Intel Arria 10 FPGA Remote System Update via PCI Express* Design Example User Guide

Date: Mar 2021

Revision: 2021.3.26

Contents

1. Overview

Remote update capability is one of the key advantages of FPGA. It enables deployed systems to be updated with design patches or enhanced capability without recalling. This is a prominent feature to manage remote upgrades of the application configuration images in the configuration device for PCI Express add in card that is designed using FPGA. It helps to reduce time to market and extends product life.

This reference design demonstrates remote system update functionality on Arria® 10 FPGA Development Kit using PCI Express as the communication protocol. The configuration image from the host system is received via PCI Express in the Intel® Arria 10 device and then written into the serial flash by Modular Scatter-Gather DMA.The reconfiguration process of remote update is controlled by the dedicated remote system upgrade circuitry in the Intel Arria 10 device and manage via PCI Express. When an error occurs, the circuitry detects the error, reverts to a safe configuration image, and provides error status to the design.

A smart host can use the PCI Express protocol and the application topology to update the entire FPGA dynamically without requiring a system power down or host reboot. With a simple software model for configuration, it allows quick update of design for changing application loads.

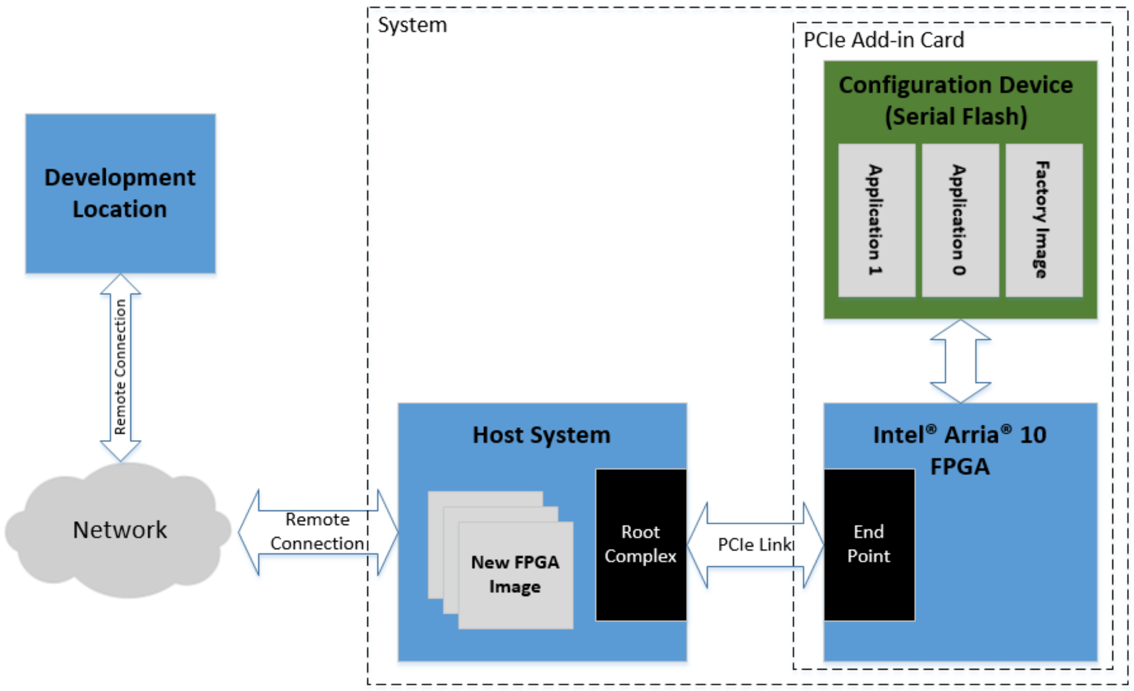


Figure 1: FPGA Design Image Deployment from Development Site to System Online via Remote System Update

2. Reference Design Hardware and Software Requirements

2.1. Hardware Requirements

1. Intel Arria 10 GX FPGA Development Kit.
2. Micro USB to USB Type A cable for configuring the FPGA device.
3. A computer with a PCIe Gen3 x8 or x16 slot running Linux Operating System. CentOS Linux 7.4 64bit was used for this reference design testing.
4. A Windows or Linux computer to run Quartus Prime Pro software and configure the Intel Arria 10 GX FPGA Development Kit.

2.2. Software Requirements

1. The Linux software driver installed on the Linux computer. The reference design is available in the Intel FPGA Design Store and the Linux software driver is included in the package. Besides, the Intel Quartus Prime Pro Edition Platform Archive File (.par) includes the recommended synthesis, fitter, and timing analysis settings for the parameters specified in the reference designs.
2. Quartus Prime Pro software, version 20.4.

Related Information
Intel Arria 10 Remote System Update via PCI Express Reference Design
Intel Quartus Prime Pro Edition Download Center
Intel Arria 10 GX FPGA Development Kit

Note: To download this reference designs, first make sure that you have access to the Intel Design Store by logging into Design Store. You can then click on the link provided above to download the design.

3. Working with the Reference Design

This reference design consists of minimal design components to enable remote system update functionality on Arria 10 FPGA Development Kit through PCI Express interface. The same design will be used to create one factory image and two application images. The difference between these images is the on-board user LED light setting. Factory image will have four red LED lights lit up when loaded, while application image 1 will have one red LED light lit up and application image 2 will have two red LED lights lit up when loaded.

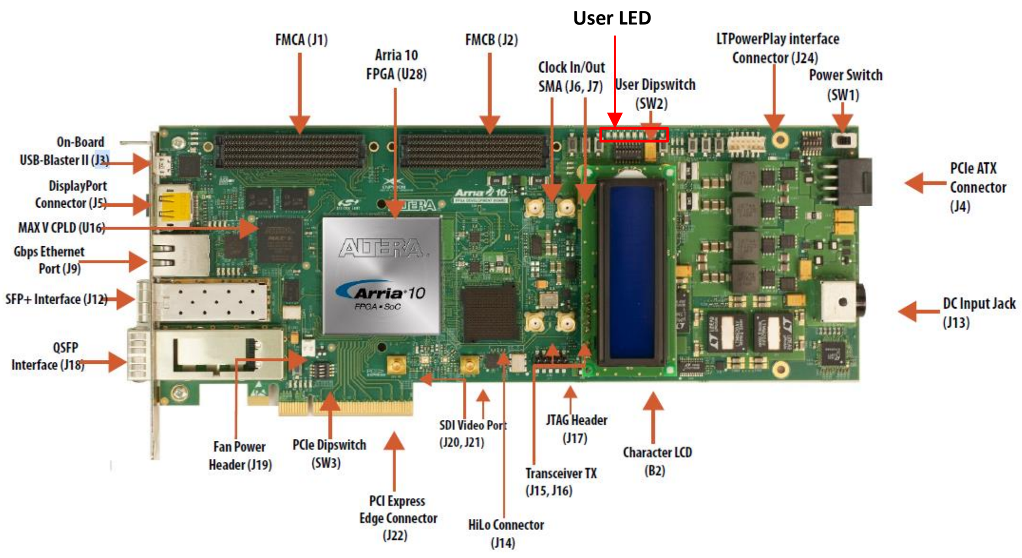


Figure 2: Arria 10 FPGA Development Kit

3.1. Reference Design Package

The reference design uses the following structure:

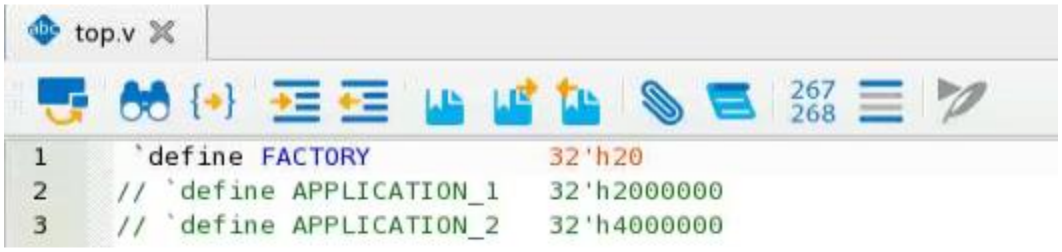
- top.v – The top-level module instantiates rsu_pcie.
- rsu_pcie.v – Platform Designer top-level files. If you modify the design (rsu_pcie.qsys) using Platform Designer, you must regenerate the system for the changes to take effect.

The reference design contains the following folders:

- driver – contains the Linux driver and software application package, rsu_over_pcie_arria10_driver.tar.gz
- master_image – constains known working image files for hardware testing
- platform – contains the IP and RTL source files

3.2. Compiling the Factory Design

1. Download the reference design project archive from Intel FPGA Design Store.
2. Launch Quartus Prime Pro software.
3. Restore the project by following the instructions on the Intel FPGA Design Store download page with the project folder named as “factory”.
4. On the **File** menu of Quartus Prime Pro software, click **Open** button, and select the top.v file in the project directory and click **Open** button.
5. At the top part of the top.v file, make sure that the FACTORY definition is uncommented, while APPLICATION_1 and APPLICATION_2 definition is commented.

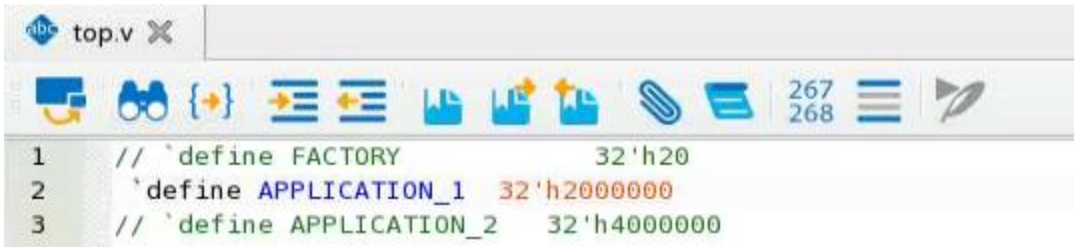


```
 `define FACTORY              32'h20
// `define APPLICATION_1      32'h2000000
// `define APPLICATION_2      32'h4000000
```

6. On the **File** menu, click **Save** to save the modification of top.v file.
7. On the **Processing** menu, click **Start Compilation** to run full compile of the design.

3.3. Compiling the Application 1 Design

1. Launch Quartus Prime Pro software.
2. Restore the project by following the instructions on the Intel FPGA Design Store download page with the project folder named as “application_1”.
3. On the **File** menu of Quartus Prime Pro software, click **Open** button, and select the top.v file in the project directory and click **Open** button.
4. At the top part of the top.v file, make sure that the APPLICATION_1 definition is uncommented, while FACTORY and APPLICATION_2 definition is commented.

