Partial Reconfiguration with the Intel® Arria® 10 HPS

2017.01.25

AN-798 Subscribe Send Feedback

Partial reconfiguration (PR) allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function.

With partial reconfiguration, you can create multiple logic implementations for specific physical regions of the device, and reload any desired implementation at runtime. This methodology is effective in systems where multiple functions time-share the same FPGA device resources. PR enables the implementation of more complex FPGA systems.

Because Arria 10 SoC devices include the hard processor system (HPS), you can use software running on the device itself to load and reload multiple logic images. The HPS can retrieve images from a remote location, for example via Ethernet, allowing system updates and remote logic image management.

Advantages of Partial Reconfiguration

PR provides the following advantages over designs without PR:

- Allows runtime design reconfiguration
- Improves design scalability by time-sharing hardware resources
- Lowers cost and power consumption through efficient use of board space
- Improves initial programming time through smaller bitstreams
- Reduces system down-time by enabling live updates
- Facilitates system update by allowing fast, low-risk remote hardware changes

PR is a way to load different or updated soft logic without disturbing the HPS host software. Performing a full FPGA image configuration in an Intel® SoC FPGA resets and reconfigures all shared I/O and DDR memory interfaces. Therefore, full reconfiguration can crash the HPS host software if it is using the shared I/O or DDR memory at the time. In contrast, loading a reconfigurable logic region does not affect these critical interfaces.

Scope of This Document

This document is an overview of the coordinated hardware and software workflows required for partial reconfiguration. A basic review of the PR feature from the perspectives of the FPGA logic designer and host software designer is provided. A simple design example is included, with steps to generate it. This basic example enables software developers to generate FPGA images for testing without requiring in-depth knowledge of FPGA design. The example also enables FPGA designers to test more complicated FPGA designs without expert level experience with Linux.

The PR design methodology and design flow have a significant number of guidelines and restrictions. This document does not provide a complete definition of the PR design flow. Enough information is provided to let you get started. Following steps in the document, you generate an example PR design based on the SoC GHRD. FPGA logic design considerations for PR that impact the host software developer are also covered.

This document describes how to create FPGA configuration images. Step by step instructions for modifying a basic design example are included.

The FPGA logic designer must provide certain information to the host software developer for PR management to work correctly and reliably with the HPS. Guidelines for this handoff are provided.

If you are an FPGA logic designer, Intel encourages you to review the software steps as an introduction to enabling PR on the device. You should also review the *Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation* and the *Arria 10 GSRD v16.1 Getting Started Guide* on **RocketBoards.org** for further information about PR restrictions, advanced features, and best practices to help you create more complicated and impactful designs.

This document provides an overview of the process used to manage reconfigurable regions using Linux running on the HPS. There are ways to manage the reconfigurable logic regions without involving the HPS host software, but this document does not cover those methods.

Related Information

- **Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation**
- **Arria 10 GSRD v16.1 Getting Started Guide**

Prerequisites for Using This Document

Knowledge Prerequisites

Implementing a PR design requires understanding of the following subject areas:

- The Quartus Prime Pro compilation flow, especially project revisions
- SoC device capabilities
- The SoC Golden System Reference Design
- Linux device trees and device tree overlays
- Linux drivers
- The SoC boot and configuration flow

Related Information

- **Important Partial Reconfiguration Terminology** on page 42
- **"Optimizing with Project Revisions" in the Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation**
  Information about using Quartus Prime Pro project revisions
- **The Creating a Partial Reconfiguration Design chapter in the Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation**
  General information about creating PR designs with Quartus Prime Pro
- **Arria 10 GSRD v16.1 Getting Started Guide**
- **"Partial Reconfiguration of the SoC FPGA" in the Arria 10 SoC Device Design Guidelines**

- Device Tree Overlay Notes on GitHub
- FPGA Region Device Tree Binding on GitHub
- Booting and Configuration chapter in the Arria 10 Hard Processor System Technical Reference Manual

Software Tool Prerequisites

This document is based on Version 16.1 of the Quartus Prime Design Suite. The provided example requires the Intel Quartus Prime Pro Edition and the Intel SoC FPGA Embedded Design Suite (SoC EDS) tools.

The Quartus Prime Pro tools are required to perform the partial reconfiguration design flow. The PR design flow is not supported by the non-Pro Edition Quartus tools.

The SoC EDS tools provide the example hardware design that is used as a starting point for this design. The SoC EDS tools also provide the Linux build utilities required for this example.

Related Information

Arria 10 GSRD v16.1 Getting Started Guide

Hardware Prerequisites

This example is based on the 16.1 release of the SoC Golden System Reference Design (GSRD). For information about the GSRD, go to the **Getting Started** page of rocketboards.org, and look for documentation for the 16.1 SoC development kit.

An SoC Development Kit is required. It must be configured to boot from the SD card for this example. The SD card must be updated to the 16.1 version of the GSRD.

To create an SD card, follow the instructions in the *Arria 10 GSRD v16.1 Getting Started Guide* on **rocketboards.org**. Alternatively, you can use the follwoing shortcut to create an SD card in a Linux development environment:

1. Download the GSRD 16.1 binary tar file from **rocketboards.org**.
2. Extract the sdcard image as follows:

```
tar xvfz linux-socfpga-gsrd-16.1-a10-bin.tar.gz sdimage.tar.gz
tar xvfz sdimage.tar.gz
```

3. Copy the SD card image to the SD card as follows:

```
sudo dd if=sdimage.img of=/dev/{sd card device}
```

Note: The SD card must be at least 2 GB in size.

Related Information

- Arria 10 GSRD v16.1 Getting Started Guide
- A10 GSRD v16.1 - User Manual
- A10 GSRD v16.1 - Creating and Updating the SD Card
- Download the Arria 10 SoC v16.1 Golden System Reference Design

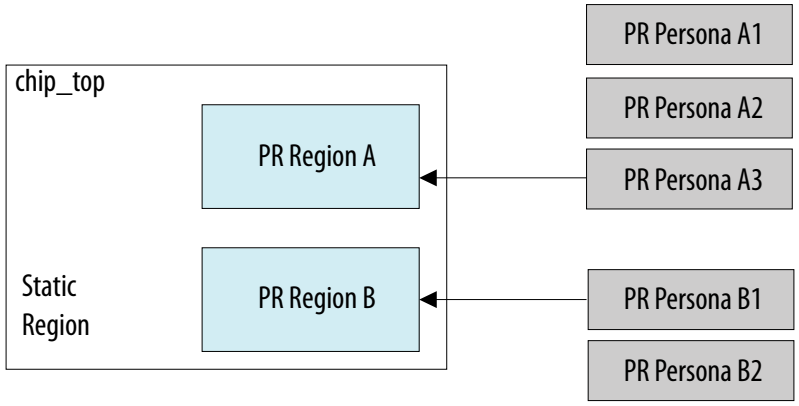
Partial Reconfiguration Tools and Methods

Hardware Tools and Methods

The PR design flow requires you to use the project revisions feature in the Quartus Prime Pro software. Your initial design is the base revision. In this revision, you define the physical boundaries in the device of the static region, which does not change under partial reconfiguration, and the reconfigurable regions, which do change.

From the base revision, you create multiple revisions. These revisions contain the different implementations for the PR regions.

Figure 1: Regions and Personas



The implementation of an FPGA design with PR results in multiple FPGA configuration image files. One file is loaded initially, and the others are subsequently loaded to reconfigure FPGA logic. The first image loaded is an FPGA configuration containing the base static region and the default implementation, or *persona*, for each reconfigurable region. One additional configuration image is generated for each persona of each reconfiguration region. Loading one of these additional files causes the reconfiguration of the associated region to the alternate persona.

The management of reconfigurable logic regions is performed most efficiently using the FPGA Manager peripheral module within the HPS block, independent of the type of host software.

PR requires logical isolation of reconfigurable FPGA regions while they are being modified. This is called freeze logic.

Related Information

- **Important Partial Reconfiguration Terminology** on page 42
- **"Optimizing with Project Revisions" in the Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation**
  Information about using Quartus Prime Pro project revisions

Arria 10 SoC Partial Reconfiguration Support

The SoC FPGA device provides an Arm*-based hard processor system (HPS) which can be used to run Linux, a real time operating system (RTOS), or a bare-metal software stack. The HPS, running appropriate software, can act as the PR host, and efficiently manage the reconfigurable FPGA logic regions.

