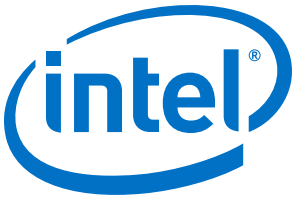


Intel® Cyclone® 10 GX CvP Initialization over PCI Express User Guide

Subscribe
Send Feedback

UG-20099 | 2018.01.02
Latest document on the web: PDF | HTML

Contents

1 CvP Initialization in Intel® Cyclone® 10 GX....3
1.1 Benefits of Using CvP....3
1.2 CvP System....3
1.3 CvP Support for Intel Cyclone 10 GX Devices....4
1.4 CvP Initialization....5
1.5 CvP Compression and Encryption Features....6
1.6 CvP Pins....6

2 Design Considerations for CvP Initialization in Intel Cyclone 10 GX....8
2.1 Designing CvP for an Open System....8
2.2 Designing CvP for a Closed System....10

3 Understanding the Design Steps for CvP Initialization in Intel Cyclone 10 GX....11
3.1 Generating the Synthesis HDL Files for Intel Cyclone 10 GX PCI Express IP Core....13
3.2 Setting up the CvP Parameters in Device and Pin Options....13
3.3 Compiling the Design....16
3.4 Splitting the SOF File....16
3.5 Bringing up the Hardware....17
3.5.1 Installing Jungo WinDriver in Windows Systems....18
3.5.2 Installing Jungo WinDriver in Linux Systems....18
3.5.3 Modifying MSEL/DIP switch on Intel Cyclone 10 GX Dev-Kit....18
3.5.4 Programming CvP Images....19

4 CvP Driver and Registers....21
4.1 CvP Driver Support....21
4.2 CvP Driver Flow....21
4.3 VSEC Registers for CvP....23
4.3.1 Intel-defined Vendor Specific Capability Header Register....23
4.3.2 Intel-defined Vendor Specific Header Register....23
4.3.3 Intel Marker Register....24
4.3.4 CvP Status Register....24
4.3.5 CvP Mode Control Register....24
4.3.6 CvP Data Registers....26
4.3.7 CvP Programming Control Register....26
4.3.8 Uncorrectable Internal Error Status Register....27
4.3.9 Uncorrectable Internal Error Mask Register....28
4.3.10 Correctable Internal Error Status Register....28
4.3.11 Correctable Internal Error Mask Register....29

A Document Revision History for Intel Cyclone 10 GX CvP Initialization over PCI Express User Guide....30

1 CvP Initialization in Intel® Cyclone® 10 GX

Configuration via Protocol (CvP) is a configuration scheme supported in Arria® V, Cyclone® V, Stratix® V, Intel® Arria 10, Intel Stratix 10, and Intel Cyclone 10 GX device families. The CvP configuration scheme creates separate images for the periphery and core logic. You can store the periphery image in a local configuration device and the core image in the host memory, reducing system costs and increasing the security for the proprietary core image. CvP configures the FPGA fabric through the PCI Express* (PCIe) link and it is available for Endpoint variants only. CvP allows the PCIe endpoint to wake up within 200 ms.

Related Links

- Introduction to Intel FPGA IP Cores
  Provides general information about all Intel FPGA IP cores, including parameterizing, generating, upgrading, and simulating IP.
- Project Management Best Practices
  Guidelines for efficient management and portability of your project and IP files.

1.1 Benefits of Using CvP

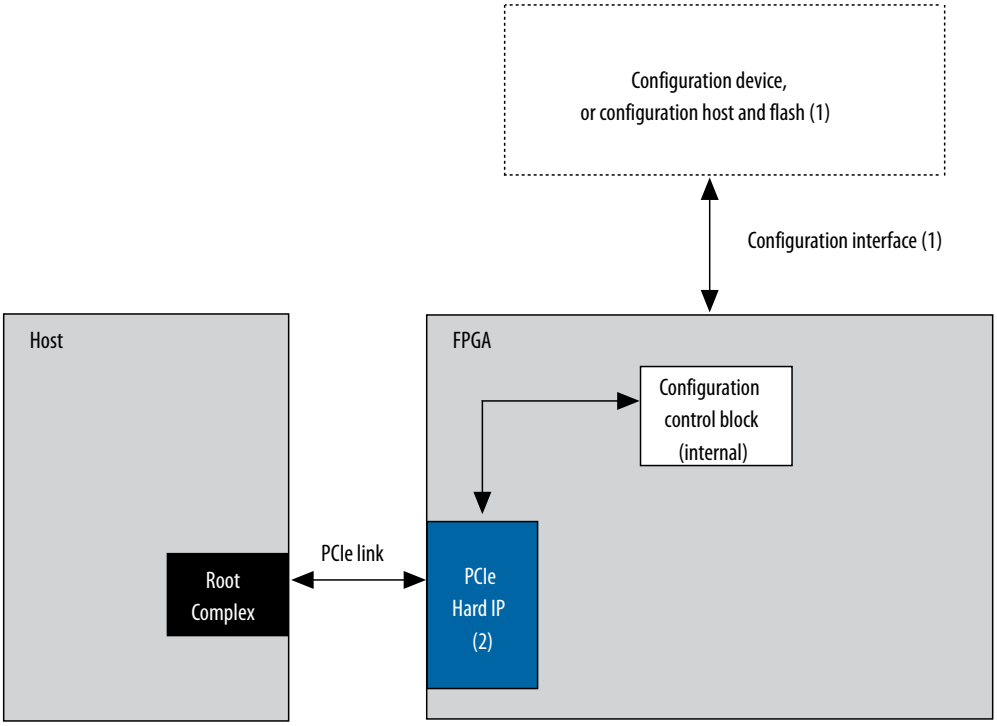
The CvP configuration scheme has the following advantages:

- Reduces system costs by reducing the size of the local flash device used to store the periphery configuration data.
- Improves security for the proprietary core bitstream. CvP ensures that the PCIe host can exclusively access the FPGA core image.
- 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 adaptable hardware acceleration.