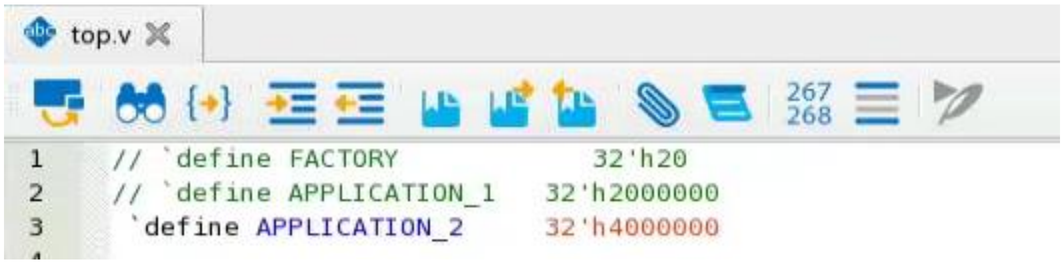


```
// `define FACTORY              32'h20
 `define APPLICATION_1  32'h2000000
// `define APPLICATION_2    32'h4000000
```

5. On the **File** menu, click **Save** to save the modification of top.v file.
6. On the **Processing** menu, click **Start Compilation** to run full compile of the design.

3.4. Compiling the Application 2 Design

1. Launch Quartus Prime Pro software.
2. Restore the project by following the instructions on the Intel FPGA Design Store download page with the project folder named as “application_2”.
3. On the **File** menu of Quartus Prime Pro software, click **Open** button, and select the top.v file in the project directory and click **Open** button.
4. At the top part of the top.v file, make sure that the APPLICATION_2 definition is uncommented, while FACTORY and APPLICATION_1 definition is commented.



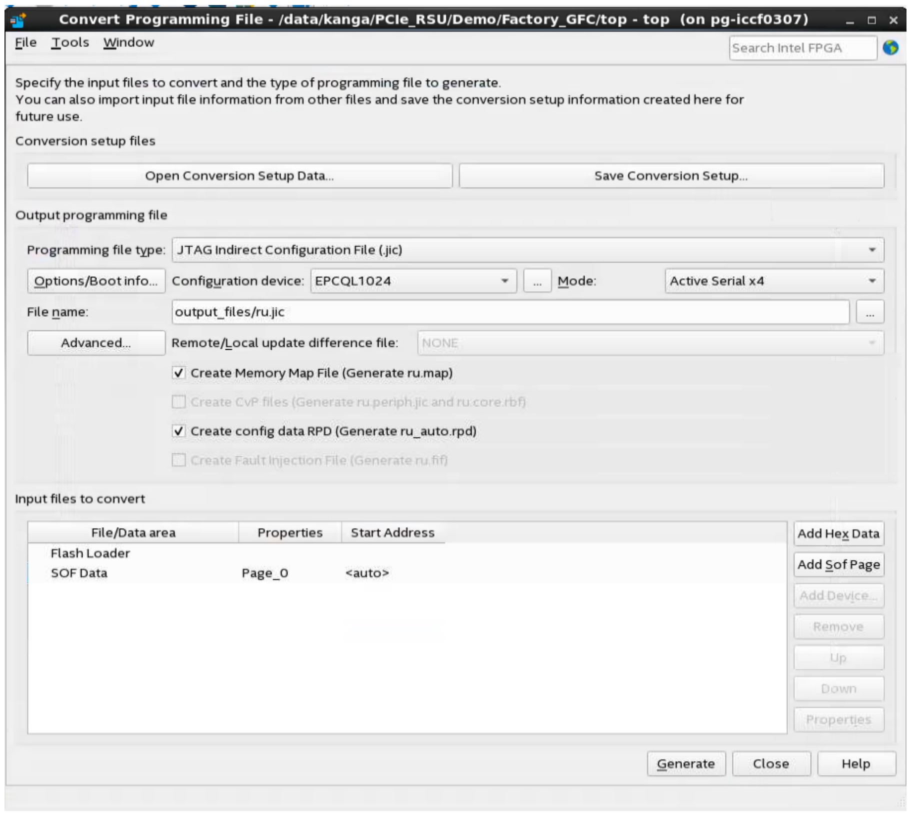
```
// `define FACTORY              32'h20
// `define APPLICATION_1    32'h2000000
 `define APPLICATION_2      32'h4000000
```

5. On the File menu, click Save to save the modification of top.v file.
6. On the Processing menu, click Start Compilation to run full compile of the design.

3.5. Creating Raw Programming Data File for Remote System Update via PCIe

Raw Programming Data file is a binary file (with the extension .rpd) containing configuration data for use outside the Quartus Prime software. Below are the steps to generate a Raw Programming Data file that contains Factory, Application 1 and Application 2 images using the Convert Programming Files Wizard.

1. Launch Quartus Prime Pro software.
2. On the **File** menu, click **Convert Programming Files…** to bring up the dialog box as shown in figure below.



Convert Programming File - /data/kanga/PCIe_RSU/Demo/Factory_GFC/top - top (on pg-iccf0307)

File Tools Window

Search Intel FPGA

Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.

Conversion setup files

Open Conversion Setup Data... | Save Conversion Setup...

Output programming file

Programming file type: JTAG Indirect Configuration File (.jic)

Options/Boot info... Configuration device: EPCQL1024 ... Mode: Active Serial x4

File name: output_files/ru.jic

Advanced... Remote/Local update difference file: NONE

✓ Create Memory Map File (Generate ru.map)

Create CvP files (Generate ru.periph.jic and ru.core.rbf)

✓ Create config data RPD (Generate ru_auto.rpd)

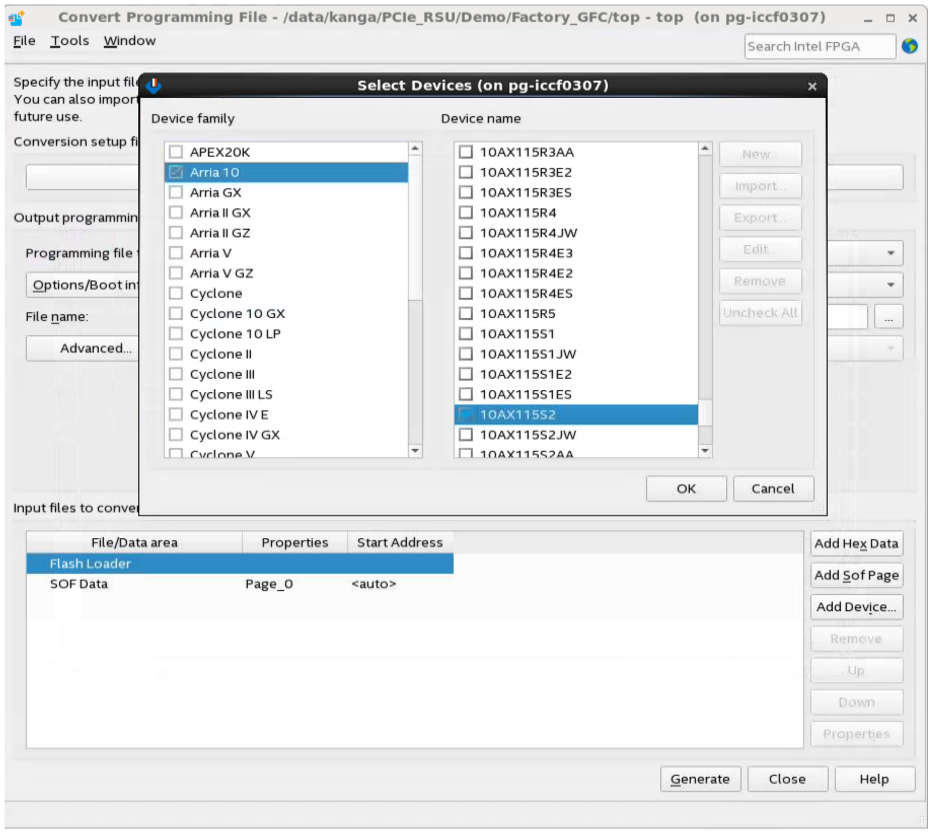
Create Fault Injection File (Generate ru.fif)

Input files to convert

File/Data area	Properties	Start Address
Flash Loader		
SOF Data	Page_0	<auto>

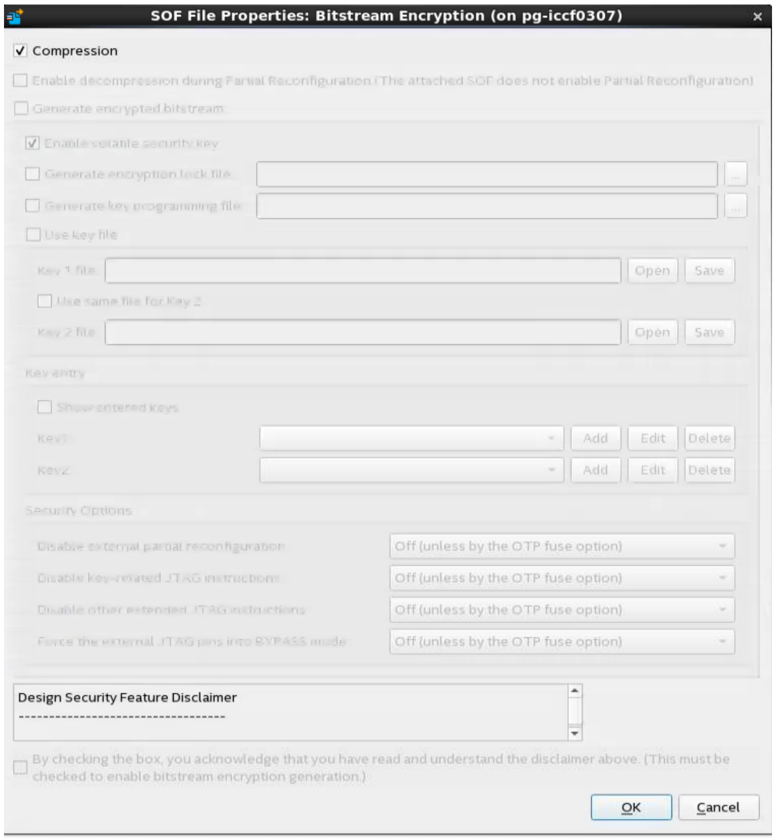
3. In the dialog box, configure it with the setting as below:
   - **Programming file type**: JTAG Indirect Configuration File (.jic)
   - **Configuration device**: EPCQL1024
   - **Mode**: Active Serial x4
   - **File name**: ru.jic
   - Check the **Create Memory Map File (Generate ru.map)** option
   - Check the **Create config data RPD File (Generate ru_auto.rpd)** option

4. Select the **Flash Loader** and click **Add Device…** button to bring up the dialog box as shown in figure. In the dialog box, select **Arria 10** for **Device family**, **10AX115S2** for **Device name** and then click **OK** button.



5. Select the **SOF Data** and click **Add File…** button to locate and add factory image file (.sof).

6. Select the factory image (.sof) file and click on **Properties** button button to bring up the dialog box as shown in figure. In the dialog box, check the **Compression** option and then click **OK** button.