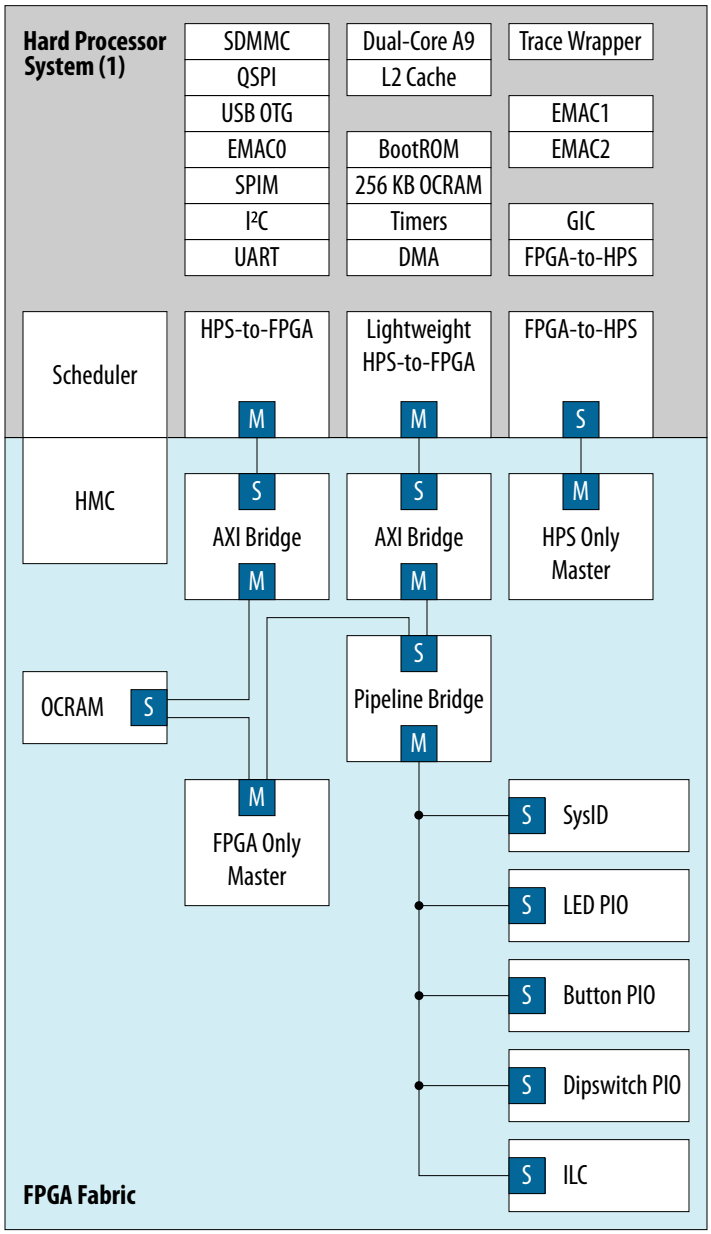
Related Information

Important Partial Reconfiguration Terminology on page 42

Overview of the Arria 10 GHRD

The Golden Hardware Reference Design (GHRD) is an example design released as part of the SoC EDS. The example demonstrates a basic Quartus Prime design for the SoC Development Kit using the Qsys design tool and HPS component.

Figure 2: Golden Hardware Reference Design Block Diagram



(1) Components not used in the GHRD are not shown

The GHRD provides several example modules connected to the data bus, including PIO and system ID modules.

The hardware example that you create with this document is based on the GHRD. You add a new branch of the data bus to connect to a reconfigurable region.

Software Tools and Methods

The host software can be based on Linux, an RTOS, or a bare-metal software stack. The example in this document is Linux-based.

In Linux v. 4.1.22-ltsi and later, Linux kernels use device tree overlays to manage drivers for devices that can be dynamically added or removed at runtime.

The device tree is a data structure for describing hardware. Rather than hard coding every detail of a device into the operating system, many aspects of the hardware can be described in a data structure that is passed to the operating system at boot time.

A device tree overlay is a similar data structure that is loaded dynamically at runtime to modify the system hardware description. Linux device drivers are loaded or unloaded when device tree overlays are applied to or removed from a running kernel.

Linux can optionally control freeze logic when the system hardware description is modified with device tree overlays.

The Quartus Prime Pro Edition software supports the PR feature for the device family.

Arria 10 SoC Partial Reconfiguration Workflow

Fully implementing PR for the SoC FPGA, using the Arm processor-based Hard Processor System (HPS) to manage reconfigurable regions, requires coordination between the FPGA logic designer and the HPS host software developer. Certain information must be provided by the FPGA logic designer to the host software developer in order for PR management to work correctly and reliably with the HPS.

Hardware Workflow

The PR hardware design flow requires initial planning. This planning involves:

- Planning design partition(s), logical divisions in the source code hierarchy. Well-planned PR partitions improve design area utilization and performance.
- Determining the placement assignments in the floorplan (the physical design layout on the device).

After you have planned the partitions and floorplan, you are ready to set up the design hierarchy and source code to support this partitioning.

Related Information

- **Important Partial Reconfiguration Terminology** on page 42
- **The Creating a Partial Reconfiguration Design chapter in the Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation**
  General information about creating PR designs with Quartus Prime Pro
- **Create Design Partitions for Partial Reconfiguration**
  For more information about defining physical and logical partitions, refer to "Create Design Partitions for Partial Reconfiguration" in the *Creating a Partial Reconfiguration Design* chapter of the *Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation*.

Software Workflow

To create software for an SoC partial reconfiguration design, you create a device tree overlay that describes the static portion of your design (the base revision). Then, you create device tree overlays for personas in each PR region.

In this application note, you create and run a PR software example based on the GHRD. You start with a base device tree for the SoC, distributed with your kernel. Then you carry out steps to adapt this device tree to the PR example design.

Partial Reconfiguration Limitations

Reconfigurable partitions can contain only core resources, such as LABs, embedded memory blocks (M20Ks and MLABs), and DSP blocks in the FPGA. All periphery resources, such as transceivers, external memory interfaces, GPIOs, I/O receivers, and hard processor system (HPS), must be in the static portion of the design. Partial reconfiguration of global network buffers for clocks and resets is not possible.

Creating the PR Example Design

Qsys Partial Reconfiguration Freeze Logic

The freeze logic used by this example consists of two modules:

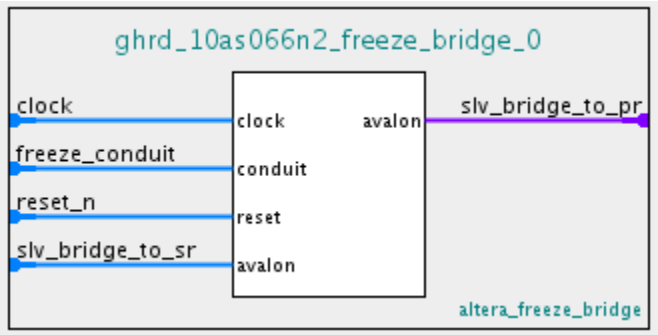
- A freeze bridge that contains isolation, reset, and handshake logic for the reconfigurable region. Each freeze bridge controls a single interface to the PR region. A PR region can require more than one freeze bridge.
- A freeze controller that provides a defined register interface for software to control one or more freeze bridges.

The example design uses both of these modules to provide isolation and freeze logic to the reconfigurable portions of the design. The device tree overlay describes these modules so that Linux can automatically control the logic during PR.

PR Freeze Bridge

A freeze bridge is designed for a specific interface, such as Avalon-MM or AXI. The Intel partial reconfiguration freeze bridge, provided with the Qsys Pro tools, is shown below. It is an example Avalon-MM freeze bridge.

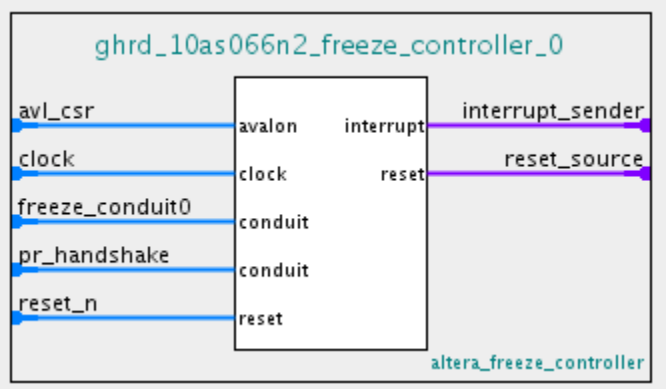
Figure 3: Partial Reconfiguration Freeze Bridge



PR Freeze Controller

The Intel partial reconfiguration freeze controller, provided by the Qsys Pro tool, is shown below.

Figure 4: Partial Reconfiguration Freeze Controller



One freeze controller can control multiple freeze bridges.

Importing the GHRD Project

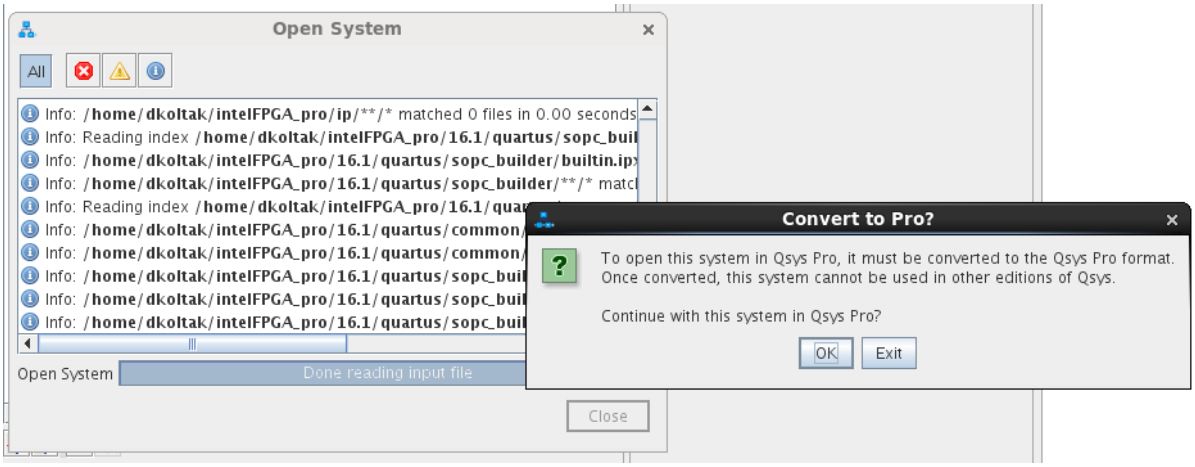
1. Extract the `.tar` file provided with the SoC EDS GHRD hardware example, as follows:

```
$ mkdir ghrd_pr
$ cd ghrd_pr/
$ tar xvfz ~/intelFPGA/16.1/embedded/examples/hardware/ \
a10_soc_devkit_ghrd/tgz/ghrd_10as066n2_16_1_*.tar.gz

(…)

$ which quartus
```

```
~/intelFPGA_pro/16.1/quartus/bin/quartus
$ quartus &
```