1.2 CvP System

The following figure shows the required components for a CvP system.

Figure 1. CvP Block Diagram



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

1. 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.
2. PCIe Hard IP block for CvP and other PCIe applications.

It must be configured as an Endpoint.

1.3 CvP Support for Intel Cyclone 10 GX Devices

Table 1. CvP Support for Intel Cyclone 10 GX Devices

Device	CvP Modes Supported	
	PCIe Gen 1	PCIe Gen 2
Intel Cyclone 10 GX	CvP initialization	CvP initialization

Related Links

- Configuration via Protocol (CvP) Implementation in V-series FPGA Devices User Guide
  Provides more information about the CvP support for Stratix V, Arria V, and Cyclone V device families.

- Intel Stratix 10 Configuration via Protocol (CvP) Implementation User Guide
  Provides more information about the CvP implementation in Intel Stratix 10 FPGA devices.
- Intel Arria 10 CvP Initialization and Partial Reconfiguration over PCI Express User Guide
  Provides more information about the CvP Implementation in Intel Arria 10 devices.

1.4 CvP Initialization

This scheme configures the core of the FPGA through the PCIe link upon system power up. Initialization refers to the initial fabric configuration image loaded in the FPGA fabric after power up.

Configuration Images

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

You use the Intel Quartus® Prime software to generate the following 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 JTAG interface, PR block, CRC block, Oscillator block, Impedance control block, Chip ID, ASMI block, Remote update block, Temperature sensor, and 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 periphery image is stored in an external configuration device and is loaded into the FPGA through the conventional configuration scheme. The core image is stored in a host memory and loaded into the FPGA through the PCIe link. All other periphery and core resources are frozen until the FPGA enters user mode.

After the periphery image configuration is complete, the `CONF_DONE` signal goes high and allows the FPGA to start PCIe link training. During PCIe link training, the FPGA is not in user mode. When PCIe link training is complete, the PCIe link transitions to `L0` state and then through PCIe enumeration. 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 the `INIT_DONE` signal is enabled, 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.

1.5 CvP Compression and Encryption Features

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 Prime software. The periphery image cannot be compressed. Compressing the core image reduces the storage requirement.

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 2. 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	Yes(1)	Yes	Yes

1.6 CvP Pins

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

Table 3. CvP pin descriptions and connection guidelines

Pin Name	Pin Type	Pin Description	Pin Connection
CvP_CONFDONE	Output	The CvP_CONFDONE pin is driven low during configuration. When configuration via PCIe is complete, this signal is released and either actively driven high, or pulled high by an external pull-up resistor. 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.	If this pin is set as dedicated output, the V_{CCPGM} power supply must meet the input voltage specification of the receiving side. 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.

continued...

(1) The frequency value is 12.5 MHz

Pin Name	Pin Type	Pin Description	Pin Connection
		If you are not using the CvP modes, you can use this pin as a user I/O pin.	
INIT_DONE	Output	When you enable this pin, a transition from low to high at the pin indicates the device has entered user mode. If the INIT_DONE output is enabled, the INIT_DONE pin cannot be used as a user I/O pin after configuration. This is a dual-purpose pin and can be used as an I/O pin when not enabled as the INIT_DONE pin.	When you use the optionally open-drain output dedicated INIT_DONE pin, connect this pin to an external 10-kΩ pull-up resistor to VCCPGM. When you use this pin in an AS or PS multi-device configuration mode, ensure you enable the INIT_DONE pin in the Intel Quartus Prime Pro Edition designs. When you do not use the dedicated INIT_DONE optionally open-drain output, and when this pin is not used as an I/O pin, connect this pin as defined in the Intel Quartus Prime Pro Edition software.
CONF_DONE	Bidirectional	Dedicated configuration done pin. As a status output, the CONF_DONE pin drives low before and during configuration. After all configuration data is received without error and the initialization cycle starts, CONF_DONE is released. As a status input, the CONF_DONE pin goes high after all data is received. Then the device initializes and enters user mode. This pin is not available as a user I/O pin.	Connect an external 10-kΩ pull-up resistors to VCCPGM. VCCPGM must be high enough to meet the VIH specification of the I/O on the device and the external host. When you use passive configuration schemes, the configuration controller monitors this pin.
nPERST	Input	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 in the reset mode.	Connect the nPERST to the PERST# pin of the PCIe slot. This pin is powered by 1.8V supply and must be driven by 1.8V compatible I/O standards. The nPERST pin is used per PCIe Hard IP. This pin has the following location: • nPERST = PCIe HIP & CvP

2 Design Considerations for CvP Initialization in Intel Cyclone 10 GX

2.1 Designing CvP for an Open System

While designing a CvP system for an Open System where you don't control both ends of the PCIe link completely, ensure that you observe the guidelines provided in this section.

