvp_tests_121\cvp_example.co
re.rbf

Found 1 matching device [ Vendor ID 0x1172, Device ID 0xE001 ]:

1. Vendor ID: 0x1172, Device ID: 0xE001

INFO: Now starting CvP...
INFO: Reached the end of the core programming file, a total of [ 8419329 ] 32-bit
words were sent
SUCCESS: CvP has finished, the Application Layer is ready for normal operation
INFO: CvP programming took 2 seconds to complete
Info: Quartus II 64-Bit CvP Programmer was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 117 megabytes
Info: Processing ended: Fri Dec 21 16:05:21 2012
Info: Elapsed time: 00:00:02
Info: Total CPU time (on all processors): 00:00:02

C:\altera\12.1\quartus\bin64>_
```

If you implement your own software driver to program the core image, refer to the **CvP Driver Support** section in Chapter 6 of CvP User Guide.

You are now ready to run your own tests.

Related Information

[CvP Driver Support](#) on page 6-1

CvP Debugging Check List

1. Check the PCIe configuration to ensure that it supports CvP. For instance, Stratix V does not support Gen3 CvP. Gen1 x2 and Gen2 CvP are currently not available.
2. Confirm that the current Quartus II software version supports CvP.
3. Check physical pin assignments support CvP. For Example, for Stratix V, only the bottom left Stratix V Hard IP for PCI Express supports CvP. Other Hard IP cores in the same device do not support CvP.
4. Check the PCB connections for PERST# and refclk.

5. Make sure the reset and clock connections to the Transceiver Reconfiguration Controller IP Core are correct. For CvP mode, you must use `refclk` to drive `reconfig_clk`. `PERST#` must drive `reconfig_reset` to the Hard IP for PCI Express IP Core.
6. Confirm that the design meets timing constraints for setup, hold time, and recovery for multi-corners.
7. Check that `test_in` bus is hardwired to 0xA8. The following `test_in` signals are most important when debugging:
 - a. Setting `test_in[7]=1` disables support for the lower power state.
 - b. Setting `test_in[5]=1` prevents the core from entering the Compliance Mode.
 - c. Setting `test_in[3]=1` indicates that the Hard IP for PCI Express is implemented in an FPGA.
8. Disable power management support in the host BIOS settings.
9. If CvP fails, try a similar Altera CvP example design to determine if the symptoms remain the same. If the failure still persists, try a non-CvP design and take note of the differences.
10. Determine if the example CvP design with similar configuration works on the same platform.
11. Confirm that the Vendor ID and Device ID arguments specified as arguments to the `quartus_cvp.exe` command match the values specified in the Stratix V Hard IP for PCI Express IP Core GUI.
12. If you are designing an open system, test with different PCs and compare the results.
13. If the first CvP update fails, check that the correct `quartus_cvp.exe` command is used. For 32-bit systems, you must use `.\quartus\bin\quartus_cvp.exe`. For 64-bit systems, the correct `quartus_cvp` version is `.\quartus\bin64`.
14. Before executing the `quartus_cvp` command, make sure that the `Memory Space Enable` bit is set in the `Command` register of PCI Express Configuration Space. If the BIOS does not enable this bit, you must use a system tool such as RW Utilities to write a value of 0x0006 to the `Command` register at offset 0x4. Writing this value will set both `Memory Space Enable` and `Master Bus Enable` bits.
15. If encryption or compression is enabled, disable them and retry. Record the symptoms.
16. Check if the design works after configuring the FPGA with the SOF via JTAG and then doing a warm reboot. The `.sof` file contains both periphery image and the core image; consequently, this test not determine which type of image causes the failure.
17. Use a PCI Express analyzer to capture the PCIe trace of the failing scenarios. Observe the transitions of LTSSM if it fails to get insight into the link failures.
18. Use the Power On Trigger of the SignalTap II Embedded Logic Analyzer and record the LTSSM transitions. Determine whether LTSSM goes to L0 (0xF) or toggles between Detect states and Polling states.
19. Disable the Transceiver Reconfiguration Controller and hardwire other inputs as described in to zero, except the `reconfig_to_xcvr()` bus which requires to drive high bit[44] of each channel. The remaining bit of `reconfig_to_xcvr()` are tied to low. The sample file is `top_wo_reconfig.v` under `./altera_pcie_cvp/hw_devkit_ed` directory.
20. If you have tried all these suggestions and your design is still not working, file a Service Request (SR). In your SR, include the following information:
 - a. Describe what you have tried and the results of your tests.