SOF File Properties: Bitstream Encryption (on pg-iccf0307)

[✓] Compression

[ ] Enable decompression during Partial Reconfiguration (The attached SOF does not enable Partial Reconfiguration)

[ ] Generate encrypted bitstream

[✓] Enable volatile security key

[ ] Generate encryption lock file

[ ] Generate key programming file

[ ] Use key file

Key 1 file

[ ] Use same file for Key 2

Key 2 file

Key entry

[ ] Show entered keys

Key1

Key2

Security Options

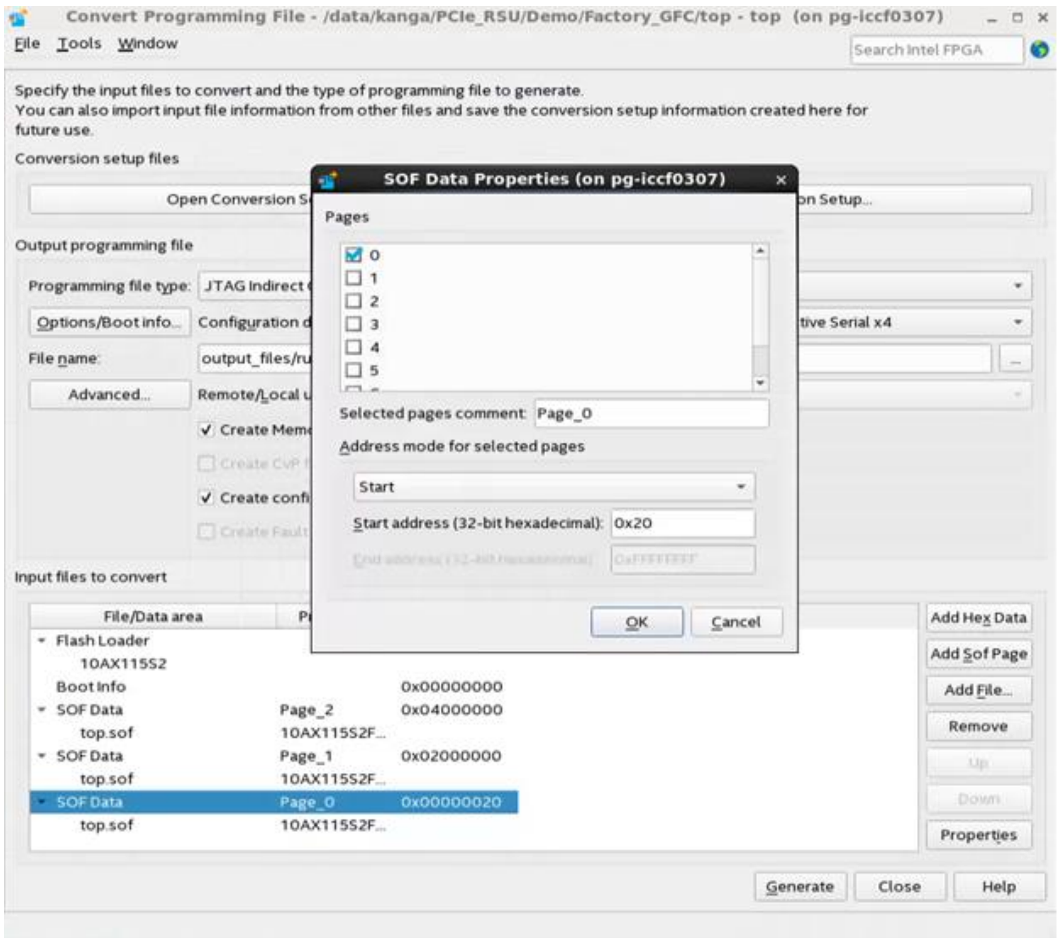
Disable external partial reconfiguration	Off (unless by the OTP fuse option)
Disable key-related JTAG instructions	Off (unless by the OTP fuse option)
Disable other extended JTAG instructions	Off (unless by the OTP fuse option)
Force the external JTAG pins into BYPASS mode	Off (unless by the OTP fuse option)

Design Security Feature Disclaimer

[ ] By checking the box, you acknowledge that you have read and understand the disclaimer above. (This must be checked to enable bitstream encryption generation.)

7. Click on **Add Sof Page** button to add a new Sof Page.
8. Select the **SOF Data** of Page_1 and click **Add File…** button to locate and add application_1 image file (.sof).
9. Select the application image (.sof) file and click on **Properties** button to bring up the dialog box as shown in figure. In the dialog box, check the **Compression** option and then click **OK** button.
10. Repeat steps 7 to 9 above to create Sof Page 2.
11. Select **SOF Data** of Page_0 and click **Properties** button to bring up SOF Data Properties dialog box. In the dialog box, configure it with the setting as below and then click **OK** button.
    - Address mode for selected pages: Start
    - Start address (32-bit hexadecimal): 0x20

12. Select SOF Data of Page_1 and click Properties button to bring up SOF Data Properties dialog box. In the dialog box, configure it with the setting as below and then click **OK** button.
    - Address mode for selected pages: Start
    - Start address (32-bit hexadecimal): 0x2000000
13. Select SOF Data of Page_2 and click Properties button to bring up SOF Data Properties dialog box. In the dialog box, configure it with the setting as below and then click **OK** button.
    - Address mode for selected pages: Start
    - Start address (32-bit hexadecimal): 0x4000000



Convert Programming File - /data/kanga/PCIe_RSU/Demo/Factory_GFC/top - top (on pg-iccf0307)

Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.

Conversion setup files

Output programming file

Programming file type: JTAG Indirect

Configuration d… tive Serial x4

File name: output_files/ru

Remote/Local u

✓ Create Mem

Create CvP f

✓ Create confi

Create Fault

SOF Data Properties (on pg-iccf0307)

Pages

- [✓] 0
- [ ] 1
- [ ] 2
- [ ] 3
- [ ] 4
- [ ] 5

Selected pages comment: Page_0

Address mode for selected pages: Start

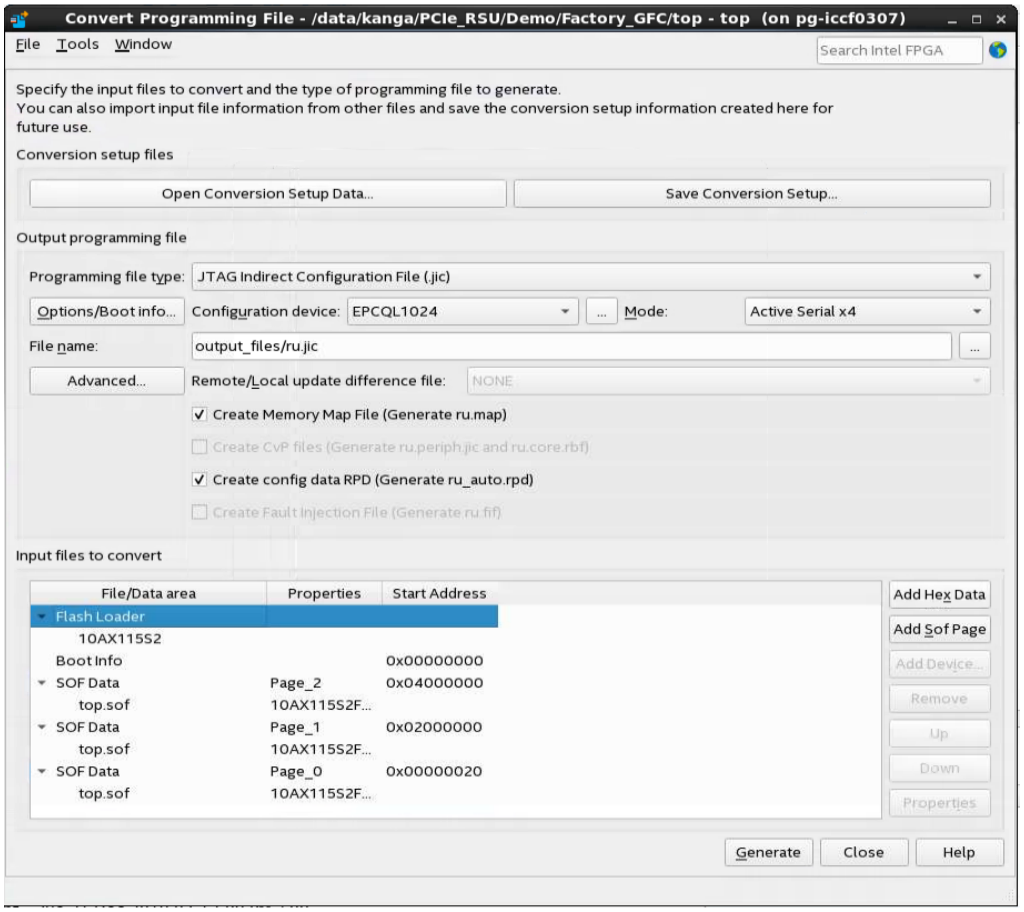
Start address (32-bit hexadecimal): 0x20

End address (32-bit hexadecimal): 0xFFFFFFFF

Input files to convert

File/Data area	P	
Flash Loader		
10AX115S2		
Boot Info		0x00000000
SOF Data	Page_2	0x04000000
top.sof	10AX115S2F...	
SOF Data	Page_1	0x02000000
top.sof	10AX115S2F...	
SOF Data	Page_0	0x00000020
top.sof	10AX115S2F...	

14. Finally, the final Convert Programming File dialog configuration should look as shown in figure below, click on the **Generate** button to generate the ru_auto.rpd file.


Convert Programming File - /data/kanga/PCIe_RSU/Demo/Factory_GFC/top - top (on pg-iccf0307)

Specify the input files to convert and the type of programming file to generate.
You can also import input file information from other files and save the conversion setup information created here for future use.

Conversion setup files

Output programming file

Programming file type: JTAG Indirect Configuration File (.jic)

Options/Boot info... Configuration device: EPCQL1024 Mode: Active Serial x4

File name: output_files/ru.jic

Advanced... Remote/Local update difference file: NONE

✓ Create Memory Map File (Generate ru.map)

☐ Create CvP files (Generate ru.periph.jic and ru.core.rbf)

✓ Create config data RPD (Generate ru_auto.rpd)

☐ Create Fault Injection File (Generate ru.fif)

Input files to convert

File/Data area	Properties	Start Address
Flash Loader		
10AX115S2		
Boot Info		0x00000000
SOF Data	Page_2	0x04000000
top.sof	10AX115S2F...	
SOF Data	Page_1	0x02000000
top.sof	10AX115S2F...	
SOF Data	Page_0	0x00000020
top.sof	10AX115S2F...	