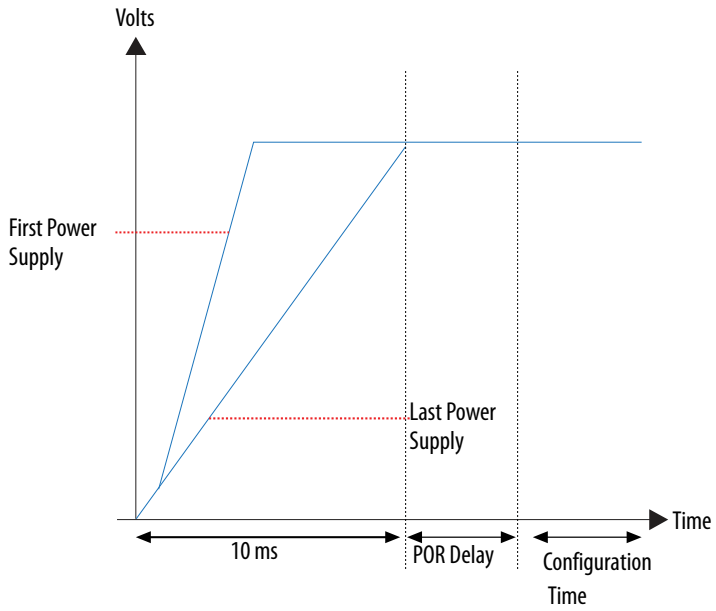
FPGA Power Supplies Ramp Time Requirement

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 in the recommended operating range. A POR event occurs from when you power up the FPGA until the power supplies reach the recommended operating range within the maximum power supply ramp time, t_{RAMP}. If t_{RAMP} is not met, the device I/O pins and programming registers remain tri-stated, during which device configuration could fail.

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 2. Power Supplies Ramp-Up Time and POR



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.

PCIe Wake-Up Time Requirement for CvP Initialization

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. The PERST# signal indicates when the FPGA power supplies are within their specified voltage tolerances and the REFCLK is stable. The embedded hard reset controller triggers after the internal status signal indicates that the periphery image has been loaded. This reset does not trigger off of PERST#. For CvP initialization mode, the PCIe link supports the FPGA core image configuration and PCIe applications in user mode.

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

Figure 3. PCIe Timing Sequence in CvP Initialization Mode

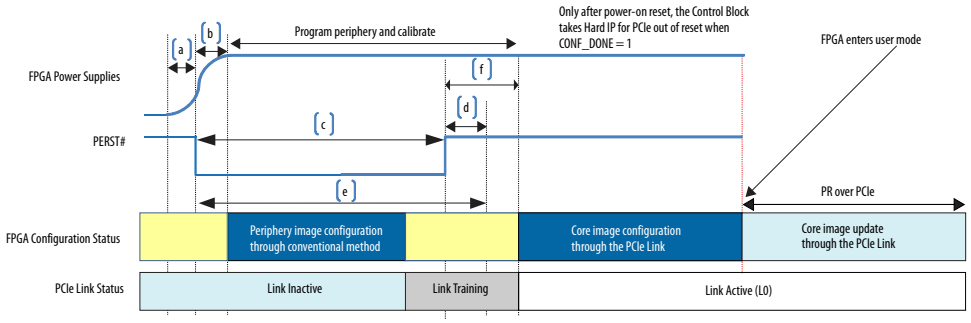


Table 4. Power-Up Sequence Timing in CvP Initialization Mode

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.

Related Links

PCI Express CARD Electromechanical Specification

2.2 Designing CvP for a Closed System

While designing CvP for a closed system where you control both ends of the PCIe link completely, estimate the periphery configuration time for CvP initialization. You must ensure that the estimated configuration time is within the time allowed by the PCIe host.

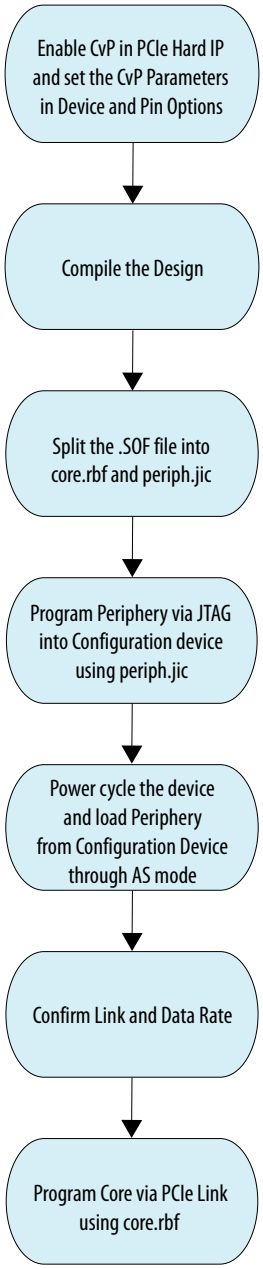
3 Understanding the Design Steps for CvP Initialization in Intel Cyclone 10 GX

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

You must specify CvP initialization mode in the Intel Quartus Prime software by selecting the CvP Settings Power up and subsequent core configuration and also turn on **Enable Configuration via Protocol (CvP)** in the **Arria 10/Cyclone 10 Hard IP for PCI Express**. You might choose CvP initialization to prevent unauthorized access to the core image as well as save cost by storing the core image in the host memory.