 - b. List the Quartus II software version, the target device, information about the system under test, and what CvP modes are being used.
 - c. Specify where the failure occurs. Does it occur after loading the periphery, on the first CvP update, or on subsequent CvP updates?
 - d. If possible, attach your design so that we can review the reset and clock connections and try to replicate your failure.
 - e. Describe the steps necessary to run your design.
21. For CvP subsequent update, you must compile both revisions for any changes to any of the following logic:

- a. The periphery logic
- b. The I/O ports or core wrapper
- c. The Quartus II software version

Related Information

- [Stratix V PCI Express User Guide](#)
- [Stratix V GX High Development Board](#)
- [Stratix V Device Handbook for MSEL\[4:0\] settings](#)

Known Issues and Solutions

1. CvP designs with Gen1 x2 configurations fail to link up after loading the periphery image. One way to work around this issue is to use Gen1 x4 configuration and let the link downtrain to Gen1 x2.
2. CvP update stress test might fail if the Transceiver Reconfiguration Controller IP Core is instantiated in the design. Refer to [Workaround for a Known Issue with Transceiver Reconfiguration Controller IP Core](#) on page 5-13 for more a workaround.

2013.11.04

UG-01101



Subscribe



Send Feedback

CvP Driver Support

You can develop your own custom CvP driver for Linux using the sample Linux driver source code provided by Altera. The sample driver is written in C and can be downloaded from the Configuration via Protocol webpage.

You can also develop your own CvP driver using the Jungo WinDriver tool. You need to purchase a WinDriver license for this purpose.

Related Information

[Configuration via Protocol](#)

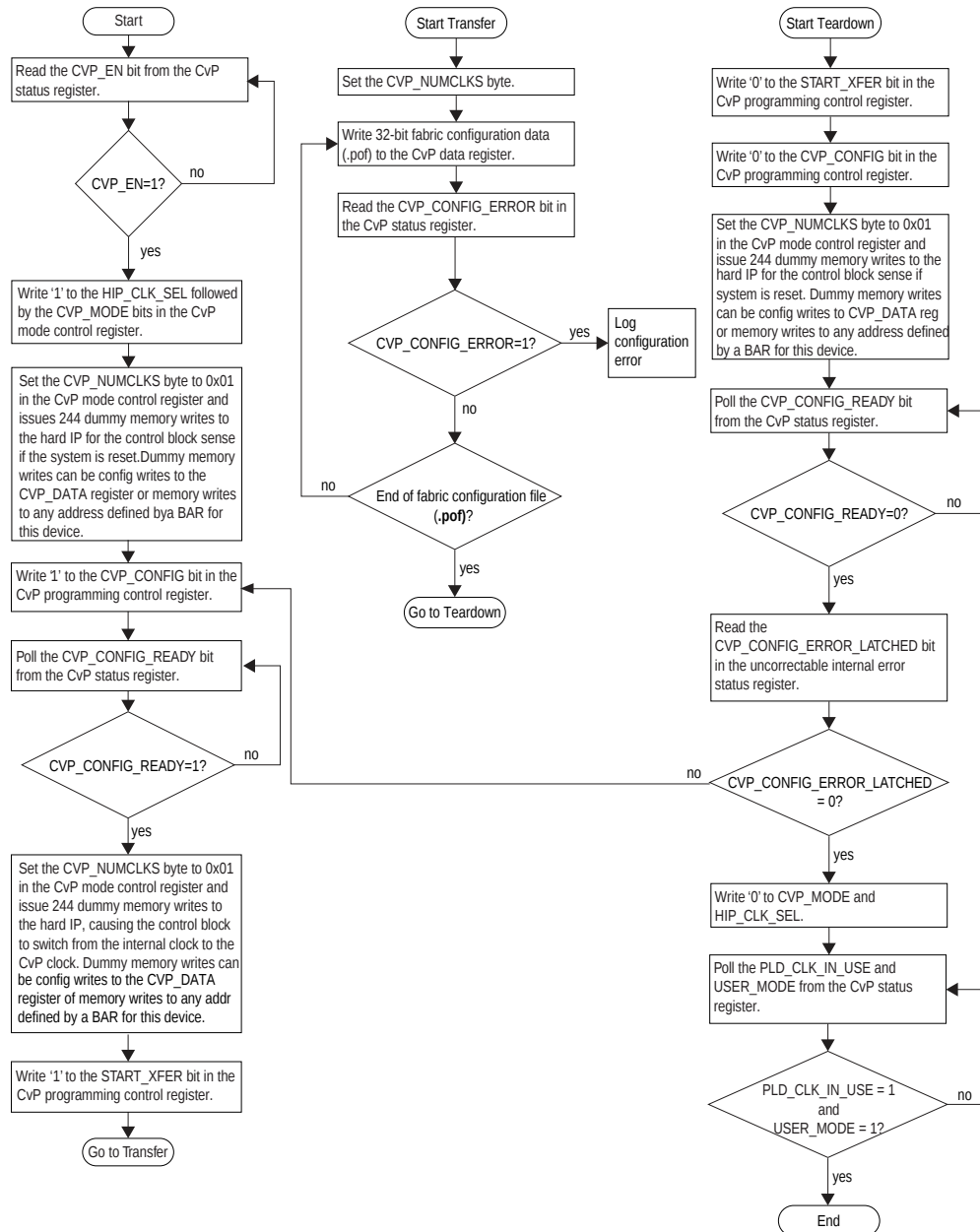
Provides more information about sample Linux device source code.