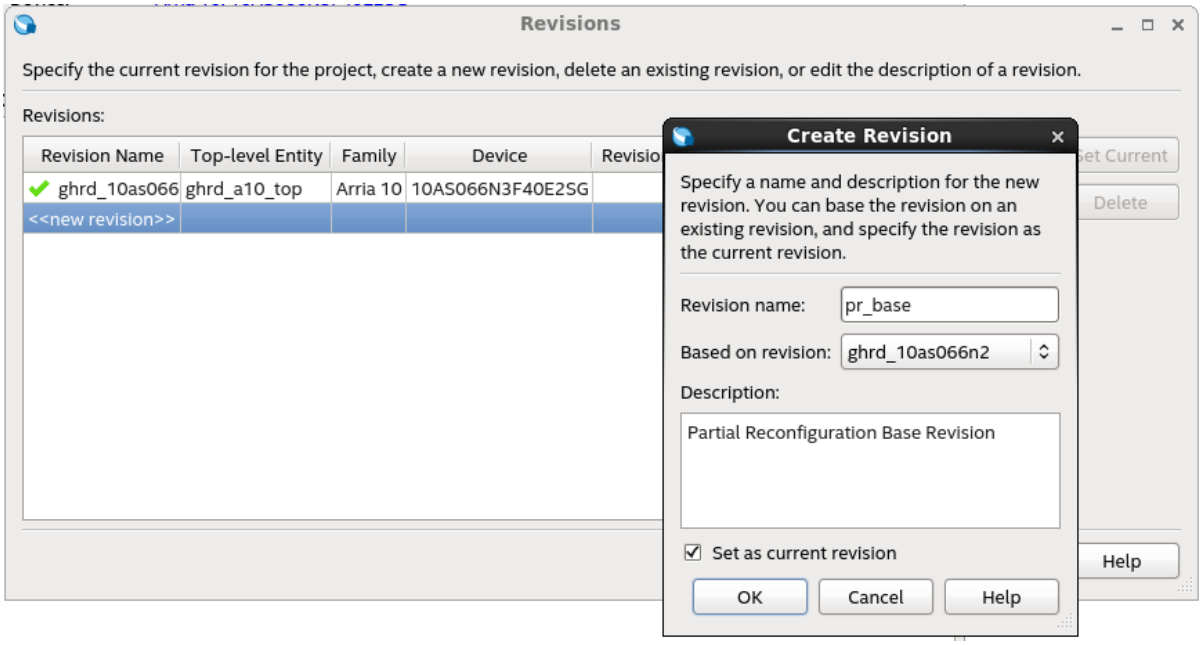
2. Open the GHRD Quartus project file.
3. Open the GHRD Qsys design saved in the `ghrd_10as066n2.qsys` file with the Qsys Pro tools. Point to **Tools** and click **Qsys Pro** to start the tool.
4. The copied design is in the standard Qsys tool format and must be converted to the Qsys Pro format. A dialog box confirming this action pops up automatically after opening the Qsys design. Click **OK** to perform the conversion.

Figure 5: Converting to Qsys Pro Format



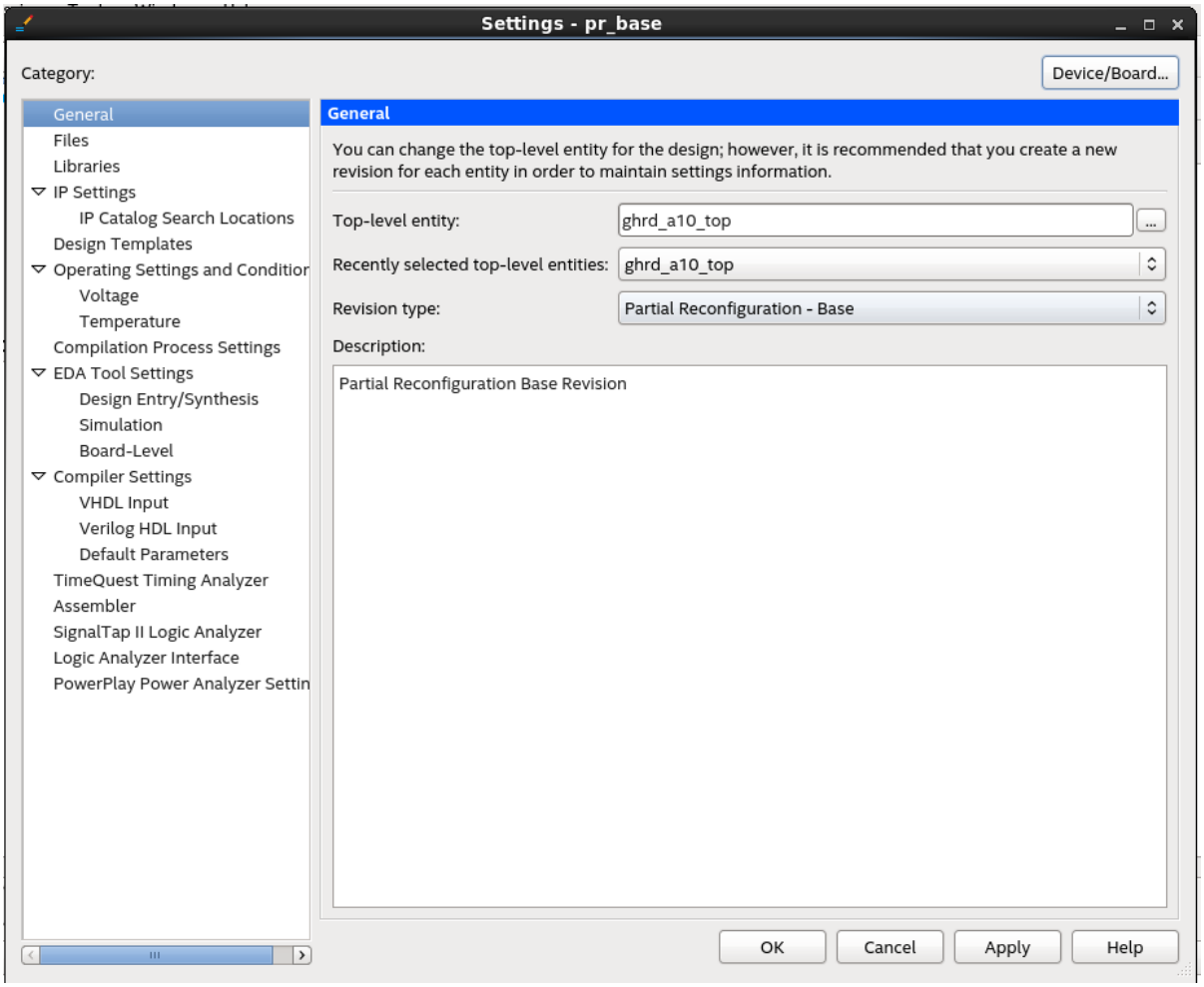
5. Save the converted design and click **Generate RTL** to ensure that the conversion process was successful.
6. Close the Qsys Pro tool.
7. In the main Quartus window, point to **Project** and click **Revisions**. Create a new revision for the PR base configuration. Double click **<< new revision >>** and enter the revision information as shown below. This revision is based on the existing revision.

Figure 6: Creating the Base Revision



8. Click **OK** to create the revision and open it as the current revision.
9. To set the revision type of the new revision, point to **Assignments**, click **Settings**, choose the **General** category, and change the **Revision Type** field to **Partial Reconfiguration – Base** as shown below.

Figure 7: Setting the Revision Type



Add a Partial Reconfiguration Region to the GHRD

All PR implementation revisions use the top-level placement and routing results from the base revision.

The base and alternate personas are created as separate Qsys modules. As a result, each PR region is a hierarchical logic grouping, which can be designated as a design partition and a LogicLock Plus region. Intel strongly recommends that you follow this practice in your own design, so that there is a clear division between static and dynamic regions.

Creating the Partial Reconfiguration Design

This design contains all the logic for the default persona of the PR region that is added to the GHRD Qsys design later.

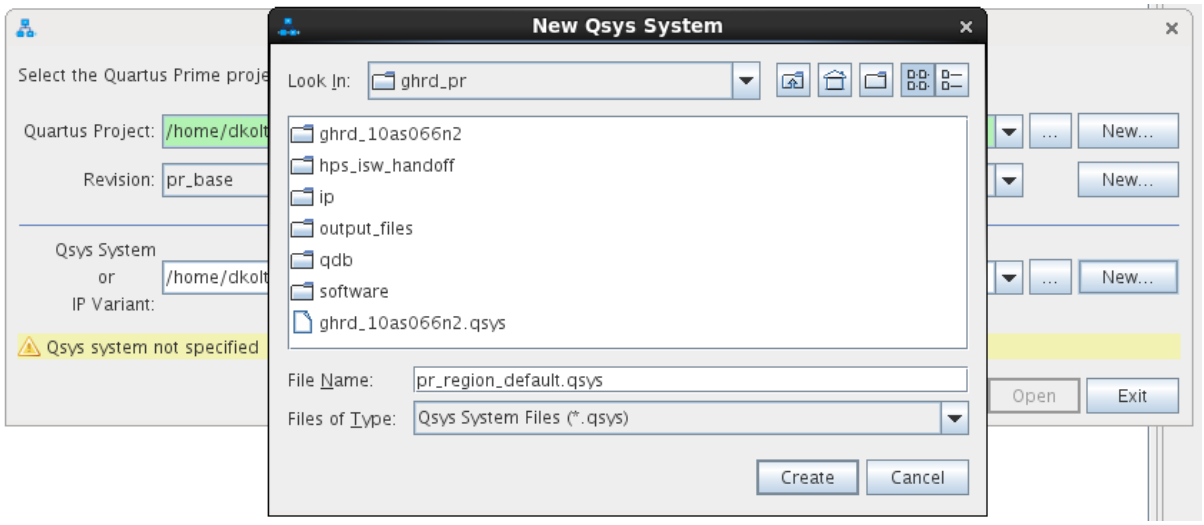
The PR region contains three components for this example:

- An Avalon-MM pipeline bridge, to split the bus address space for the other two components and isolate bus timing.
- Memory to access from the Avalon-MM bus.
- A system ID peripheral component. This component is accessible from the Avalon-MM bus and provides a static version number that can be read by software to determine what logic has been loaded into the PR region.