Figure 4. Design Flow for CvP Initialization

The following figure provides the high-level steps for CvP Initialization with Active Serial (AS) mode:



Note: For CvP initialization, you must use the CMU PLL and the Hard Reset Controller for the PCI Express Hard IP.

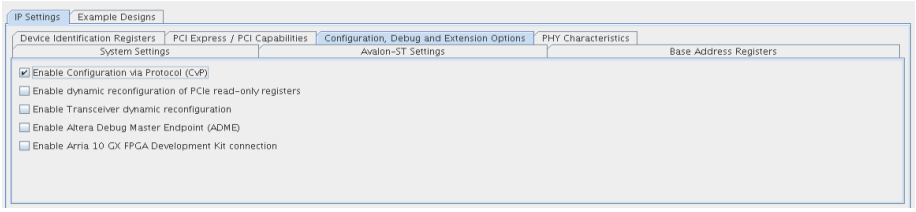
The CvP initialization demonstration walkthrough includes the steps mentioned in the following sections:

3.1 Generating the Synthesis HDL Files for Intel Cyclone 10 GX PCI Express IP Core

Follow these steps to generate the synthesis HDL files with CvP enabled:

1. Open the Intel Quartus Prime Pro Edition software.
2. On the **Tools** menu, select **Platform Designer**. The **Open System** window appears.
3. For **System**, click + and specify a **File Name** to create a new platform designer system. Click **Create**.
4. On the **System Contents** tab, delete the `clock_in` and `reset_in` components that appear by default.
5. In the **IP Catalog** locate and double-click **Arria 10/Cyclone 10 Hard IP for PCI Express**. The new window appears.
6. On the **IP Settings** tab, specify the parameters and options for your design variation.
7. Under **Configuration, Debug and Extenstion Options**, turn on **Enable Configuration via Protocol (CvP)** as shown in the following figure:

Figure 5. Illustrating the specified option in IP Settings Tab



8. Click **Finish**.
9. On the **Generation** tab, specify your parameters to generate RTL. Then click **Generate** at the bottom of the window.

Related Links

Intel Cyclone 10 GX CvP Initialization Design Example

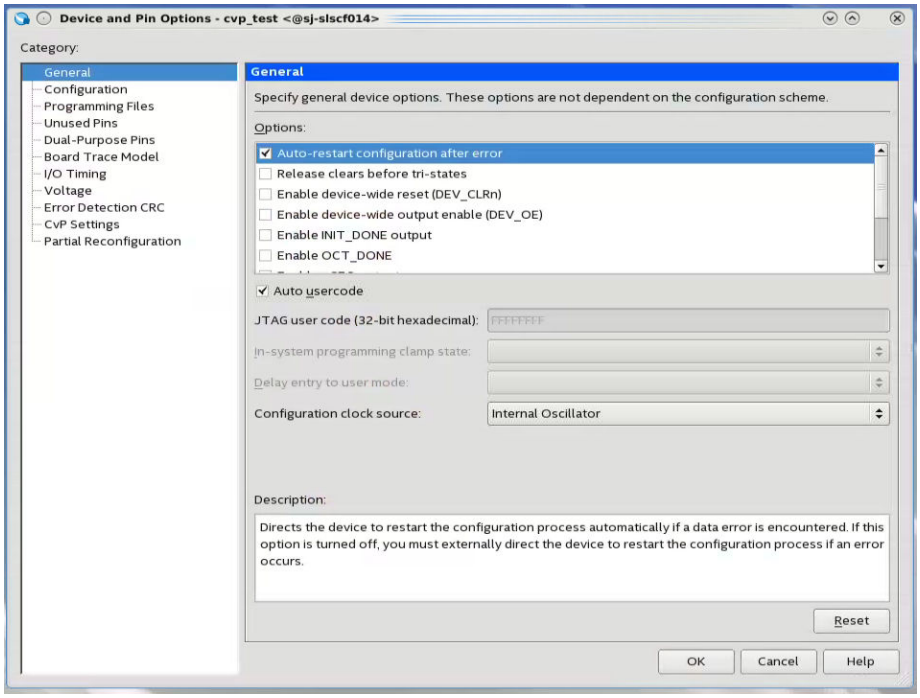
3.2 Setting up the CvP Parameters in Device and Pin Options

Follow these steps to specify CvP parameters:

1. On the Quartus Prime **Assignments** menu, select **Device**, and then click **Device and Pin Options**.
2. Under **Category**, select **General**, and then enable the following option:
   a. **Auto-restart configuration after error**, enable this option to allow automatic restart of configuration attempts if an error is detected. Any restarted configuration may exceed the required PCIe startup time to allow bus enumeration and prevent the use of quartus_cvp for core programming.