CvP Driver Flow

The following figure shows the flow of the provided CvP driver. The flow assumes that the FPGA is powered up and the control block has already configured the FPGA with the periphery image, which is indicated by the CVP_EN bit in the CvP status register.

As this figure indicates, the third step of the Start Teardown Flow requires 244 dummy configuration writes to the CVP_DATA register or 244 memory writes to an address defined by a memory space BAR for this device. Memory writes are preferred because they are higher throughput than configuration writes. The dummy writes cause a 2 ms delay, allowing the control block to complete required operations.

Figure 6-1: CvP Driver Flow



VSEC Registers for CvP

The Vendor Specific Extended Capability (VSEC) registers occupy byte offset 0x200 to 0x240 in the PCIe Configuration Space. The PCIe host uses these registers to communicate with the FPGA control block. The following table shows the VSEC register map. Subsequent tables provide the fields and descriptions of each register.

Table 6-1: VSEC Registers for CvP

Byte Offset	Register Name
0x200	Altera-defined Vendor Specific Capability Header
0x204	Altera-defined Vendor Specific Header
0x208	Altera Marker
0x20C:0x218	Reserved
0x21C	CvP Status
0x220	CvP Mode Control
0x224	CvP Data 2
0x228	CvP Data
0x22C	CvP Programming Control
0x230	Reserved
0x234	Uncorrectable Internal Error Status Register
0x238	Uncorrectable Internal Error Mask Register
0x23C	Correctable Error Status Register
0x240	Correctable Error Mask Register

Altera-defined Vendor Specific Capability Header Register

Table 6-2: Altera-defined Vendor Specific Capability Header Register (Byte Offset: 0x200)

Bits	Name	Reset Value	Access	Description
[15:0]	PCI Express Extended Capability ID	0x000B	RO	PCIe specification defined value for VSEC Capability ID.
[19:16]	Version	0x1	RO	PCIe specification defined value for VSEC version.
[31:20]	Next Capability Offset	Variable	RO	Starting address of the next Capability Structure implemented, if any.

Altera-defined Vendor Specific Header Register

Table 6-3: Altera-defined Vendor Specific Header Register (Byte Offset: 0x204)

Bits	Name	Reset Value	Access	Description
[15:0]	VSEC ID	0x1172	RO	A user configurable VSEC ID.
[19:16]	VSEC Revision	0	RO	A user configurable VSEC revision.
[31:20]	VSEC Length	0x044	RO	Total length of this structure in bytes.

Altera Marker Register

Table 6-4: Altera Marker Register (Byte Offset: 0x208)

Bits	Name	Reset Value	Access	Description
[31:0]	Altera Marker	Device Value	RO	An additional marker. If you use the standard Altera Programmer software to configure the device with CvP, this marker provides a value that the programming software reads to ensure that it is operating with the correct VSEC.

CvP Status Register

Table 6-5: CvP Status Register (Byte Offset: 0x21C)

Bits	Name	Reset Value	Access	Description
[31:26]	—	0x00	RO	Reserved.
[25]	PLD_CORE_READY	Variable	RO	From FPGA fabric. This status bit is provided for debug.
[24]	PLD_CLK_IN_USE	Variable	RO	From clock switch module to fabric. This status bit is provided for debug.