3.6. Setting Up the Hardware

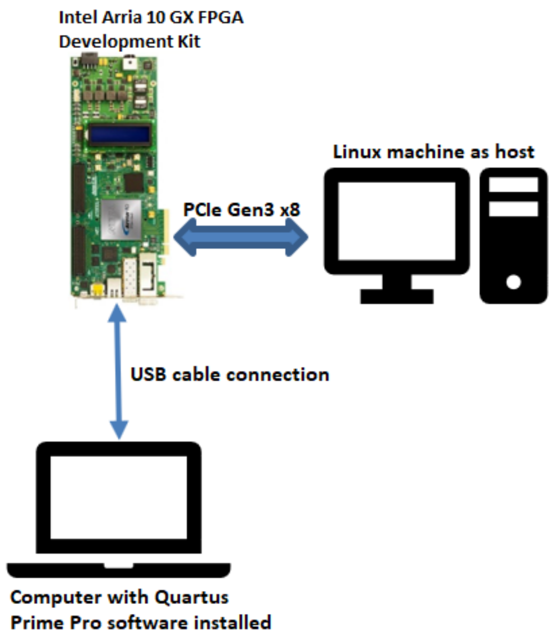

Intel Arria 10 GX FPGA Development Kit
Linux machine as host
PCIe Gen3 x8
USB cable connection
Computer with Quartus Prime Pro software installed

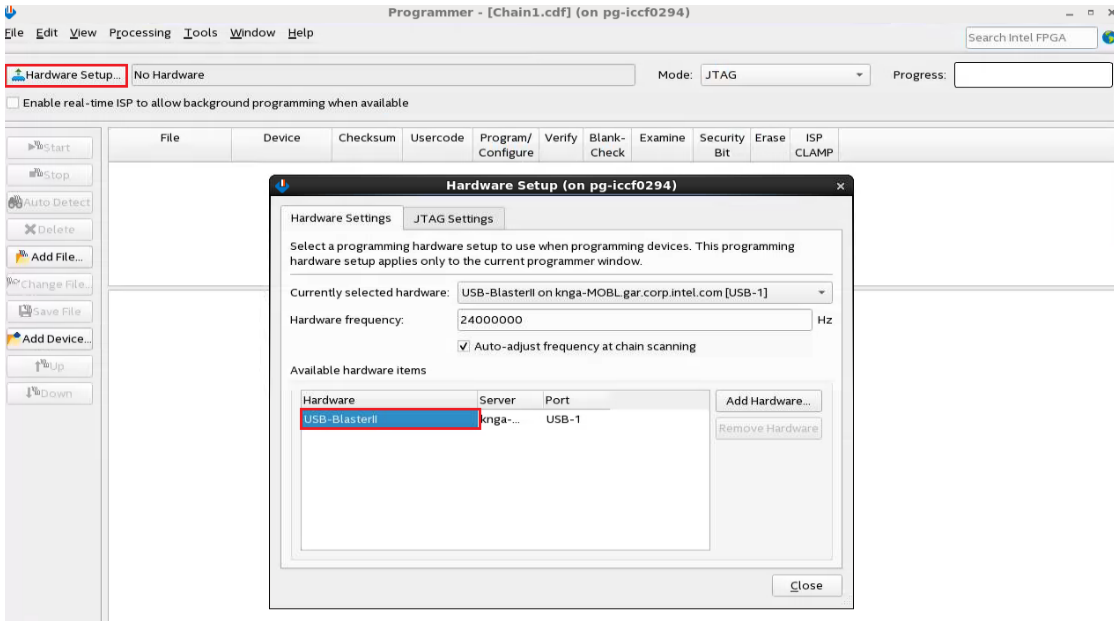
Figure 3: Hardware Setup

1. Plug the Intel Arria 10 GX FPGA Development Kit card into a PCIe slot of the Linux machine that supports Gen3 x8 or Gen3 x16.
2. Connect a USB cable from computer of which Quartus Prime Pro software was installed to the Intel Arria 10 GX FPGA Development Kit at J3 (On-Board USB-Blaster II). Otherwise, you can also use an external USB-Blaster II cable to connect the computer to the Intel Arria 10 GX FPGA Development Kit at J17 (JTAG Header).
3. To power up Intel Arria 10 GX FPGA Development Kit via the PCIe slot, power on the Linux computer. Alternatively, you can power up FPGA Development Kit using the external power adapter that ships with the kit.

3.6.1 Program the Serial Flash (EPCQL) with JTAG Indirect Configuration File

JTAG Indirect Configuration file is a binary file (with the extension .jic) containing serial configuration device data and the serial flash loader device name. Follow the instruction in section 3.5 to create JIC file. It is used to program the configuration data for a into an serial configuration device. The factory image will be loaded by default upon power up. Below are the instructions to program the a serial configuration device with JIC file.

1. On computer installed with Quartus Prime Pro software, invoke the Intel Quartus Prime programmer software from the **Tools** menu of Quartus Prime Pro software.
2. Click on the **Hardware Setup…** button to bring up bring up the Hardware Setup dialog box. In the dialog box, select USB-BalsterII for the hardware and click **Close** button.



Select a programming hardware setup to use when programming devices. This programming hardware setup applies only to the current programmer window.

Currently selected hardware: USB-BlasterII on knga-MOBL.gar.corp.intel.com [USB-1]

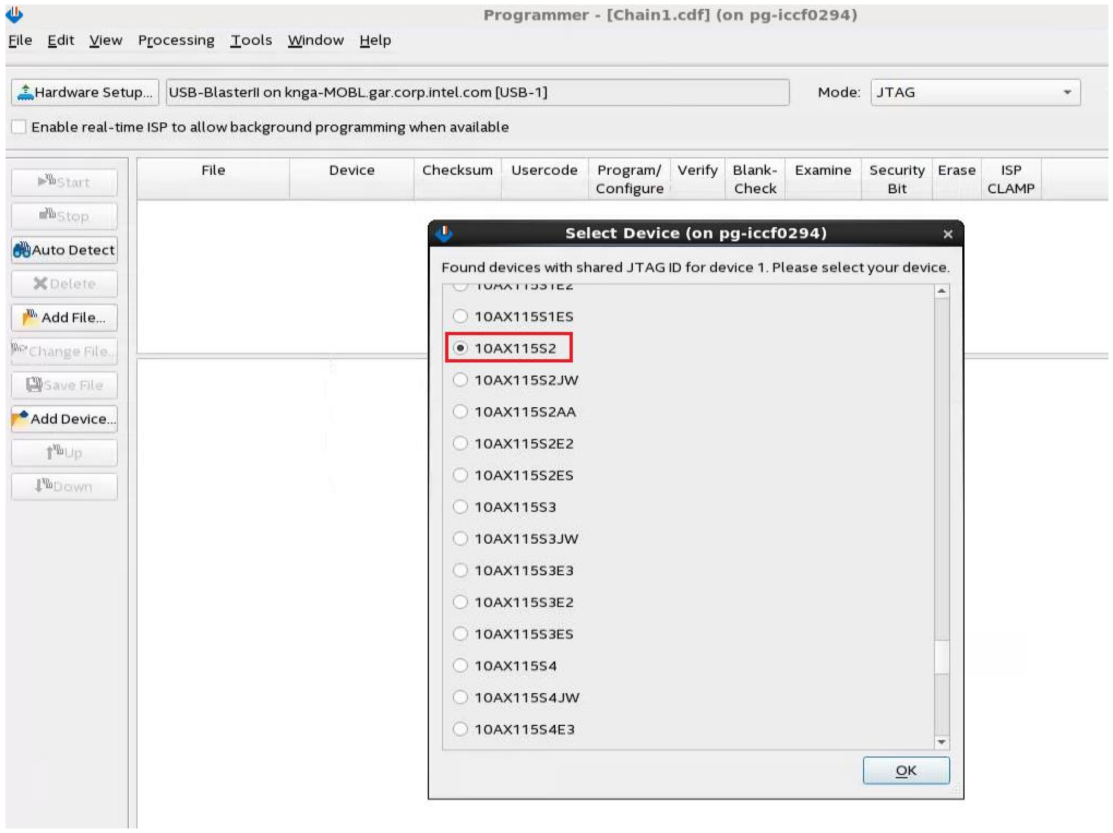
Hardware frequency: 24000000 Hz

[✓] Auto-adjust frequency at chain scanning

Available hardware items

Hardware	Server	Port
USB-BlasterII	knga-...	USB-1

3. Click on **Auto Detect** button, select **10AX115S2** device and then click **OK** button.

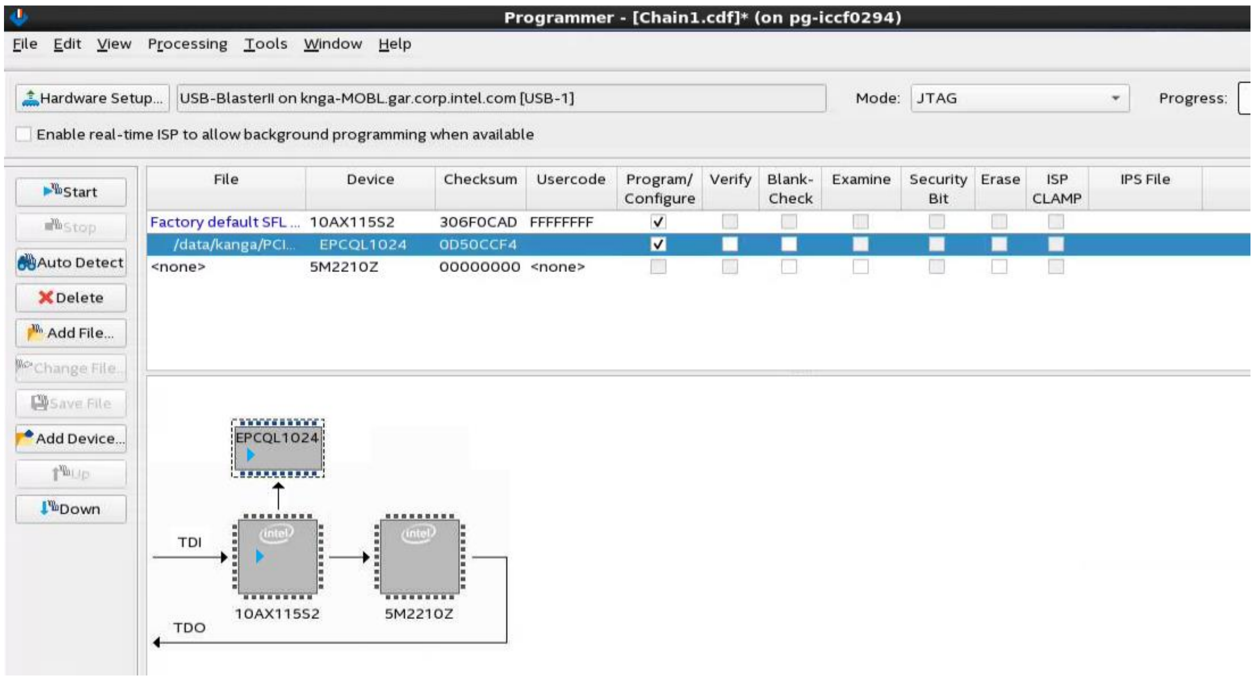


USB-BlasterII on knga-MOBL.gar.corp.intel.com [USB-1]

Found devices with shared JTAG ID for device 1. Please select your device.

- 10AX115S1E2
- 10AX115S1ES
- [✓ 10AX115S2]
- 10AX115S2JW
- 10AX115S2AA
- 10AX115S2E2
- 10AX115S2ES
- 10AX115S3
- 10AX115S3JW
- 10AX115S3E3
- 10AX115S3E2
- 10AX115S3ES
- 10AX115S4
- 10AX115S4JW
- 10AX115S4E3

4. Select **10AX115S2** device and then click on **Change File…** button to locate and specify the .jic programming file.