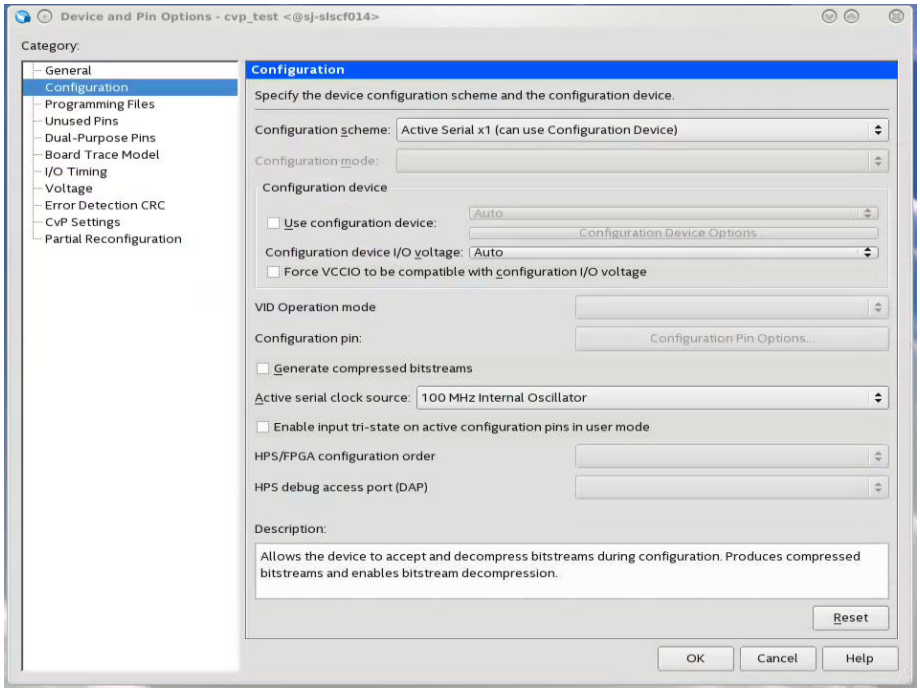
   Leave all other options disabled.

Figure 6. CvP Parameters in General Tab



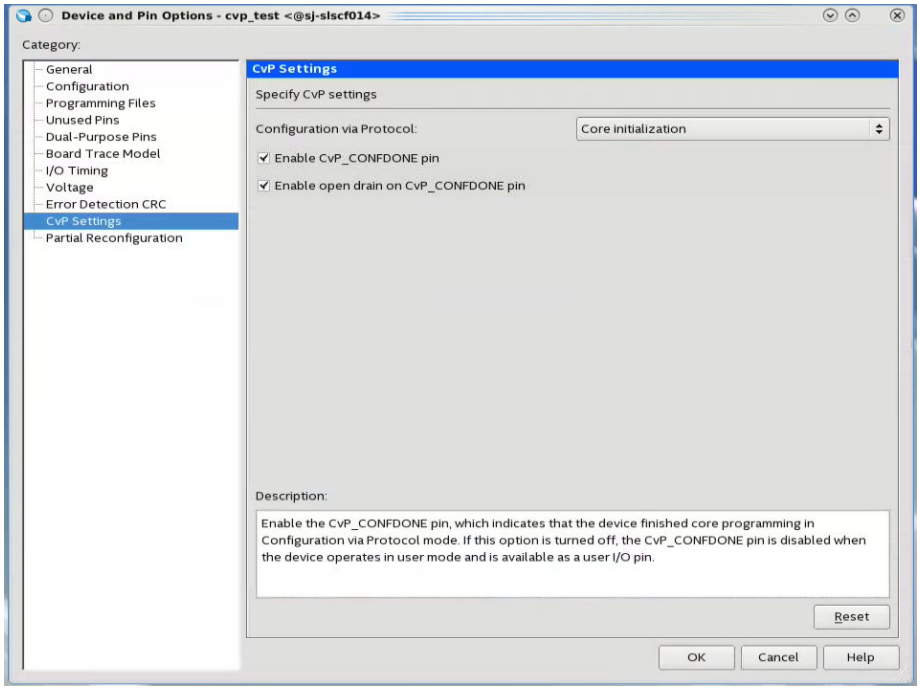
3. Under **Category**, select **Configuration**. For **Configuration scheme**, select **Active Serial x1 (can use Configuration Device) or Active Serial x4 (can use Configuration Device)**.

Figure 7. CvP Parameters in Configuration Tab



4. Under **Category**, select **CvP Settings** to specify CvP settings. For **Configuration via Protocol** select **Core Initialization** and then turn on **Enable CVP_CONFDONE pin** and **Enable open drain on CVP_CONFDONE pin**. Click **OK**.

Figure 8. CvP Parameters in CvP Settings Tab



3.3 Compiling the Design

1. To compile the design, on the **Processing** menu, select **Start Compilation** to create the `.sof` file.

3.4 Splitting the SOF File

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

1. After the `.sof` file is generated, under **File** menu, select **Convert Programming Files**.
2. Under **Output programming file** section, specify the following parameters:

Table 5. Parameters: Output Programming File Tab

Parameter	Value
Programming file type	JTAG Indirect Configuration File (*.jic)
Configuration device	EPCQL1024
Mode	Active Serial
File name	`*.periph.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**.