[23]	CVP_CONFIG_DONE	Variable	RO	Indicates that the FPGA control block has completed the device configuration via CvP and there were no errors.
[22]	—	Variable	RO	Reserved.
[21]	USERMODE	Variable	RO	Indicates if the configurable FPGA fabric is in user mode.
[20]	CVP_EN	Variable	RO	Indicates if the FPGA control block has enabled CvP mode.
[19]	CVP_CONFIG_ERROR	Variable	RO	Reflects the value of this signal from the FPGA control block, checked by software to determine if there was an error during configuration.
[18]	CVP_CONFIG_READY	Variable	RO	Reflects the value of this signal from the FPGA control block, checked by software during programming algorithm.
[17:0]	—	Variable	RO	Reserved.

CvP Mode Control Register

Table 6-6: CvP Mode Control Register (Byte Offset: 0x220)

Bits	Name	Reset Value	Access	Description
[31:16]	—	0x0000	RO	Reserved.
[15:14]	—	0x0	RO	Reserved.
[13:8]	CVP_NUMCLKS	0x00	RW	<p>This is the number of clocks to send for every CvP data write.</p> <p>Set this field to one of the values below depending on your configuration image:</p> <ul style="list-style-type: none"> • 0x01 for uncompressed and unencrypted images • 0x04 for uncompressed and encrypted images • 0x08 for all compressed images
[7:3]	—	0x0	RO	Reserved.
[2]	CVP_FULLCONFIG	1'b0	RO	A value of 1 indicates a request to the control block to reconfigure the entire FPGA including the Hard IP for PCI Express and bring the PCIe link down.
[1]	HIP_CLK_SEL	1'b0	RO	<p>Selects between PMA and fabric clock when USER_MODE = 1 and PLD_CORE_READY = 1. The following encodings are defined:</p> <ul style="list-style-type: none"> • 1: Selects internal clock from PMA which is required for CVP_MODE. • 0: Selects the clock from soft logic fabric. This setting should only be used when the fabric is configured in USER_MODE with a configuration file that connects the correct clock. <p>To ensure that there is no clock switching during CvP, you should only change this value when the Hard IP for PCI Express has been idle for 10 μs and wait 10 μs after changing this value before resuming activity.</p>

Bits	Name	Reset Value	Access	Description
[0]	CVP_MODE	1'b0	RO	<p>Controls whether the Hard IP for PCI Express IP Core is in CVP_MODE or normal mode. The following encodings are defined:</p> <ul style="list-style-type: none"> • 1: CVP_MODE is active. Signals to the FPGA control block active and all TLPs are routed to the Configuration Space. This CVP_MODE cannot be enabled if CVP_EN = 0. • 0: The IP core is in normal mode and TLPs are route to the FPGA fabric.

CvP Data Registers

Table 6-7: CvP Data Register (Byte Offsets: 0x224 - 0x228)

This table defines the CvP Data register.

Bits	Name	Reset Value	Access	Description
[31:0]	CVP_DATA2	0x00000000	RW	<p>Contains the upper 32 bits of a 64-bit configuration data. Software must ensure that all Bytes in both dwords are enabled. Use of 64-bit configuration data is optional.</p>

Bits	Name	Reset Value	Access	Description
[31:0]	CVP_DATA	0x00000000	RW	<p>Write the configuration data to this register. The data is transferred to the FPGA control block to configure the device.</p> <p>Every write to this register sets the data output to the FPGA control block and generates $\langle n \rangle$ clock cycles to the FPGA control block as specified by the CVP_NUM_CLKS field in the CvP Mode Control register. Software must ensure that all bytes in the memory write dword are enabled.</p> <p>You can access this register using configuration writes. Alternatively, when in CvP mode, this register can also be written by a memory write to any address defined by a memory space BAR for this device. Using memory writes are higher throughput than configuration writes.</p>

CvP Programming Control Register

Table 6-8: CvP Programming Control Register (Byte Offset: 0x22C)

Bits	Name	Reset Value	Access	Description
[31:2]	—	0x0000	RO	Reserved.
[1]	START_XFER	1'b0	RW	Sets the CvP output to the FPGA control block indicating the start of a transfer.
[0]	CVP_CONFIG	1'b0	RW	When set to 1, the FPGA control block begins a transfer via CvP.

Uncorrectable Internal Error Status Register

This register reports the status of the internally checked errors that are uncorrectable. When specific errors are enabled by the Uncorrectable Internal Error Mask register, they are handled as Uncorrectable Internal Errors as defined in the *PCI Express Base Specification 3.0*. This register is for debug only. Use this register to observe behavior, not to drive custom logic.