To create the PR design, perform the following steps:

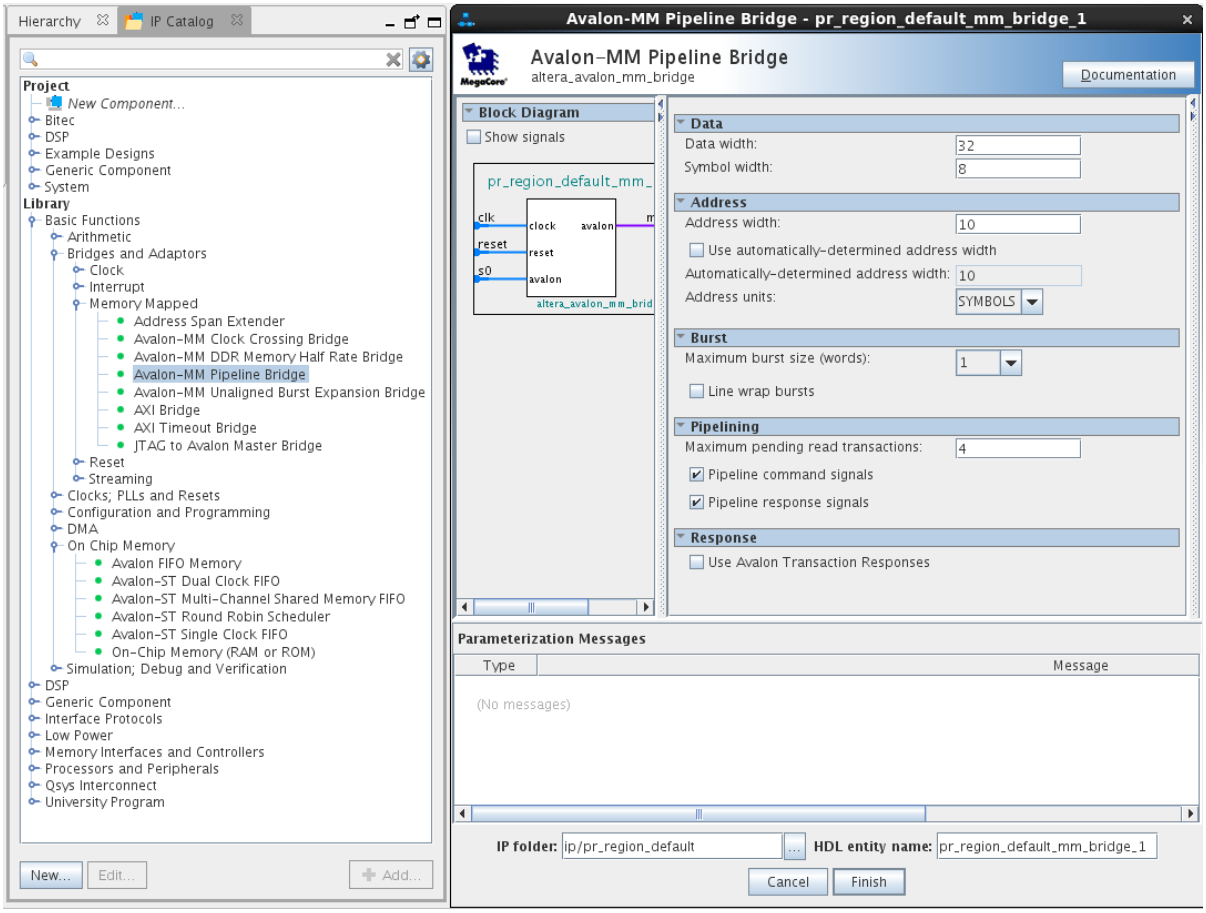
1. Open the Qsys Pro tool again and create a new system design named `pr_region_default.qsys`.

Figure 8: Creating the Qsys Project for the Partial Reconfiguration Design



2. Add an Avalon-MM pipeline bridge to the design as shown below.

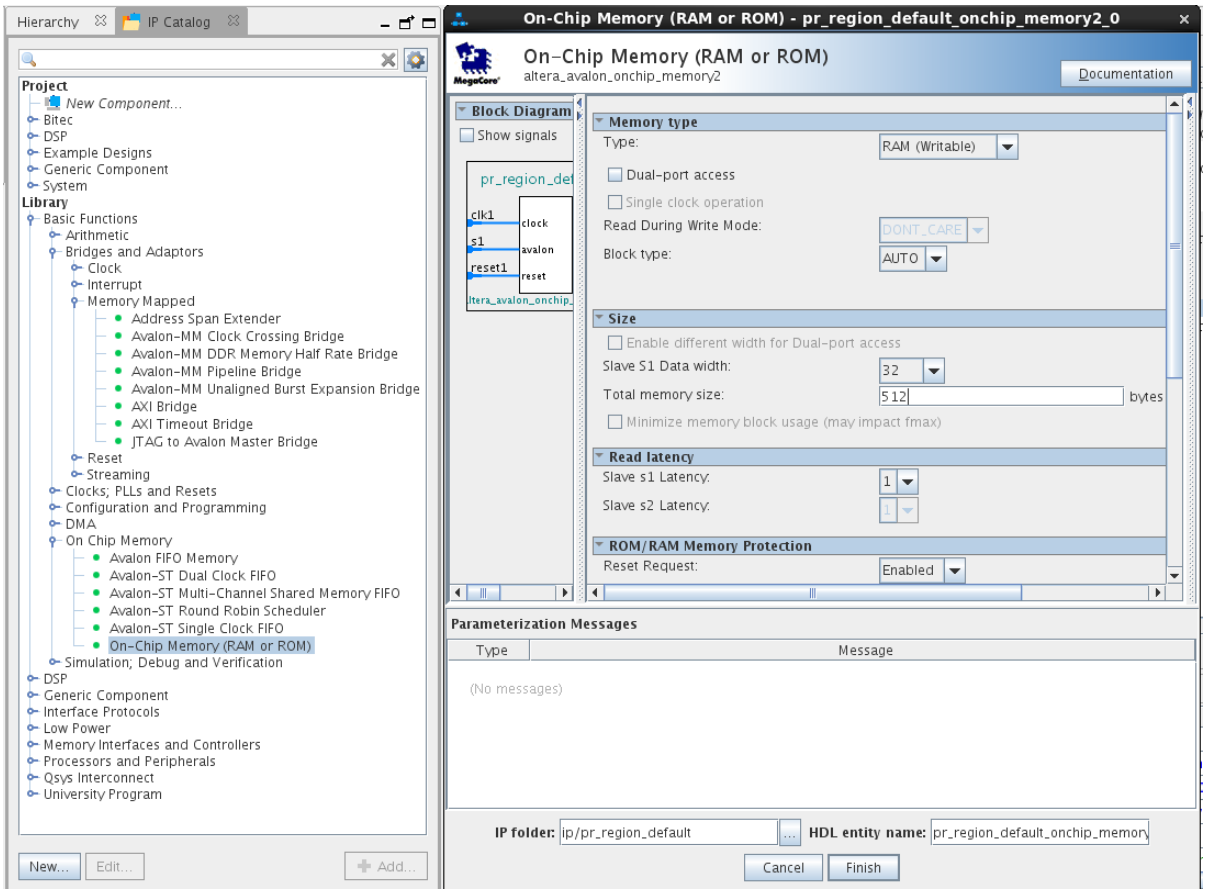
Figure 9: Adding the Avalon-MM Pipeline Bridge



3. Add a 512 byte, 32-bit wide on-chip memory as shown below.

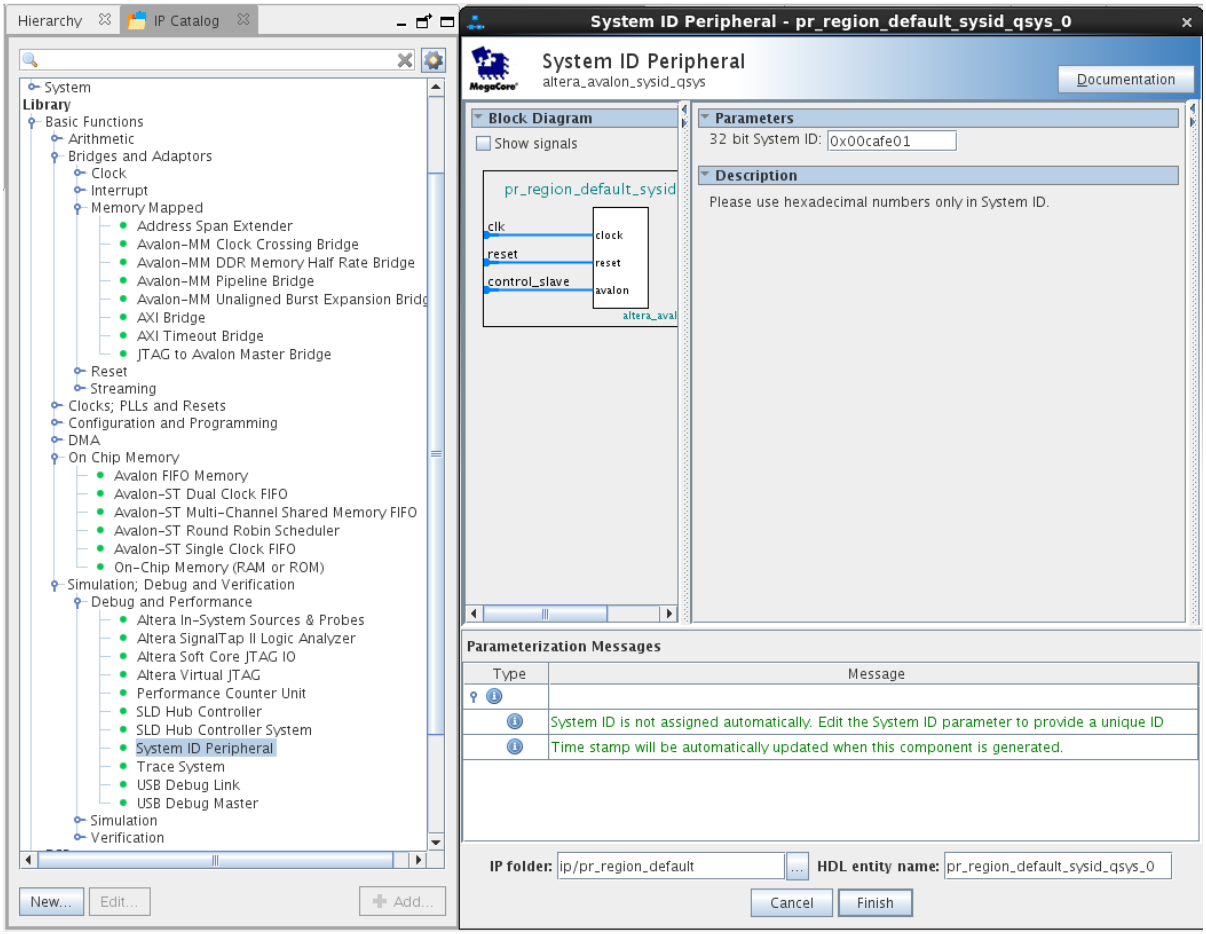
   Note: Scroll down and turn off the **Initialize Memory Content** option. Otherwise, Qsys will require an initialization data file.