3. Under **Input files to convert**, specify the following parameters:

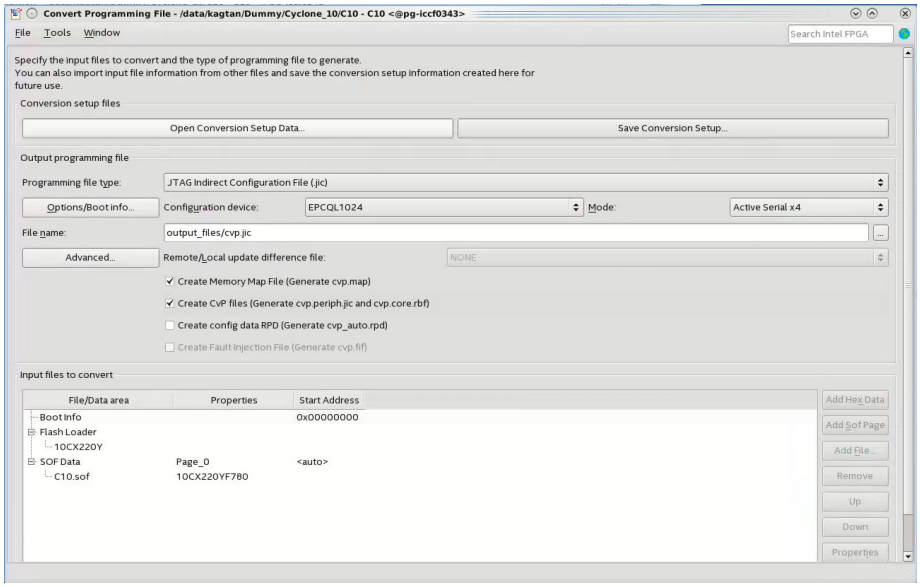
Table 6. Parameters: Input Files to Convert Tab

Parameter	Value
Flash Loader	10CX220YF780E5G
SOF Data	`*.sof`

4. Make sure to turn on the **Create CvP files**.

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

Figure 9. Illustrating the above specified options in the Covert Programming File GUI



5. Click **Generate** to create `*.periph.jic` and `*.core.rbf` files.

3.5 Bringing up the Hardware

Before testing the design in hardware, you must install Jungo WinDriver in your DUT system. 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:

- Intel Cyclone 10 GX FPGA Development Kit
- Intel FPGA Download Cable
- A DUT PC with PCI Express slot to plug in the Intel Cyclone 10 GX FPGA Development Kit
- A host PC running the Intel Quartus Prime software to program the periphery image, `.sof` or `.pof` file.

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

3.5.1 Installing Jungo WinDriver in Windows Systems

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

3.5.2 Installing Jungo WinDriver in Linux Systems

1. Navigate to <*Quartus Prime installation path*>/quartus/drivers/wdrvr/linux<*32 or 64*>(2).
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 7. Syntax for quartus_cvp Commands

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

3.5.3 Modifying MSEL/DIP switch on Intel Cyclone 10 GX Dev-Kit

The MSEL/DIP switch is labeled **S1** on the back of the Intel Cyclone 10 GX Development Kit. The MSEL [2] is hardwired to 0. Switch the MSEL [1:0] as shown in below table.

(2) You must use Linux version older than RedHat 7.2 or Kernel 3.0

Table 8. MSEL Pin Settings for Each Configuration Scheme of Intel Cyclone 10 GX Devices

- Do not drive the MSEL pins with a microprocessor or another device.

Configuration Scheme	V_{CCPGM} (V)	Power-On Reset (POR) Delay	Valid MSEL[2..0]
JTAG-based configuration	—	—	Use any valid MSEL pin settings below
AS (x1 and x4)	1.8	Fast	010
		Standard	011

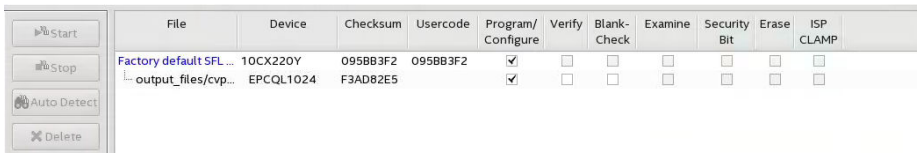
3.5.4 Programming CvP Images

You must program the periphery image (`.periph.jic`) and then download the core image (`.core.rbf`) using the PCIe Link. You can use JTAG to load different programming files (i.e. .jic/periph.pof) into your selected configuration device and configure CvP initialization enabled Intel Cyclone 10 GX device with AS configuration mode.

After loading the periphery image via the configuration device, the link should reach the expected data rate and link width. You can confirm the PCIe link status using the RW Utilities.
Follow these steps to program and test the CvP functionality:

1. Plug the Intel Cyclone 10 GX FPGA Development Kit into the PCI Express slot of the DUT PC and power it ON. Intel recommends that you use the ATX power supply that the development kit includes.
2. On the host PC, open the Intel Quartus Prime **Tools** menu and select **Programmer**.
3. Click **Auto Detect** to verify that the USB Blaster recognizes the Intel Cyclone 10 GX FPGA.
4. Follow these steps to program the periphery image:
   a. Select Intel Cyclone 10 GX device, and then right click **None** under **File** column.
   b. Navigate to `.periph.jic` file and click **Open**.
   c. Under **Program/Configure** column, select the respective devices. For example, **10CX220YF780E5G** and **EPCQL1024**.
   d. Click **Start** to program the periphery image into **EPCQL1024** flash.