Table 6-9: Uncorrectable Internal Error Status Register (Byte Offset: 0x234)

Bits	Reset Value	Access	Description
[31:12]	0x00	RO	Reserved.
[11]	1'b0	RW1CS	A value of 1 indicates an RX buffer overflow condition in a posted request or Completion segment.
[10]	1'b0	RW1CS	A value of 1 indicates a parity error was detected on the R2CSEB interface.
[9]	1'b0	RW1CS	A value of 1 indicates a parity error was detected on the Configuration Space to TX bus interface.
[8]	1'b0	RW1CS	A value of 1 indicates a parity error was detected on the TX to Configuration Space bus interface.
[7]	1'b0	RW1CS	A value of 1 indicates a parity error was detected in a TX TLP and the TLP is not sent.
[6]	1'b0	RW1CS	A value of 1 indicates that the Application Layer has detected an uncorrectable internal error.
[5]	1'b0	RW1CS	A value of 1 indicates a configuration error has been detected in CvP mode which is reported as uncorrectable. This bit is set whenever a CVP_CONFIG_ERROR is asserted while in CVP_MODE.
[4]	1'b0	RW1CS	A value of 1 indicates a parity error was detected by the TX Data Link Layer.
[3]	1'b0	RW1CS	A value of 1 indicates a parity error has been detected on the RX to Configuration Space bus interface.
[2]	1'b0	RW1CS	A value of 1 indicates a parity error was detected at input to the RX Buffer.
[1]	1'b0	RW1CS	A value of 1 indicates a retry buffer uncorrectable ECC error.
[0]	1'b0	RW1CS	A value of 1 indicates a RX buffer uncorrectable ECC error.

Uncorrectable Internal Error Mask Register

This register controls which errors are forwarded as internal uncorrectable errors. With the exception of the configuration errors detected in CvP mode, all of the errors are severe and may place the device or PCIe link in an inconsistent state. The configuration error detected in CvP mode may be correctable depending on the design of the programming software.

Table 6-10: Uncorrectable Internal Error Mask Register (Byte Offset: 0x238)

Bits	Reset Value	Access	Description
[31:12]	0x00	RO	Reserved.

Bits	Reset Value	Access	Description
[11]	1'b1	RWS	Mask for RX buffer posted and completion overflow error.
[10]	1'b1	RWS	Mask for parity error on the R2CSEB interface.
[9]	1'b1	RWS	Mask for parity error on the Configuration Space to TX bus interface.
[8]	1'b1	RWS	Mask for parity error on the TX to Configuration Space bus interface.
[7]	1'b1	RWS	Mask for parity error in the transaction layer packet.
[6]	1'b1	RWS	Mask for parity error in the application layer.
[5]	1'b0	RWS	Mask for configuration error in CvP mode.
[4]	1'b1	RWS	Mask for data parity errors detected during TX Data Link LCRC generation.
[3]	1'b1	RWS	Mask for data parity errors detected on the RX to Configuration Space Bus interface.
[2]	1'b1	RWS	Mask for data parity error detected at the input to the RX Buffer.
[1]	1'b1	RWS	Mask for the retry buffer uncorrectable ECC error.
[0]	1'b1	RWS	Mask for the RX buffer uncorrectable ECC error.

Correctable Internal Error Status Register

This register reports the status of the internally checked errors that are correctable. When these specific errors are enabled by the `Correctable Internal Error Mask` register, they are forwarded as Correctable Internal Errors as defined in the *PCI Express Base Specification 3.0*. This register is for debug only. Use this register to observe behavior, not to drive custom logic.

Table 6-11: Correctable Internal Error Status Register (Byte Offset: 0x23C)

Bits	Reset Value	Access	Description
[31:7]	0x000	RO	Reserved.
[6]	1'b0	RW1CS	A value of 1 indicates that the Application Layer has detected a correctable internal error.
[5]	1'b0	RW1CS	A value of 1 indicates a configuration error has been detected in CvP mode, which is reported as correctable. This bit is set whenever a <code>CVP_CONFIG_ERROR</code> occurs while in <code>CVP_MODE</code> .
[4:2]	0x0	RO	Reserved.
[1]	1'b0	RW1CS	A value of 1 indicates a retry buffer correctable ECC error.