Figure 10: Adding On-Chip Memory



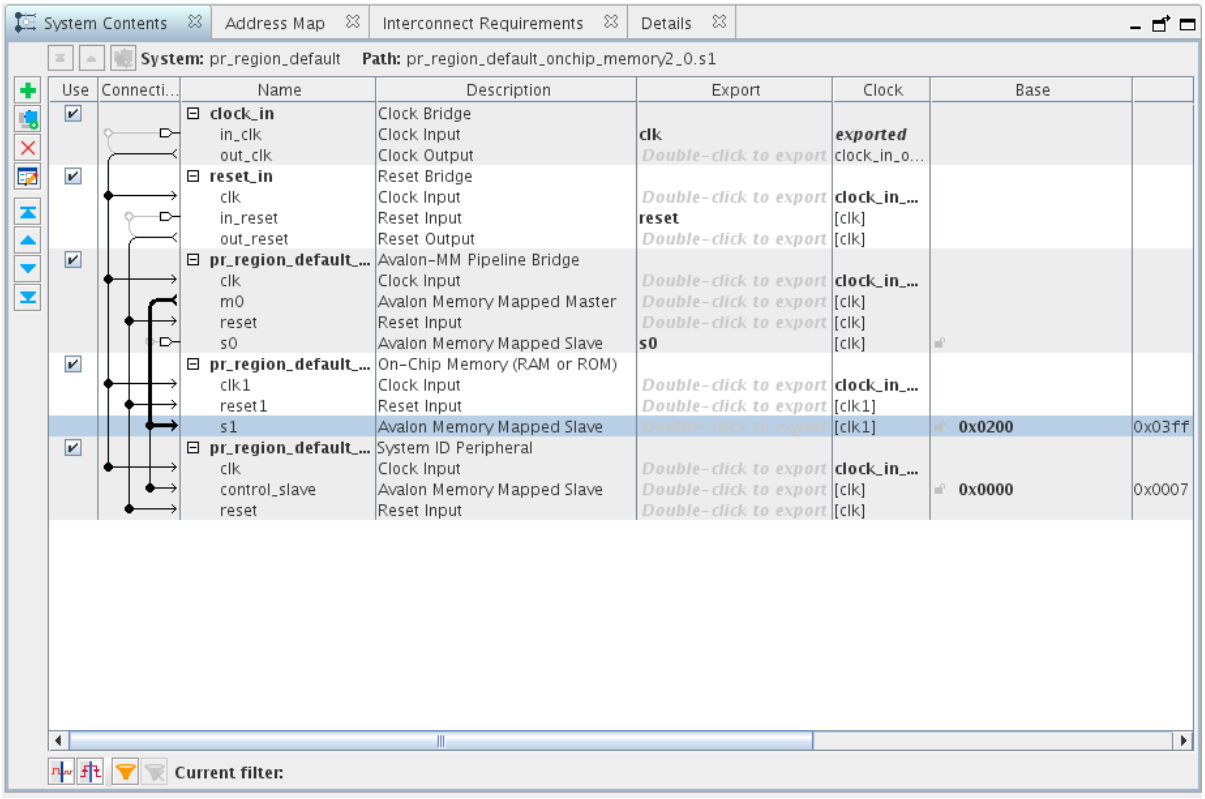
4. Add a system ID peripheral component as shown below. Set the **32-bit System ID** field to something unique.

Figure 11: Adding the System ID



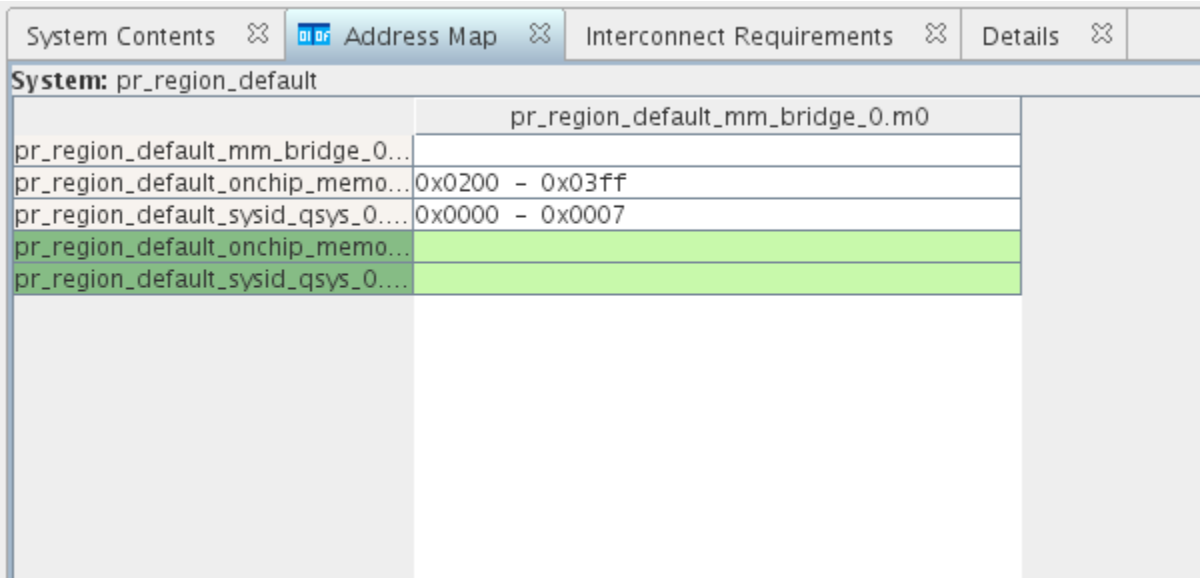
5. Hook up the Qsys system as shown below by clicking on the appropriate connection bubbles.

Figure 12: Configuring the System Connections



6. Export the Avalon-MM pipeline bridge slave port and give it an exported name of `s0`.
7. Change the on-chip base address to 0x0200 and leave the system ID peripheral base address at 0x0000.
8. Confirm the address map is correct by reviewing the configuration in the **Address Map** tab. It should look like the configuration shown below.