USB-BlasterII on knga-MOBL.gar.corp.intel.com [USB-1]

Mode: JTAG

Progress:

Enable real-time ISP to allow background programming when available

File	Device	Checksum	Usercode	Program/ Configure	Verify	Blank-Check	Examine	Security Bit	Erase	ISP CLAMP	IPS File
Factory default SFL ...	10AX115S2	306F0CAD	FFFFFFFF	✓							
/data/kanga/PCI...	EPCQL1024	0D50CCF4		✓							
<none>	5M2210Z	00000000	<none>								

5. Check the Program/Configuration option for the EPCQL device as shown in figure and then click on **Start** button to program the EPCQL device.

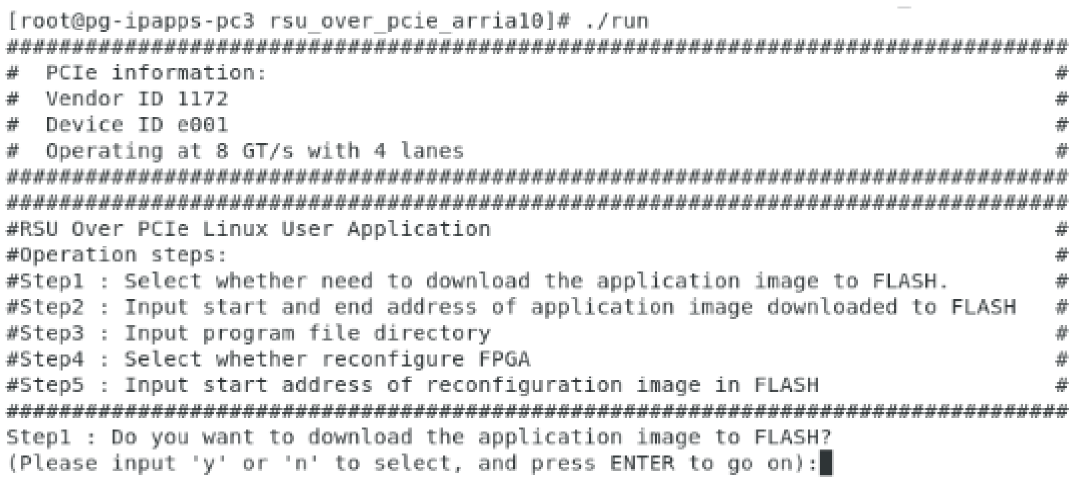
6. Shut down the host machine and then power up again, to allow the FPGA to be configured with the factory image in the EPCQL device, so that the system enumeration could take place accordingly.

Note: If SOF configuration file is used to configure the FPGA, the FPGA must be reconfigured whenever the Arria 10 FPGA Development Kit loses power. Hence, make sure that there is no power interruption during system reboot, otherwise the enumeration will fail and the device can not be detected. With the serial configuration device, EPCQL been configured with the JIC file, the factory image will be loaded automatically upon power cycle.

7. Notice that four red User LEDs lit up when the factory FPGA image is loaded sucessfully.

3.7. Installing the Remote System Update via PCIe Design Driver and Running the Application Software

1. Copy the Linux software driver package, rsu_over_pcie_arria10_driver.tar.gz from reference design directory, “<project_directory>/driver” into the Linux machine of which the Intel Arria 10 GX FPGA Development Kit was installed.
2. Extract the Linux software driver package by running the following commands in a terminal window:
   - cd /<reference_design_install_directory>/<project_directory>/driver
   - tar -xzf rsu_over_pcie_arria10_driver.tar.gz
3. In the software driver directory, run the following command to install the Linux driver for the hardware:
   - sudo ./install
4. Run the following command to execute the remote system update via PCIe application:
   - ./run
5. The application software prints the operation steps as shown below. Refer to figure 4 for the software flow.

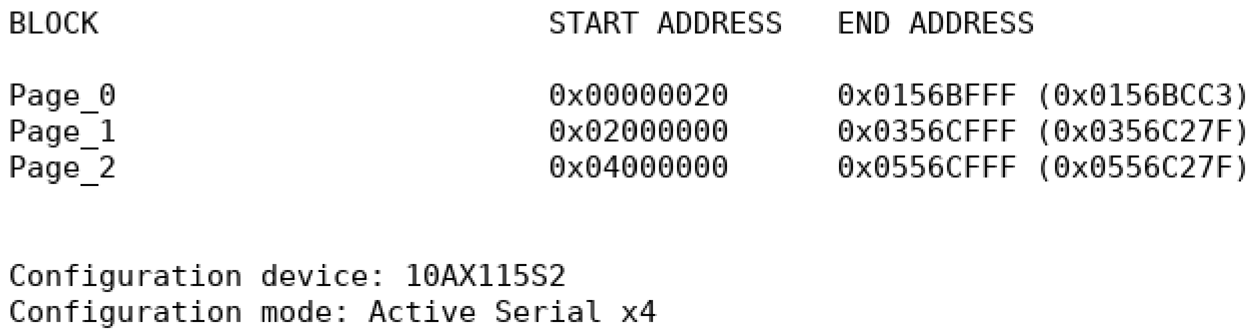


```
[root@pg-ipapps-pc3 rsu_over_pcie_arria10]# ./run
################################################################################
#  PCIe information:                                                           #
#  Vendor ID 1172                                                              #
#  Device ID e001                                                              #
#  Operating at 8 GT/s with 4 lanes                                            #
################################################################################
################################################################################
#RSU Over PCIe Linux User Application                                          #
#Operation steps:                                                              #
#Step1 : Select whether need to download the application image to FLASH.       #
#Step2 : Input start and end address of application image downloaded to FLASH   #
#Step3 : Input program file directory                                          #
#Step4 : Select whether reconfigure FPGA                                       #
#Step5 : Input start address of reconfiguration image in FLASH                 #
################################################################################
Step1 : Do you want to download the application image to FLASH?
(Please input 'y' or 'n' to select, and press ENTER to go on):
```

3.7.1. Writing Application Image 1 into the Serial Flash

1. User will prompted the operation steps while running the remote system update via PCIe application software. To download the application image into the serial Flash or EPCQL, enter “y” and press the Enter key to specify the image to be downloaded into the serial flash device.

2. Enter 0x2000000 for the Start Address and 0x356cfff for the End Address. Refer to the ru.map file generated by the **Convert Programming File** wizard for the image address mapping.

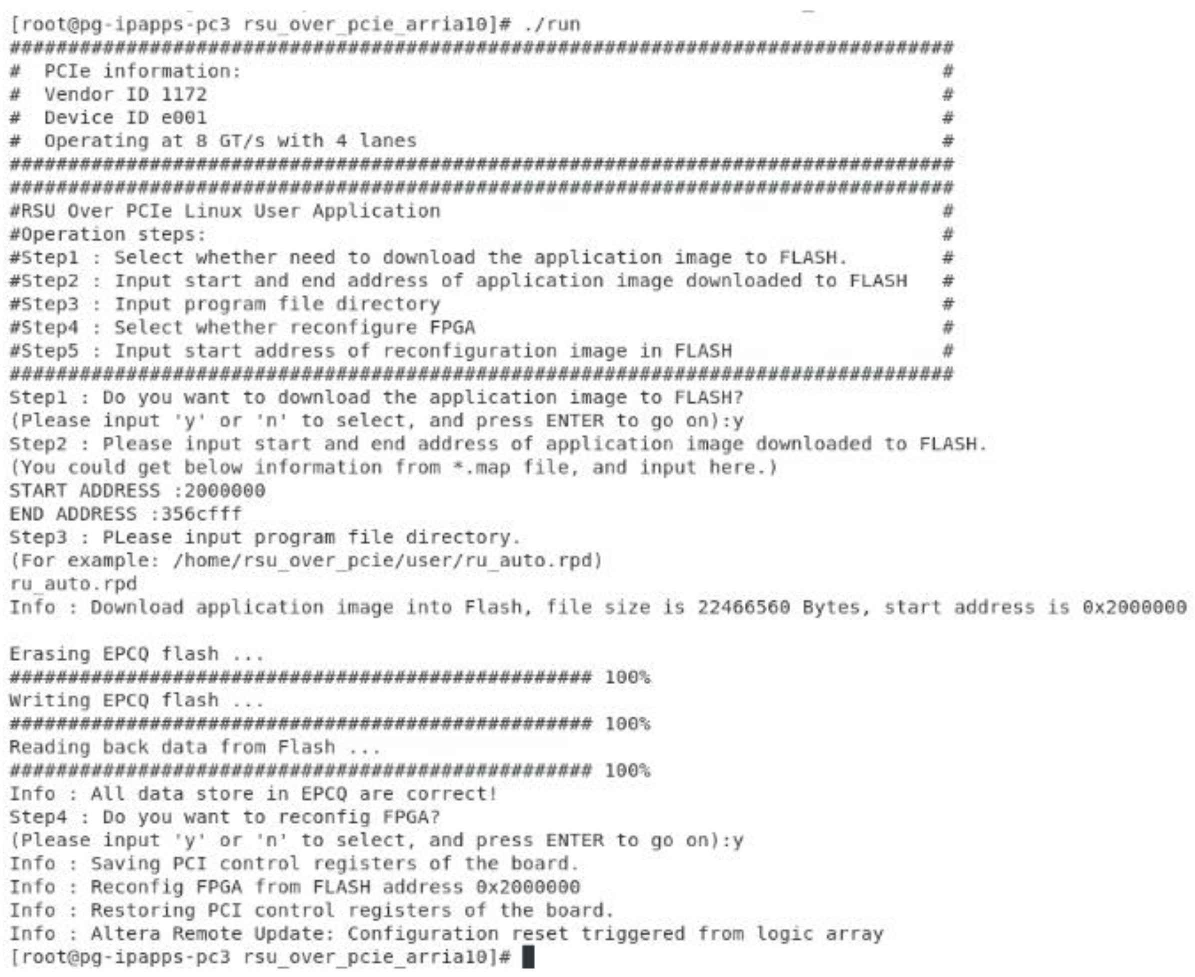


```
BLOCK                          START ADDRESS   END ADDRESS

Page_0                         0x00000020      0x0156BFFF (0x0156BCC3)
Page_1                         0x02000000      0x0356CFFF (0x0356C27F)
Page_2                         0x04000000      0x0556CFFF (0x0556C27F)


Configuration device: 10AX115S2
Configuration mode: Active Serial x4
```

Note: Any change in the design might result a different fpga image size and thus a different end address in the .map file. Please follow the addresses in your .map file.

3. Copy the previously generated ru.rpd file into the Linux machine that running the remote system update via PCIe application software.
4. When prompted in the remote system update via PCIe application software, specify the path to ru.rpd file and press Enter key to proceed.
5. Once the application 1 image has been successfully downloaded into the serial flash device, user will be prompted “All data store in EPCQL are correct!” and option to reconfigure the FPGA with the newly loaded image.