Figure 10. Illustrating the specified options to the program periphery image

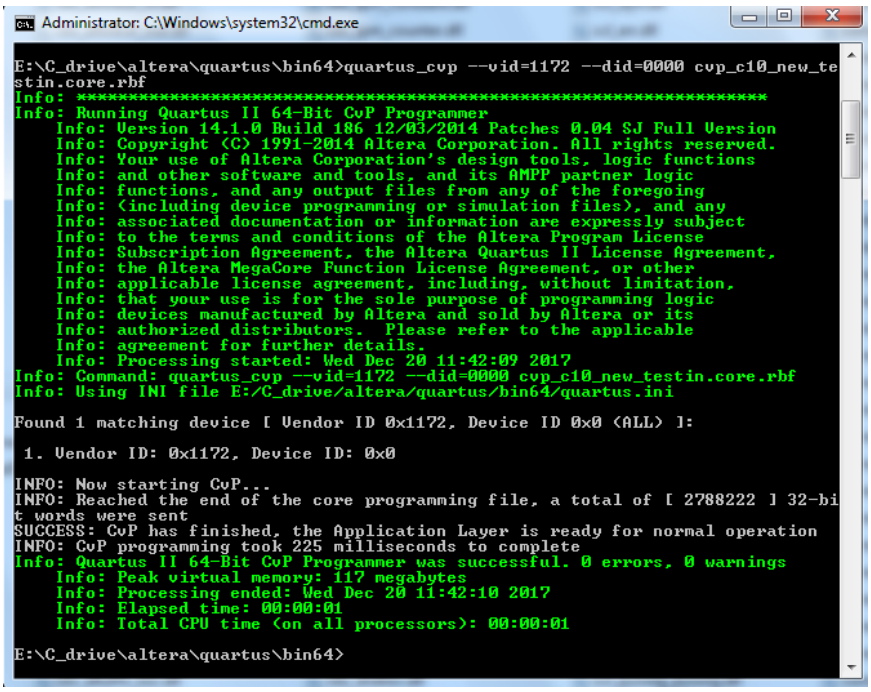


5. After the `.periph.jic` is programmed, the FPGA must be powered cycle to allow the new peripheral image to load from the on-board flash into the FPGA. To force the host PC to re-enumerate the link with the new image, power cycle the DUT PC and the Intel Cyclone 10 GX Development Kit.
6. You can use RW Utilities or another system software driver to verify the link status. You can also confirm expected link speed and width.
7. Follow these steps to program the core image:
   a. Copy the `.core.rbf` file to appropriate Intel Quartus Prime **bin** install directory. Depending on the 32-bit or 64-bit system, the folder is *…./quartus/bin32* or *…./quartus/bin64*.
   b. Open a **Command Prompt** in Windows, change the directory to the same mentioned above where the file is copied.
   c. Type the following command to program the core image:

      `quartus_cvp --vid=<Vendor ID> --did=<Device ID> xxx.core.rbf`

      where the value of Vendor ID and Device ID are in hexadecimal and specified in the Hard IP for PCI Express dialog box. For example, *quartus_cvp --vid=1172 --did=0000 xxx.core.rbf*.
   d. You can print out the kernel message using the dmesg to ensure the CvP is completed successfully.

Figure 11. Command Prompt Console



4 CvP Driver and Registers

4.1 CvP Driver Support

You can develop your own custom CvP driver for Linux using the sample Linux driver source code provided by Intel. 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.