Figure 13: Address Map Example

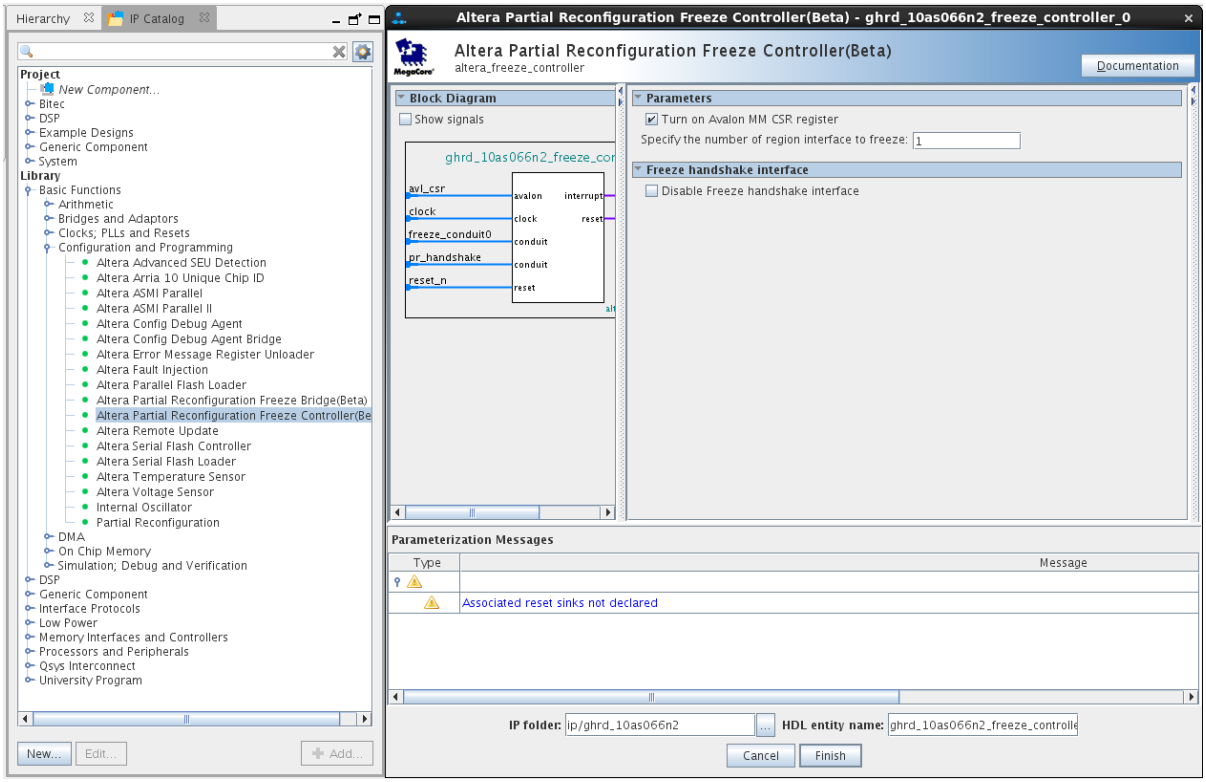


9. Save the design and click **Finish**.

Adding the PR Region

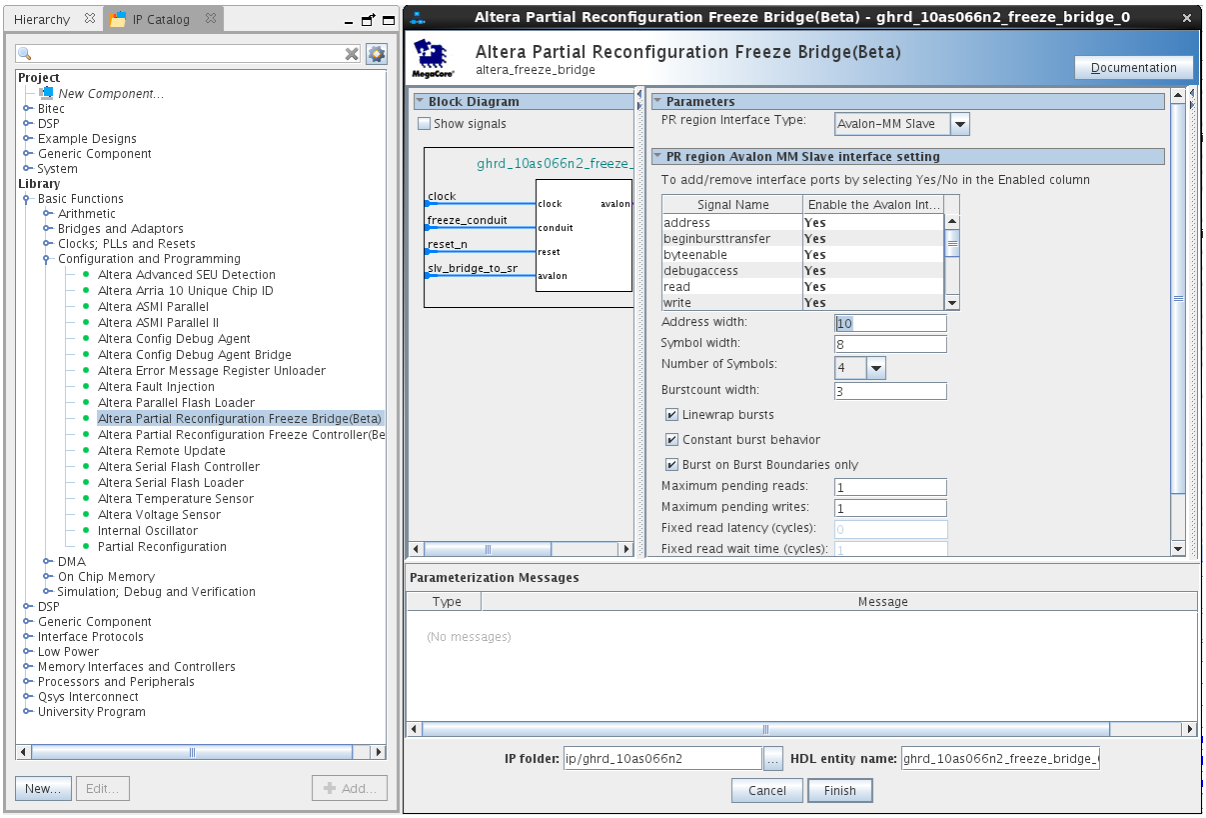
1. Reopen the GHRD Qsys system design.
2. Add an Intel partial reconfiguration freeze controller.

Figure 14: Adding the Partial Reconfiguration Freeze Controller



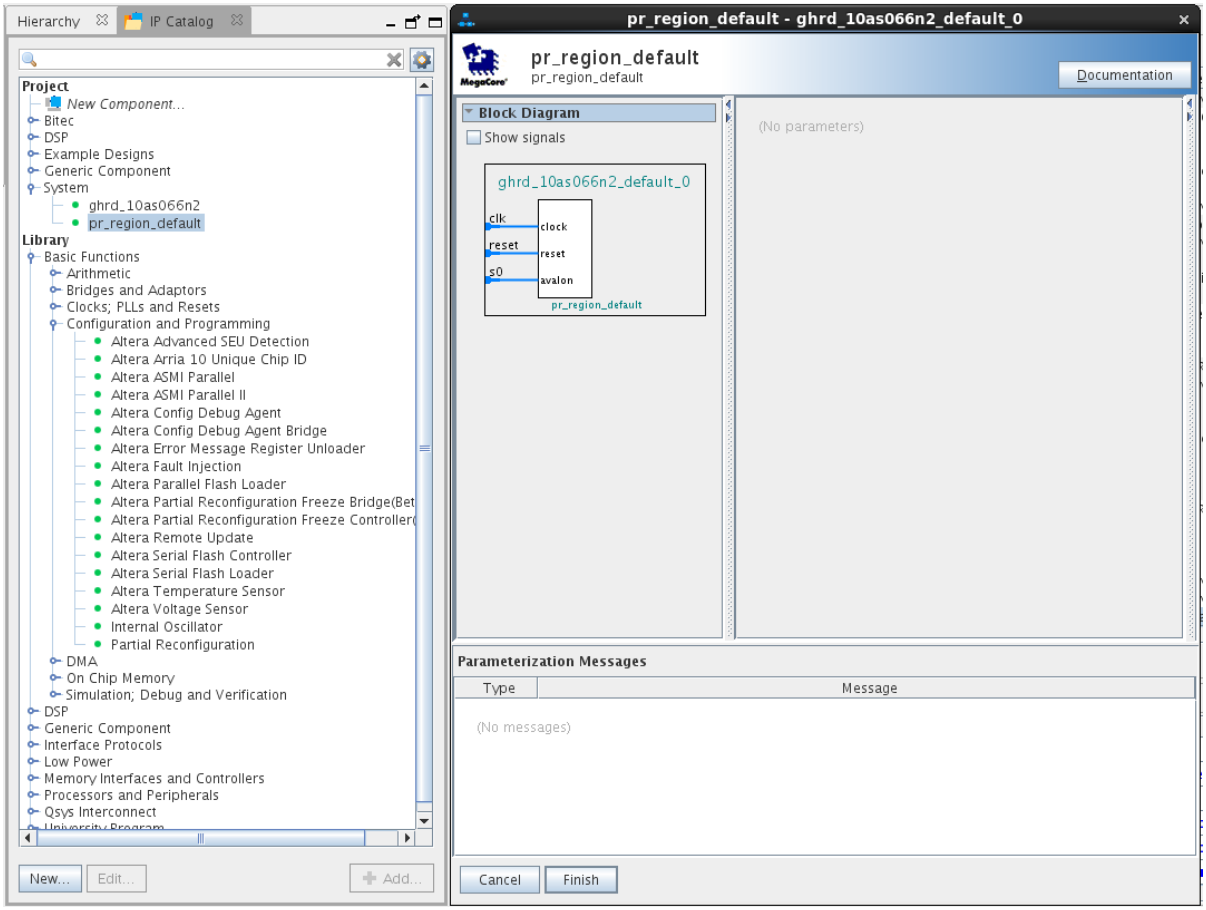
3. Add an Intel partial reconfiguration freeze bridge.

Figure 15: Adding the Partial Reconfiguration Freeze Bridge



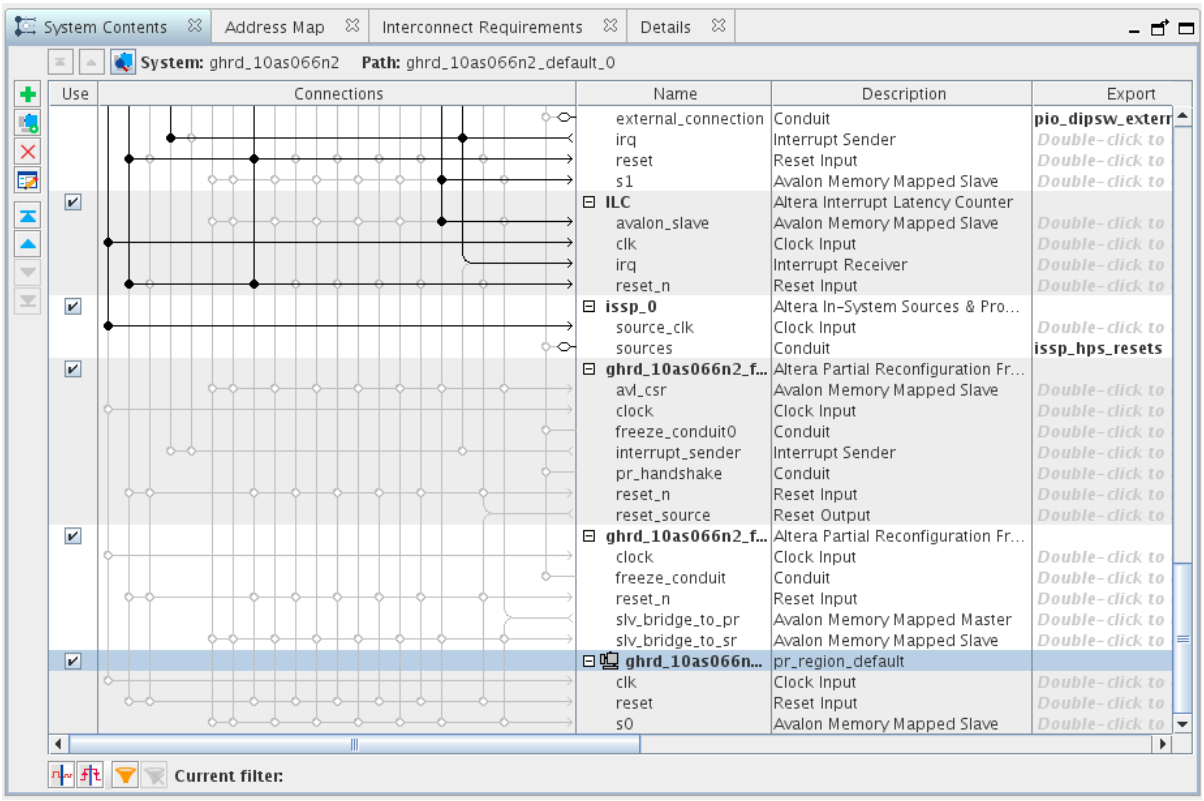
4. Add the PR Region Default Qsys system to the design as shown below. The previously created pr_region_default Qsys system is in the **System** category of the IP catalog.