6. Enter “y” and press Enter key to reconfigure the FPGA with the newly loaded image. The FPGA is then reconfigured with application 1 image and only one red LED light lit up on the Intel Arria 10 GX FPGA Development Kit.



```
[root@pg-ipapps-pc3 rsu_over_pcie_arria10]# ./run
################################################################################
#  PCIe information:                                                           #
#  Vendor ID 1172                                                              #
#  Device ID e001                                                              #
#  Operating at 8 GT/s with 4 lanes                                            #
################################################################################
################################################################################
#RSU Over PCIe Linux User Application                                          #
#Operation steps:                                                              #
#Step1 : Select whether need to download the application image to FLASH.       #
#Step2 : Input start and end address of application image downloaded to FLASH  #
#Step3 : Input program file directory                                          #
#Step4 : Select whether reconfigure FPGA                                       #
#Step5 : Input start address of reconfiguration image in FLASH                 #
################################################################################
Step1 : Do you want to download the application image to FLASH?
(Please input 'y' or 'n' to select, and press ENTER to go on):y
Step2 : Please input start and end address of application image downloaded to FLASH.
(You could get below information from *.map file, and input here.)
START ADDRESS :2000000
END ADDRESS :356cfff
Step3 : PLease input program file directory.
(For example: /home/rsu_over_pcie/user/ru_auto.rpd)
ru_auto.rpd
Info : Download application image into Flash, file size is 22466560 Bytes, start address is 0x2000000

Erasing EPCQ flash ...
################################################## 100%
Writing EPCQ flash ...
################################################## 100%
Reading back data from Flash ...
################################################## 100%
Info : All data store in EPCQ are correct!
Step4 : Do you want to reconfig FPGA?
(Please input 'y' or 'n' to select, and press ENTER to go on):y
Info : Saving PCI control registers of the board.
Info : Reconfig FPGA from FLASH address 0x2000000
Info : Restoring PCI control registers of the board.
Info : Altera Remote Update: Configuration reset triggered from logic array
[root@pg-ipapps-pc3 rsu_over_pcie_arria10]# █
```

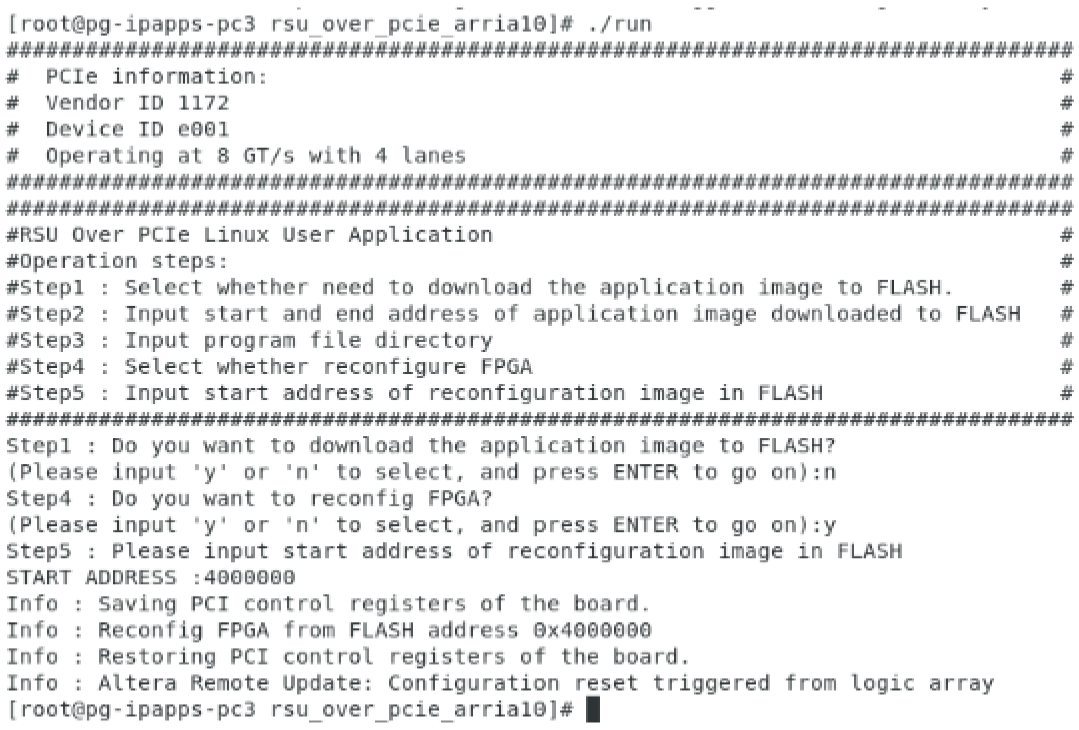
3.7.2. Writing Application Image 2 into the Serial Flash

1. Run the remote system update via PCIe application software.
2. Enter “y” and press the Enter key to specify the image to be downloaded into the serial flash or EPCQL device.
3. Enter 0x4000000 for the Start Address and 0x556cfff for the End Address. Refer to the ru.map file generated by the **Convert Programming File** wizard for the image address mapping.
4. Specify the path to ru.rpd file and press Enter key to proceed.
5. Once the application 2 image has been successfully downloaded into the serial flash device, you will be prompted “All data store in EPCQL are correct!” and option to reconfigure the FPGA with the newly loaded image.

6. Enter “n” and press Enter key to exit the application.

3.7.3. Trigger FPGA Reconfiguration

1. Run the remote system update via PCIe application software.
2. Enter “n” and press Enter key to skip the procedure to download the application image into serial flash device.
3. Enter 0x4000000 for the Start Address of the reconfiguration image in the serial flash device and press Enter key.
   The FPGA is then reconfigured with application 2 image and two red LED lights lit up on the Intel Arria 10 GX FPGA Development Kit.



```
[root@pg-ipapps-pc3 rsu_over_pcie_arria10]# ./run
#################################################################################
#  PCIe information:                                                            #
#  Vendor ID 1172                                                               #
#  Device ID e001                                                               #
#  Operating at 8 GT/s with 4 lanes                                             #
#################################################################################
#################################################################################
#RSU Over PCIe Linux User Application                                           #
#Operation steps:                                                               #
#Step1 : Select whether need to download the application image to FLASH.        #
#Step2 : Input start and end address of application image downloaded to FLASH   #
#Step3 : Input program file directory                                           #
#Step4 : Select whether reconfigure FPGA                                        #
#Step5 : Input start address of reconfiguration image in FLASH                  #
#################################################################################
Step1 : Do you want to download the application image to FLASH?
(Please input 'y' or 'n' to select, and press ENTER to go on):n
Step4 : Do you want to reconfig FPGA?
(Please input 'y' or 'n' to select, and press ENTER to go on):y
Step5 : Please input start address of reconfiguration image in FLASH
START ADDRESS :4000000
Info : Saving PCI control registers of the board.
Info : Reconfig FPGA from FLASH address 0x4000000
Info : Restoring PCI control registers of the board.
Info : Altera Remote Update: Configuration reset triggered from logic array
[root@pg-ipapps-pc3 rsu_over_pcie_arria10]# █
```

3.8 Summary

Once the Arria 10 FPGA is loaded with the reference design and enumerated successfully, the Remote System Update via PCIe application software can be used to write an application image in Raw Programmming File format to the target section of the serial flash and trigger FPGA reconfiguration to with an application image in the serial flash.

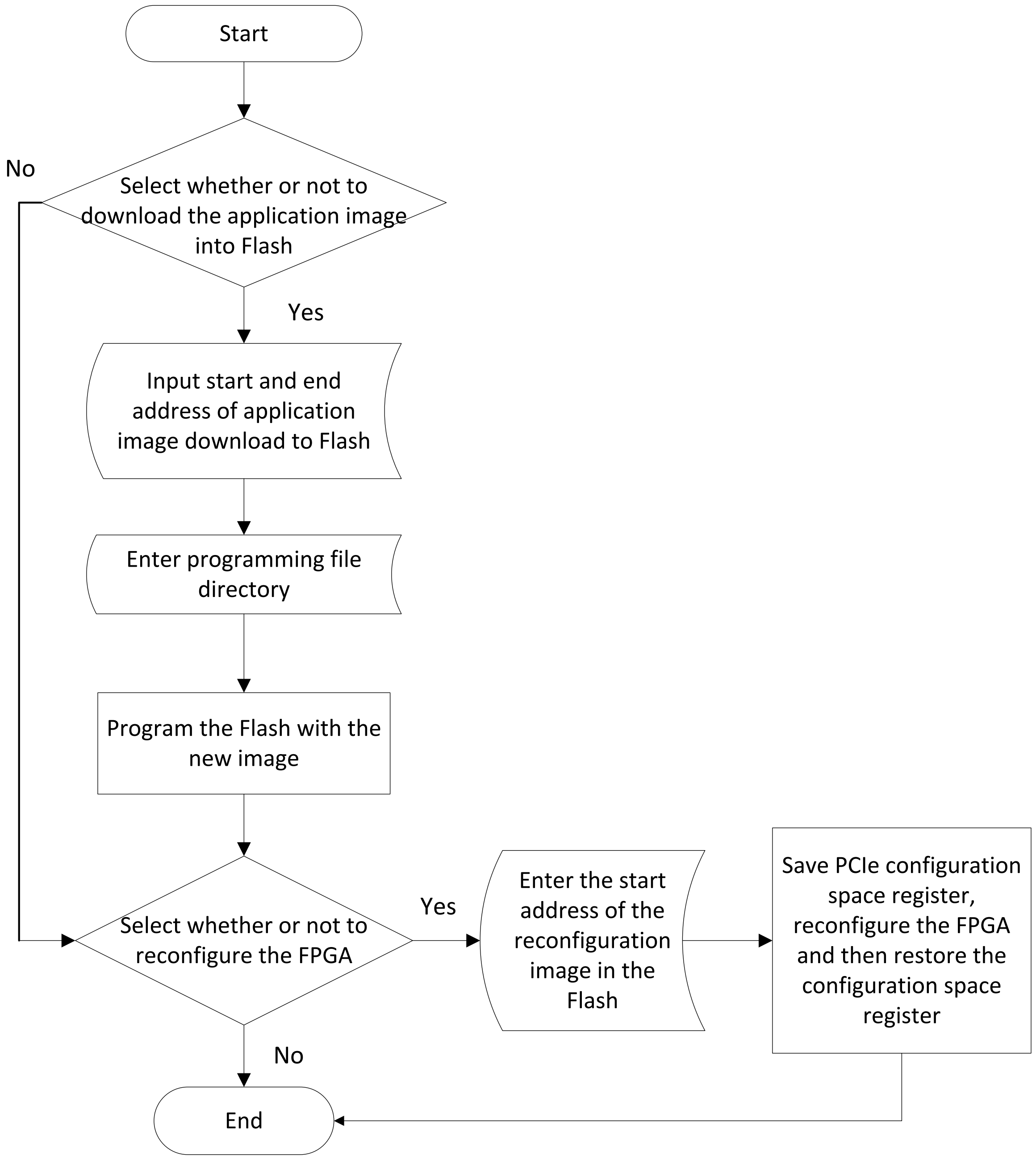

Start
Select whether or not to download the application image into Flash
No
Yes
Input start and end address of application image download to Flash
Enter programming file directory
Program the Flash with the new image
Select whether or not to reconfigure the FPGA
Yes
Enter the start address of the reconfiguration image in the Flash
Save PCIe configuration space register, reconfigure the FPGA and then restore the configuration space register
No
End

Figure 4: Remote System Update via PCIe User Application Process Flow

4. Design Description

The reference design top level module instantiates rsu_pcie module which is constructed using Platform Designer. It consists of Intel® Arria 10 PCIe Gen3 x4 Hard IP, Modular Scatter-Gather DMA Intel FPGA IP, Generic Serial Flash Interface Intel FPGA IP, Remote Update Intel FPGA IP to demonstrate remote system update via PCI Express capability.