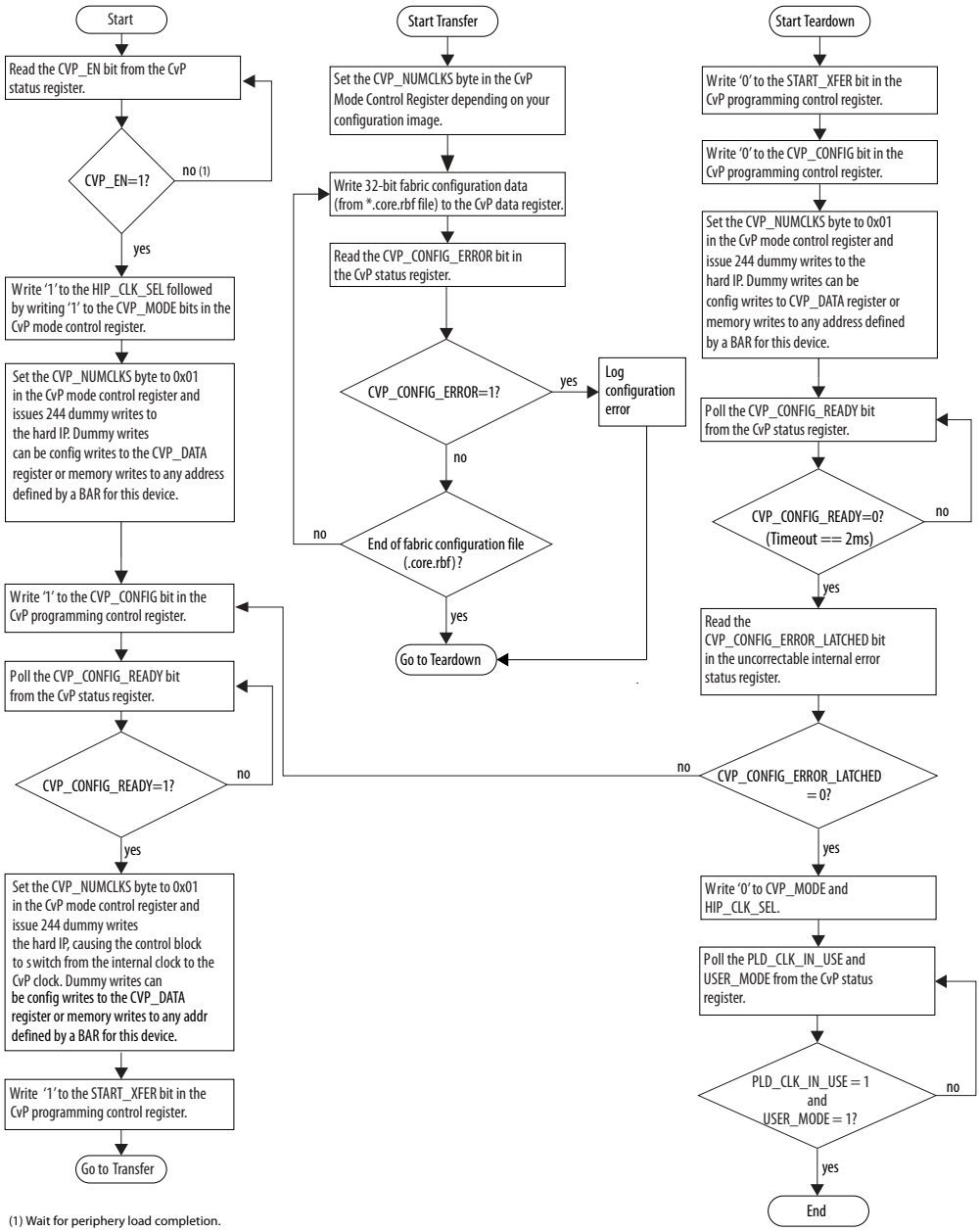
4.2 CvP Driver Flow

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 244 memory writes cause a 2 ms delay, allowing the control block to complete required operations.

Figure 12. CvP Driver Flow

For high density devices such as Intel Cyclone 10 GX, it may be necessary to wait up to 500 ms for the CvP status register bit assertion.



(1) Wait for periphery load completion.

4.3 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 9. VSEC Registers for CvP

Byte Offset	Register Name
0x200	Intel-defined Vendor Specific Capability Header
0x204	Intel-defined Vendor Specific Header
0x208	Intel 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 Internal Error Status Register
0x240	Correctable Internal Error Mask Register

4.3.1 Intel-defined Vendor Specific Capability Header Register

Table 10. Intel-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.

4.3.2 Intel-defined Vendor Specific Header Register

Table 11. Intel-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.

4.3.3 Intel Marker Register

Table 12. Intel Marker Register (Byte Offset: 0x208)

Bits	Name	Reset Value	Access	Description
[31:0]	Intel Marker	Device Value	RO	An additional marker. If you use the standard Intel 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.

4.3.4 CvP Status Register

Table 13. 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.

4.3.5 CvP Mode Control Register

Table 14. CvP Mode Control Register (Byte Offset: 0x220)

Bits	Name	Reset Value	Access	Description
[31:16]	—	0x0000	RO	Reserved.
[15:8]	CVP_NUMCLKS	0x00	RW	This is the number of clocks to send for every CvP data write. This is also known as CDRATIO (clock to data ratio).

continued...

Bits	Name	Reset Value	Access	Description
				Set this field to one of the values below depending on your configuration image: • 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	RW	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	RW	Selects between PMA and fabric clock when USER_MODE = 1 and PLD_CORE_READY = 1. The following encodings are defined: • 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. 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.
[0]	CVP_MODE	1'b0	RW	Controls whether the Hard IP for PCI Express is in CVP_MODE or normal mode. The following encodings are defined: • 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.

4.3.6 CvP Data Registers

Table 15. CvP Data Register (Byte Offsets: 0x224 - 0x228)

Bits	Name	Reset Value	Access	Description
[31:0]	CVP_DATA2	0x00000000	RW	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. Write this register first, followed by CVP_DATA register.
[31:0]	CVP_DATA	0x00000000	RW	Write the configuration data to this register. The data is transferred to the FPGA control block to configure the device. Every write to this register sets the data output to the FPGA control block and generates <*n*> 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. 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.

4.3.7 CvP Programming Control Register

Table 16. 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.

4.3.8 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 17. 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 CVP_CONFIG_ERROR_LATCHED 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.

4.3.9 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 18. Uncorrectable Internal Error Mask Register (Byte Offset: 0x238)

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

4.3.10 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 19. 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 CVP_CONFIG_ERROR occurs while in CVP_MODE.

continued...

Bits	Reset Value	Access	Description
[4:2]	0x0	RO	Reserved.
[1]	1'b0	RW1CS	A value of 1 indicates a retry buffer correctable ECC error.
[0]	1'b0	RW1CS	A value of 1 indicates an RX buffer correctable ECC error.

4.3.11 Correctable Internal Error Mask Register

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

Table 20. 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.

A Document Revision History for Intel Cyclone 10 GX CvP Initialization over PCI Express User Guide

Date	Version	Changes
January 2018	2018.01.02	Initial release.