Figure 16: Adding the PR Region Default Qsys System



5. Confirm the additions to the GHRD Qsys system design by comparing the **System Contents** tab shown below.

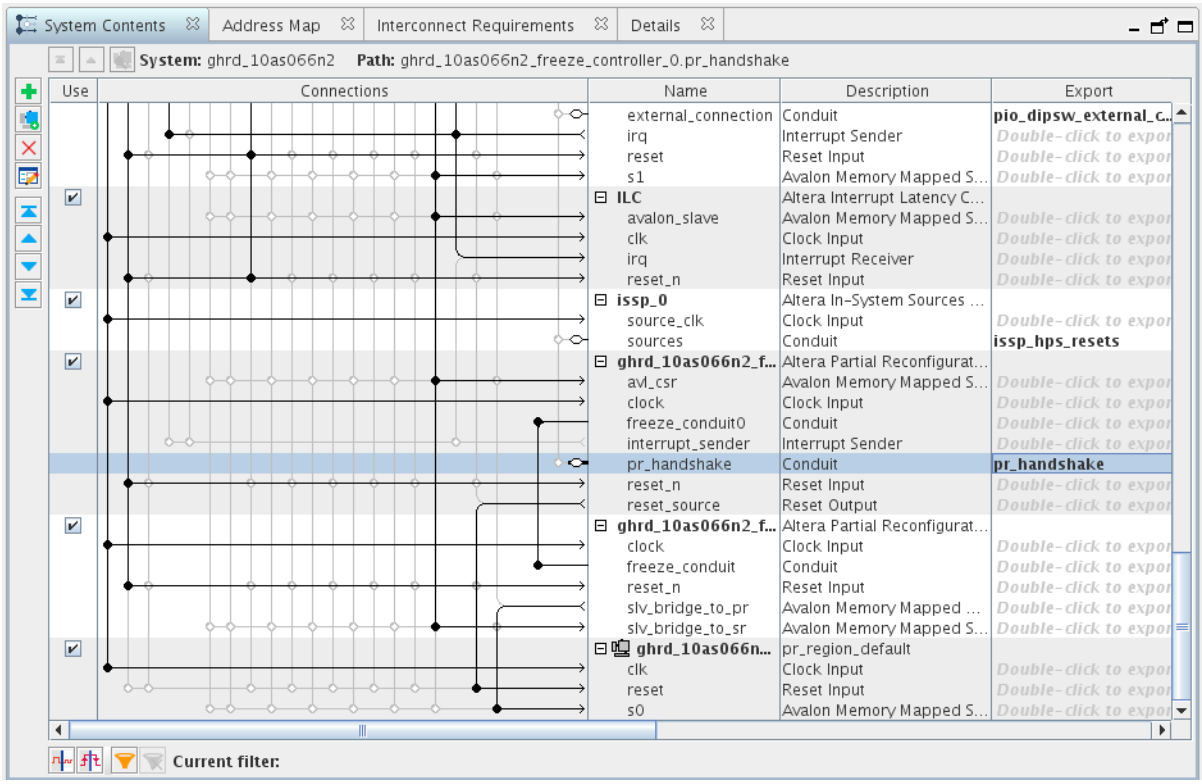
Figure 17: Resulting Design



6. Connect the new components as shown below. Export the `pr_handshake` conduit interface of the Intel partial reconfiguration freeze controller to the top level with the name `pr_handshake`.

   Note: The freeze controller's **interrupt_sender** output signal is disconnected, because the Linux driver polls a status bit rather than using an interrupt. You might see a Qsys warning message, which you can ignore.

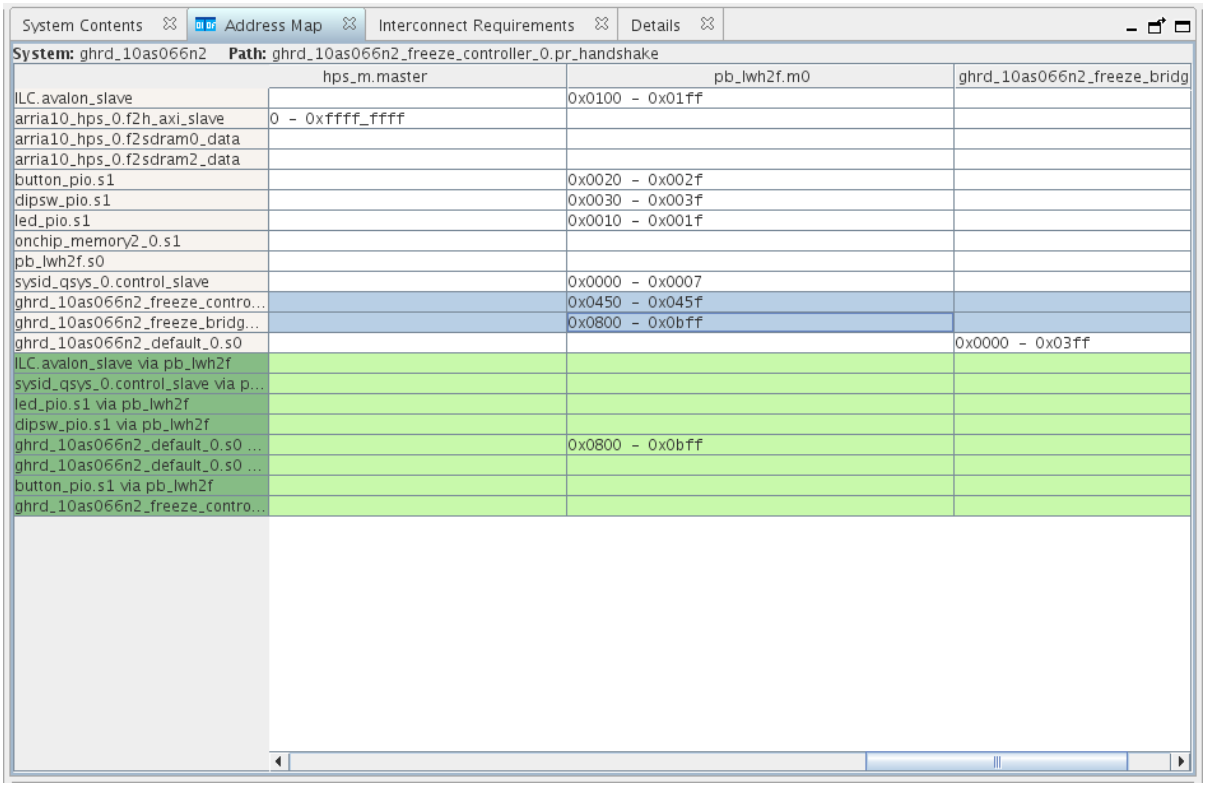
Figure 18: PR Region Connections



7. Edit the system address memory map using the **Address Map** tab as shown below.

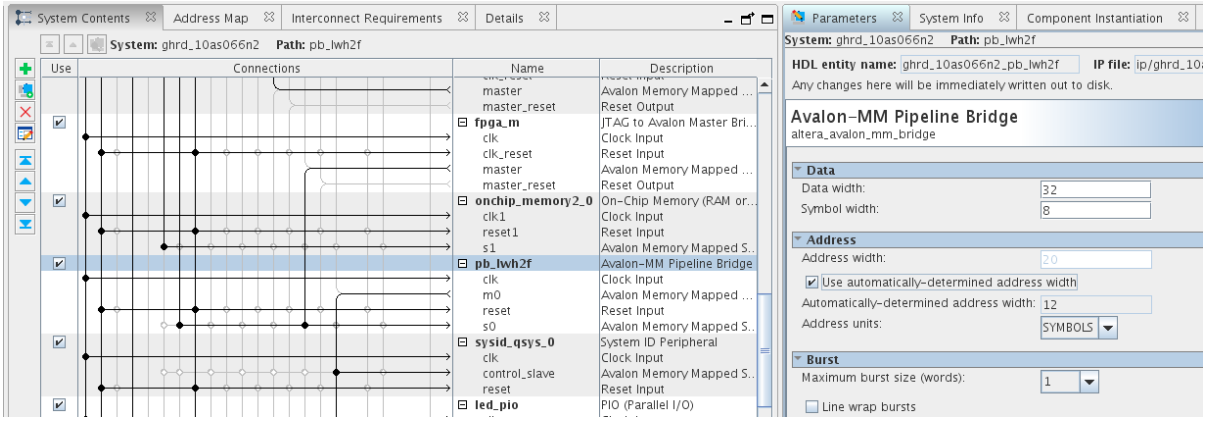
   Note: Pay attention to the freeze controller and freeze bridge base addresses. These are crucial parts of the handoff information provided to the host software designer.

Figure 19: System Address Memory Map



8. Force the `pb_lwh2f` address width to automatically readjust by unchecking, and then re-checking the **Use automatically-determined address width** box in the component dialog. Qsys sets the address width to 12 bits.