The 50MHz input reference clock is used for IOPLL to generate 20MHz clock for Remote Update FPGA IP and 100MHz clock for Generic Serial Flash Interface FPGA IP. All the IP can be reset asynchronously during user mode via cpu_resetn push button.

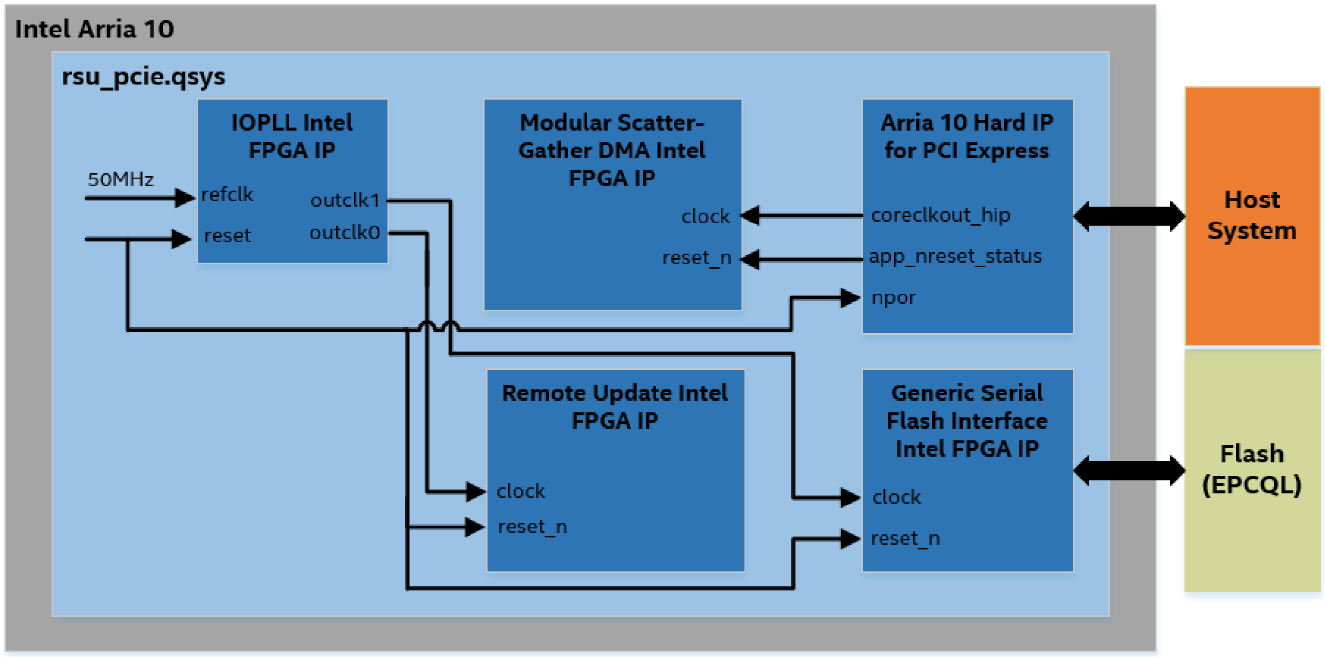
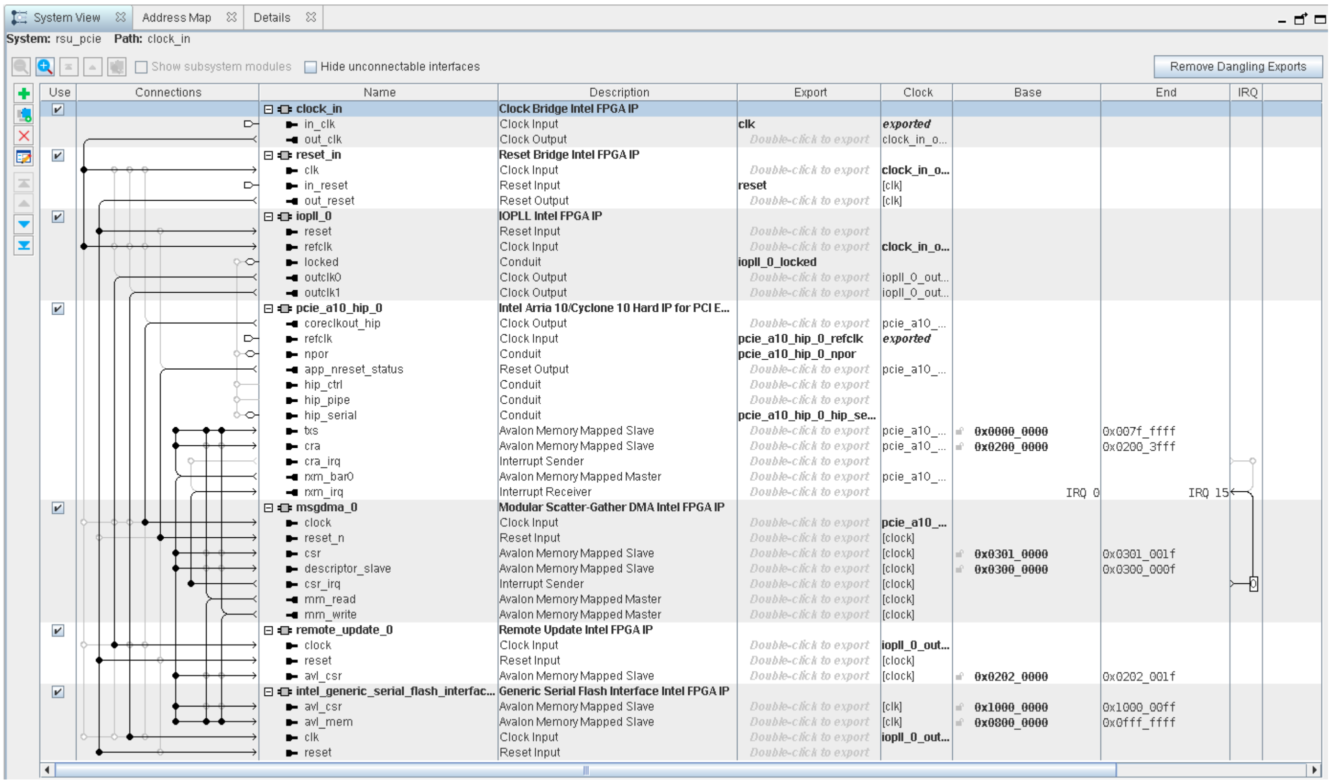


Figure 5: Reference Design Clock and Reset Block Diagram

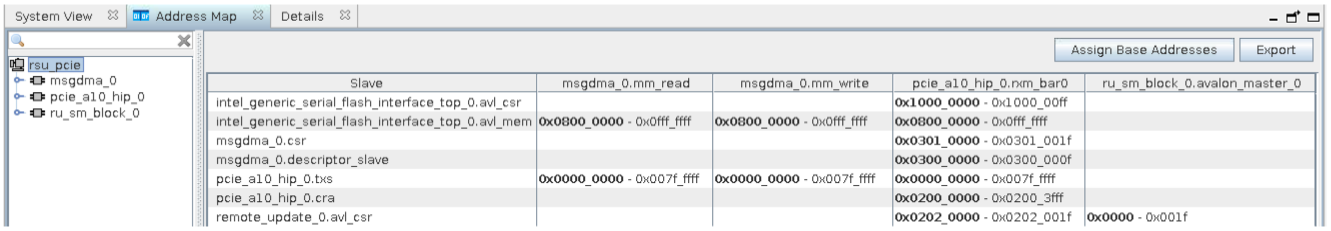
4.1. Design Components

Platform Designer is used to integrate various design components or FPGA IPs to serve the functionalities as discussed for Remote System Update via PCIe Reference Design. All the FPGA IP used in this design uses Avalon Memory Mapped Interface and can be hooked up directly in Platform Designer.



Use	Name	Description	Export	Clock	Base	End	IRQ
✓	clock_in	Clock Bridge Intel FPGA IP					
	in_clk	Clock Input	clk	exported			
	out_clk	Clock Output	Double-click to export	clock_in_o...			
✓	reset_in	Reset Bridge Intel FPGA IP					
	clk	Clock Input	Double-click to export	clock_in_o...			
	in_reset	Reset Input	reset	[clk]			
	out_reset	Reset Output	Double-click to export	[clk]			
✓	iopll_0	IOPLL Intel FPGA IP					
	reset	Reset Input	Double-click to export				
	refclk	Clock Input	Double-click to export	clock_in_o...			
	locked	Conduit	iopll_0_locked				
	outclk0	Clock Output	Double-click to export	iopll_0_out...			
	outclk1	Clock Output	Double-click to export	iopll_0_out...			
✓	pcie_a10_hip_0	Intel Arria 10/Cyclone 10 Hard IP for PCI E...					
	coreclkout_hip	Clock Output	Double-click to export	pcie_a10_...			
	refclk	Clock Input	pcie_a10_hip_0_refclk	exported			
	npor	Conduit	pcie_a10_hip_0_npor				
	app_nreset_status	Reset Output	Double-click to export	pcie_a10_...			
	hip_ctrl	Conduit	Double-click to export				
	hip_pipe	Conduit	Double-click to export				
	hip_serial	Conduit	pcie_a10_hip_0_hip_se...				
	txs	Avalon Memory Mapped Slave	Double-click to export	pcie_a10_...	0x0000_0000	0x007f_ffff	
	cra	Avalon Memory Mapped Slave	Double-click to export	pcie_a10_...	0x0200_0000	0x0200_3fff	
	cra_irq	Interrupt Sender	Double-click to export				
	rxm_bar0	Avalon Memory Mapped Master	Double-click to export	pcie_a10_...			
	rxm_irq	Interrupt Receiver	Double-click to export		IRQ 0	IRQ 15	
✓	msgdma_0	Modular Scatter-Gather DMA Intel FPGA IP					
	clock	Clock Input	Double-click to export	pcie_a10_...			
	reset_n	Reset Input	Double-click to export	[clock]			
	csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0301_0000	0x0301_001f	
	descriptor_slave	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0300_0000	0x0300_000f	
	csr_irq	Interrupt Sender	Double-click to export	[clock]			0
	mm_read	Avalon Memory Mapped Master	Double-click to export	[clock]			
	mm_write	Avalon Memory Mapped Master	Double-click to export	[clock]			
✓	remote_update_0	Remote Update Intel FPGA IP					
	clock	Clock Input	Double-click to export	iopll_0_out...			
	reset	Reset Input	Double-click to export	[clock]			
	avl_csr	Avalon Memory Mapped Slave	Double-click to export	[clock]	0x0202_0000	0x0202_001f	
✓	intel_generic_serial_flash_interfac...	Generic Serial Flash Interface Intel FPGA IP					
	avl_csr	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x1000_0000	0x1000_00ff	
	avl_mem	Avalon Memory Mapped Slave	Double-click to export	[clk]	0x0800_0000	0x0fff_ffff	
	clk	Clock Input	Double-click to export	iopll_0_out...			
	reset	Reset Input	Double-click to export				