Bits	Reset Value	Access	Description
[0]	1'b0	RW1CS	A value of 1 indicates an RX buffer correctable ECC error.

Correctable Internal Error Mask Register

This register controls which errors are forwarded as Internal Correctable Errors. This register is for debug only.

Table 6-12: Correctable Internal Error Mask Register (Byte Offset: 0x240)

Bits	Reset Value	Access	Description
[31:7]	0x000	RO	Reserved.
[6]	1'b0	RWS	Mask for corrected internal error reported by the Application Layer.
[5]	1'b0	RWS	Mask for configuration error detected in CvP mode.
[4:2]	0x0	RO	Reserved.
[1]	1'b0	RWS	Mask for retry buffer correctable ECC error.
[0]	1'b0	RWS	Mask for RX buffer correctable ECC error.

2013.11.04

UG-01101

 [Subscribe](#)
 [Send Feedback](#)

Additional information about the document and Altera.

Document Revision History

Table 7-1: Document Revision History

Date	Version	Changes
November 2013	2013.11.04	<ul style="list-style-type: none"> Added optional CVP_DATA2 register for use when configuration data is 64 bits wide. Added design constraint for input reference clock when more than one Transceiver Reconfiguration Controller connects to the transceivers on one side of the device. Corrected errors in software driver flow diagram. Added clarification that you must select either CvP Initialization mode or CvP Update mode. The two modes cannot be combined. Updated CvP Pins section. Updated CvP Example Designs chapter. Updated CvP Features section in Design Considerations chapter.
August 2013	2013.08.26	<ul style="list-style-type: none"> Added support for 64-bit data. Changed supported clock frequencies for CvP updates using encrypted data and a non-volatile key. Only 12.5 MHz is supported. Added reasons for using a compressed bitstream. Clarified process to create separate design hierarchy for periphery and core logic. Added table showing supported features in CvP Initialization and Update Mode and CvP Update Mode. Clarified use of dummy writes for CvP driver in the teardown flow.

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

ISO
9001:2008
Registered



Date	Version	Changes
May 2013	2013.05.15	<ul style="list-style-type: none"> Added CvP Example Designs. Added Partitioning a Design for the CvP Design Flow section. Updated the Power Supplies Ramp-Up Time and POR, PCIe Timing Sequence in CvP Initialization Mode, and PCIe Timing Sequence in CvP Update Mode figures. Moved all links in all topics to the Related Information section for easy reference.
December 2012	1.3	<ul style="list-style-type: none"> Removed Gen 3 support from the CvP Update Mode in Table 2-1. Added Table 5-3 to include the supported clock source for encrypted configuration data in CvP mode. Added Table 6-2 to include the <code>quartus_cvp</code> command for compression and encryption modes. Added Table 6-3 to include the <code>CVP_NUMCLKS</code> settings. Updated Bit 1 and Bit 0 to reserved bits in Table 6-8. Updated the <code>CVP_NUMCLKS</code> settings in Figure 6-1.
July 2012	1.2	<ul style="list-style-type: none"> Updated "Data Compression" and Data Encryption" sections. Moved Table 5-6. Uncompressed <code>.rbf</code> sizes to Stratix V, Arria V, and Cyclone V <i>Configuration, Design Security, and Remote System Upgrades</i> chapters. Added Jungo WinDriver and the Linux-based Plain Text C CvP driver in "Software Support" chapter. Updated <code>CVP_NUMCLKS</code> byte settings in Table 6-8 and Figure 6-1.
January 2012	1.1	<ul style="list-style-type: none"> Added Arria V and Cyclone V devices. Removed references to the CvP Off mode. Updated Chapter 6, Software Support with PCIe driver and CvP programming information. Added "Generating Periphery Image and Fabric Image" and "VSEC for CvP" sections.
May 2011	1.0	Initial release.

How to Contact Altera

Table 7-2: Altera Contact Information

Contact ⁽⁴⁾	Contact Method	Address
Technical support	Website	www.altera.com/support

⁽⁴⁾ You can also contact your local Altera sales office or sales representative.

Contact ⁽⁴⁾		Contact Method	Address
Technical training		Website	www.altera.com/training
		Email	custrain@altera.com
Product literature		Website	www.altera.com/literature
Nontechnical support	General	Email	nacomp@altera.com
	Software licensing	Email	apgcs@altera.com

Related Information

- [Support](#)
- [Training](#)
- [Documentation](#)

⁽⁴⁾ You can also contact your local Altera sales office or sales representative.