Figure 6: Platform Designer View of the Remote System Update via PCIe Reference Design



Slave	msgdma_0.mm_read	msgdma_0.mm_write	pcie_a10_hip_0.rxm_bar0	ru_sm_block_0.avalon_master_0
intel_generic_serial_flash_interface_top_0.avl_csr			0x1000_0000 - 0x1000_00ff	
intel_generic_serial_flash_interface_top_0.avl_mem	0x0800_0000 - 0x0fff_ffff	0x0800_0000 - 0x0fff_ffff	0x0800_0000 - 0x0fff_ffff	
msgdma_0.csr			0x0301_0000 - 0x0301_001f	
msgdma_0.descriptor_slave			0x0300_0000 - 0x0300_000f	
pcie_a10_hip_0.txs	0x0000_0000 - 0x007f_ffff	0x0000_0000 - 0x007f_ffff	0x0000_0000 - 0x007f_ffff	
pcie_a10_hip_0.cra			0x0200_0000 - 0x0200_3fff	
remote_update_0.avl_csr			0x0202_0000 - 0x0202_001f	0x0000 - 0x001f

Figure 7: System Address Map

4.1.1. Intel Arria 10 PCIe Gen3 Hard IP

Intel Arria 10 FPGAs include a configurable, hardened protocol stack for PCI Express* that is compliant with PCI Express Base Specification 3.0. The Hard IP for PCI Express IP core using the Avalon® Memory-Mapped (Avalon-MM) interface removes some of the complexities associated with the PCIe* protocol. For example, it handles all of the

Transaction Layer Packet (TLP) encoding and decoding. Consequently, you can complete your design more quickly. The Avalon-MM interface is implemented as a bridge in soft logic. It is available in Platform Designer.

In this design, Modular Scatter-Gather DMA IP is used as data mover to move remote system update configuration data between buffer in host system memory and serial flash through PCIe link.

The system is running on 32-bits Avalon-MM bus which means the 32-bits Avalon-MM address of a received request on the TX Avalon-MM slave port would need to be translated to the 64-bits PCI Express address before the request packet is sent to the Transaction Layer. Translation setting of 1 address page with size of 4MB page size is used in this design.

In the translation, the lower 21 bits LSB will be carried forward between the 2 address spaces while MSB of the Avalon-MM address will be replaced with the value from the translation table entry. Address in Avalon-MM address space is the serial flash address while address in PCIe address space is the host memory buffer address. Only 1 address page is used in this design. The value in the translation table entry is set to host memory buffer address.

- *N*—the number of pass-through bits.
- *M*—the number of Avalon-MM address bits.
- *P*—the number of PCIe address bits.
- *Q*—the number of translation table entries.
- `Sp[1:0]`—the space indication for each entry.

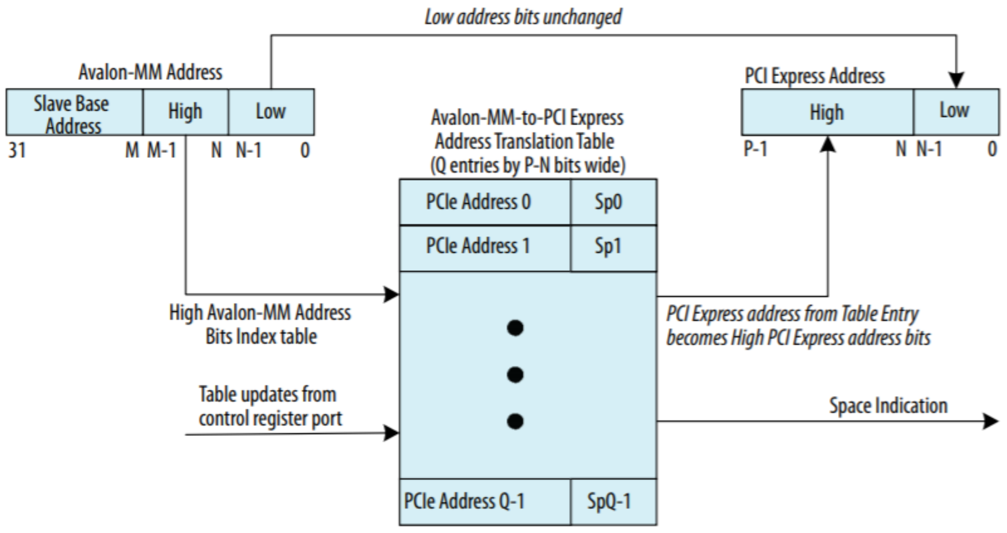

Low address bits unchanged
Avalon-MM Address
Slave Base Address
High
Low
31
M M-1
N N-1
0
Avalon-MM-to-PCI Express Address Translation Table (Q entries by P-N bits wide)
PCI Express Address
High
Low
P-1
N N-1
0
PCIe Address 0
Sp0
PCIe Address 1
Sp1
PCIe Address Q-1
SpQ-1
High Avalon-MM Address Bits Index table
Table updates from control register port
PCI Express address from Table Entry becomes High PCI Express address bits
Space Indication

Figure 8: Avalon Memory Mapped to PCIe Address Translation

Related Information:
[Intel® Arria® 10 and Intel® Cyclone® 10 GX Avalon® Memory Mapped (Avalon-MM) Interface for PCI Express* User Guide]

4.1.2. Modular Scatter-Gather DMA Intel FPGA IP

In a processor subsystem, data transfers between two memory spaces can happen frequently. In order to offload the processor from moving data around a system, a Direct Memory Access (DMA) engine is introduced to perform this function instead. The Modular Scatter-Gather DMA (mSGDMA) is capable of performing data movement operations with preloaded instructions, called descriptors. Multiple descriptors with different transfer sizes, and source and destination addresses have the option to trigger interrupts.

The mSGDMA core has a modular design that facilitates easy integration with the FPGA fabric. The core consists of a dispatcher block with optional read master and write master blocks. The descriptor block receives and decodes the descriptor, and dispatches instructions to the read master and write master blocks for further operation. The block is also configured to transfer additional information to the host. In this context, the read master block reads data through its Avalon-MM master interface, and channels it into the Avalon-ST source interface, based on instruction given by the dispatcher block. Conversely, the write master block receives data from its Avalon-ST sink interface and writes it to the destination address via its Avalon-MM master interface.

The mSGDMA provides three configuration structures for handling data transfers between the Avalon-MM to Avalon-MM, Avalon-MM to Avalon-ST, and Avalon-ST to Avalon-MM modes. The sub-core of the mSGDMA is instantiated automatically according to the structure configured for the mSGDMA use model.

Please refer to *section 30.7.1.6. Software Programming Model* in Embedded Peripherals IP User Guide for details mSGDMA operation flow.

Related Information:
Modular Scatter-Gather DMA Core

4.1.3. Generic Serial Flash Interface Intel FPGA IP

The Generic Serial Flash Interface Intel® FPGA IP core provides access to Serial Peripheral Interface (SPI) flash devices. The Generic Serial Flash Interface IP is a more efficient alternative compared to the ASMI Parallel and ASMI Parallel II Intel FPGA IP cores. The Generic Serial Flash Interface Intel FPGA IP core supports Intel configuration devices as well as flash from different vendors. Intel recommends you to use the Generic Serial Flash Interface Intel FPGA IP core for new designs.

Below are the settings used in the design:
Addressing mode: 4-bytes addressing
Chip select: First device
Baud rate: 50MHz
Transfer mode: Standard SPI mode

Please refer to *section 1.5. Using Generic Serial Flash Interface IP* in Generic Serial Flash Interface Intel FPGA IP Core User Guide for GSFI operation flow.

Related Information:
Generic Serial Flash Interface Intel FPGA IP Core User Guide

4.1.4. Remote Update Intel FPGA IP

The Remote Update Intel® FPGA IP core implements a device reconfiguration using dedicated remote system upgrade circuitry available in supported devices. Remote system upgrade helps you deliver feature enhancements and bug fixes without recalling your product, reduces time-to-market, and extends product life. The Remote Update Intel FPGA IP core commands the configuration circuitry to start a reconfiguration cycle.

The dedicated circuitry performs error detection during and after the configuration process. When the dedicated circuitry detects errors, the circuitry facilitates system recovery by reverting back to a safe, default factory configuration image and then provides error status information.

Please refer to *section 1.2.1.1.1. Switching from Factory to Application Image or from Initial Application Image to Other Application Image* in Remote Update Intel® FPGA IP User Guide for Remote Update IP operation flow.

Related Information:
Remote Update Intel® FPGA IP User Guide

5. User Application and Loadable Device Driver

User application provides an user interface to the user and interact with driver in accessing PCIe BAR space register to performed configuration on IP integrated in the design such as mSGDMA, Generic Serial Flash Interface and Remote Update FPGA IP. It is capable to read programming file locally and write the programming file into the serial flash on PCIe Card as well as reading back the programming file content stored in the serial flash for content verification. Loadble device driver is using read or write functions of file_operations to offer PCIe read or write interface to the application.

In order to achieve a better performance and eficiency, this design is using mSGDMA as data mover in transferring the large programming file from system RAM into the serial flash, EPCQL and transferring the programming data from serial flash to system RAM for verification.

6. Useful APIs

API	Return Value	Descriptions
write32 (ssize_t dev_id, int bar_id, unsigned int dev_addr, unsigned int val)	void	This API is used to perform 32 bits write from PCIe RXM BAR Avalon-MM Master interface to Avalon-MM Slave interface. Example: write32(dev_id, 0, ADDR_GSFI_CSR+(0x1<<2), 0x1) *This operation performs 32-bits write from PCIe RXM BAR 0 to GSFI SPI Clock Baud-rate Register (0x1) in 32-bits system with value of 0x1.*
read32 (ssize_t dev_id, int bar_id, unsigned int dev_addr)	unsigned int	This API is used to perform 32 bits read from PCIe RXM BAR Avalon-MM Master interface to Avalon-MM Slave interface. Example: unsigned int r_value=read32(dev_id, 0, ADDR_GSFI_CSR+(0xc<<2)) *This operation performs 32-bits read using PCIe RXM BAR 0 from GSFI Flash Command Read Data 0 Register(0xc) in 32-bits system. The return value is in unsigned int data type*

dev_id – rsu_over_pcie device.

bar_id – PCIe RXM BAR number, BAR 0 is used in this design.

dev_addr – address to perform write/read.

val – value to write.

7. Document Revision History

List the revision history for the application note.

Date	Version	Changes
Mar 2021	2021.3.26	Initial Release