Figure 20: Pipeline Bridge Data Width



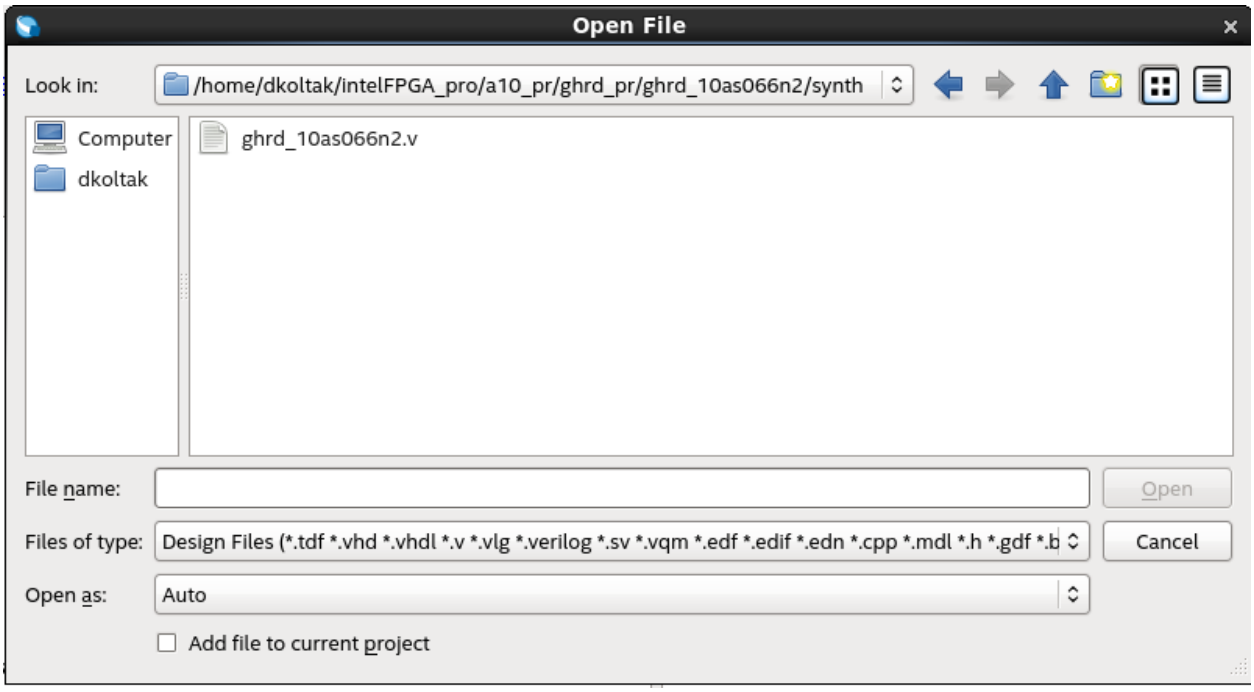
9. Adjust the address map to eliminate any overlaps or alignment issues.
10. Click **Generate RTL** to create the new system.
11. Click **Finish**.

Connecting New Ports in the GHRD Module Instantiation

In an earlier step, you added new ports to the top level GHRD module, by exporting the `pr_handshake` interface conduit. You must connect these ports in the GHRD module instantiation before the design can be synthesized.

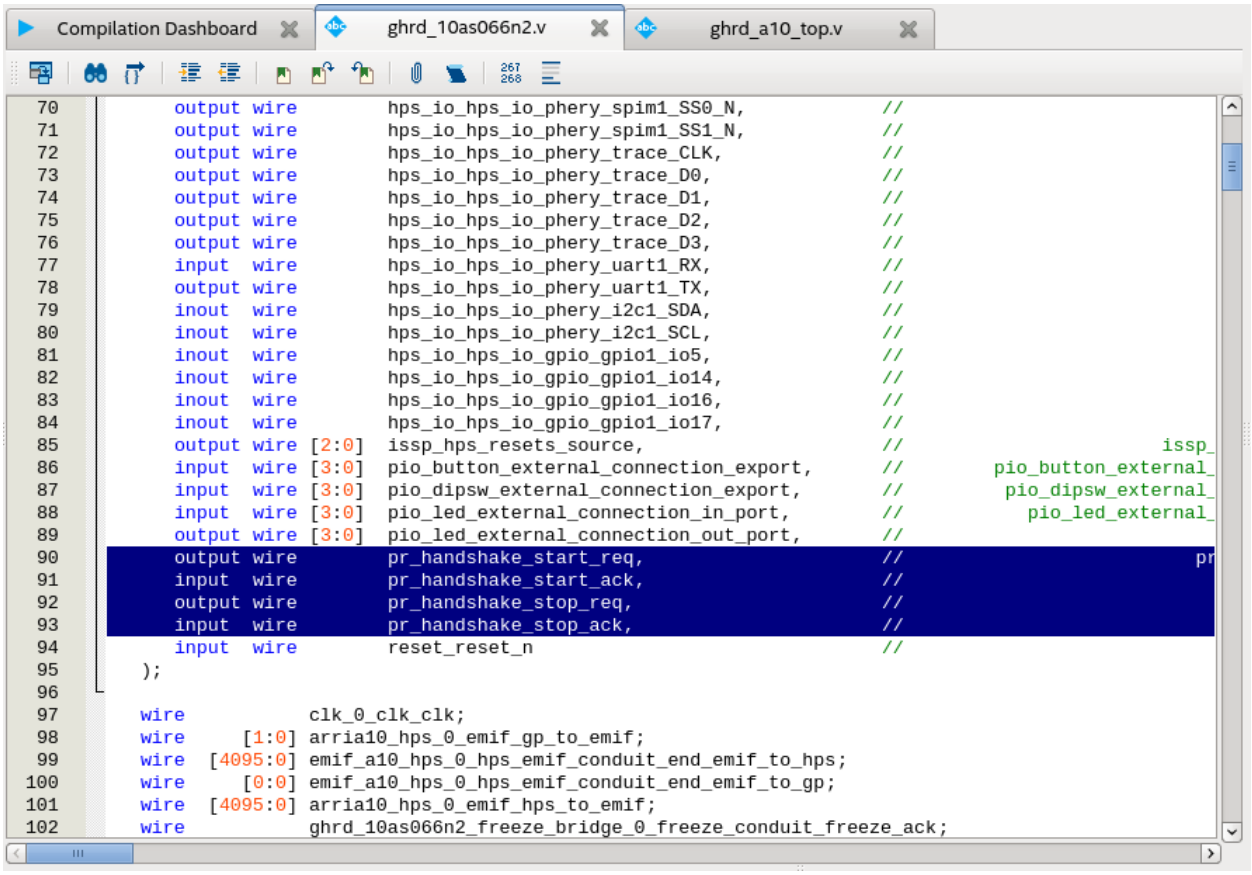
Open the GHRD top level module, which you generated in earlier steps, as shown below.

Figure 21: Opening the GHRD Top Level Module



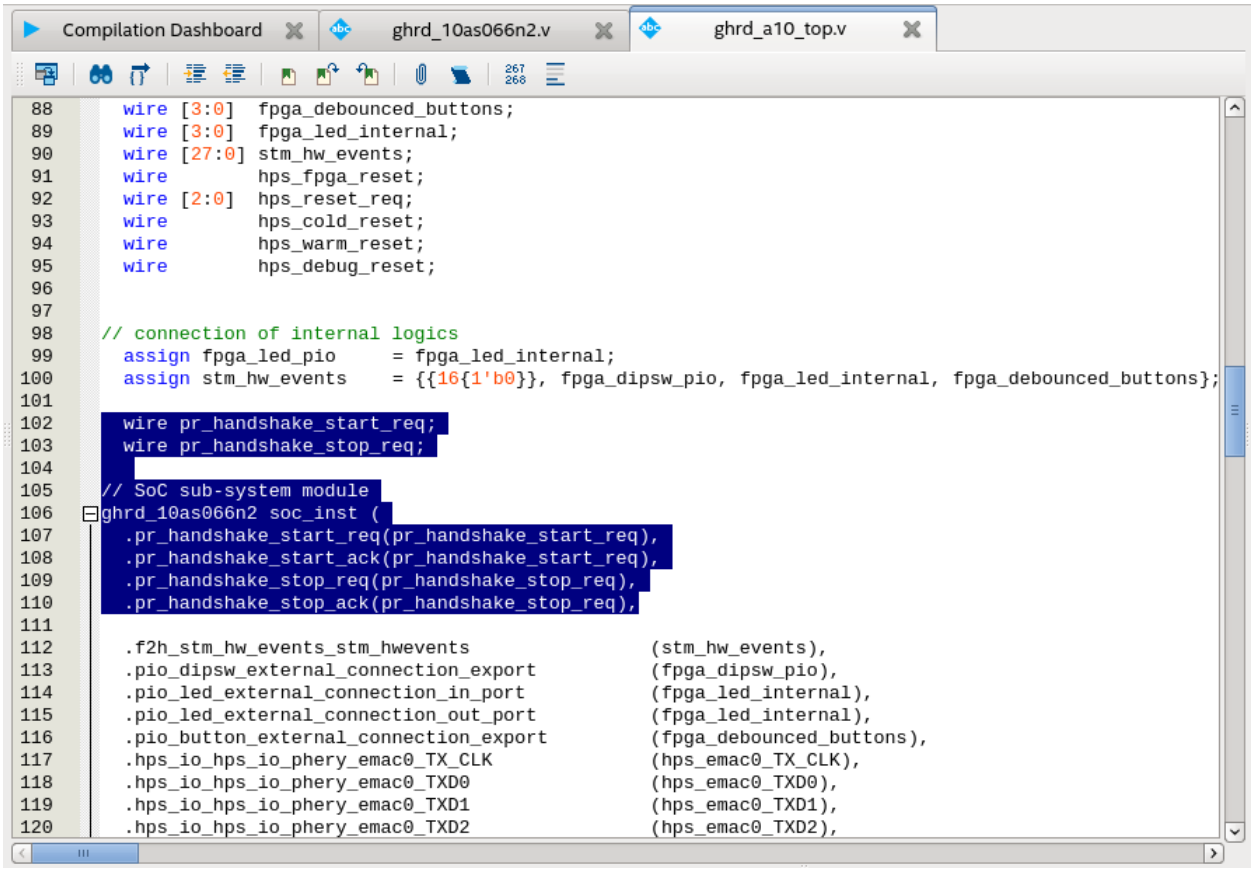
The new `pr_handshake` ports are highlighted below. The conduit interface exported in the Qsys design consists of the four PR Start and End request/acknowledge handshake signals shown.

Figure 22: Partial Reconfiguration Handshake Signals



The PR Start and End request/acknowledge handshake signals are required by the Intel partial reconfiguration freeze controller. You must directly connect these signals in the top level design for proper freeze controller functionality. The connection is shown below.

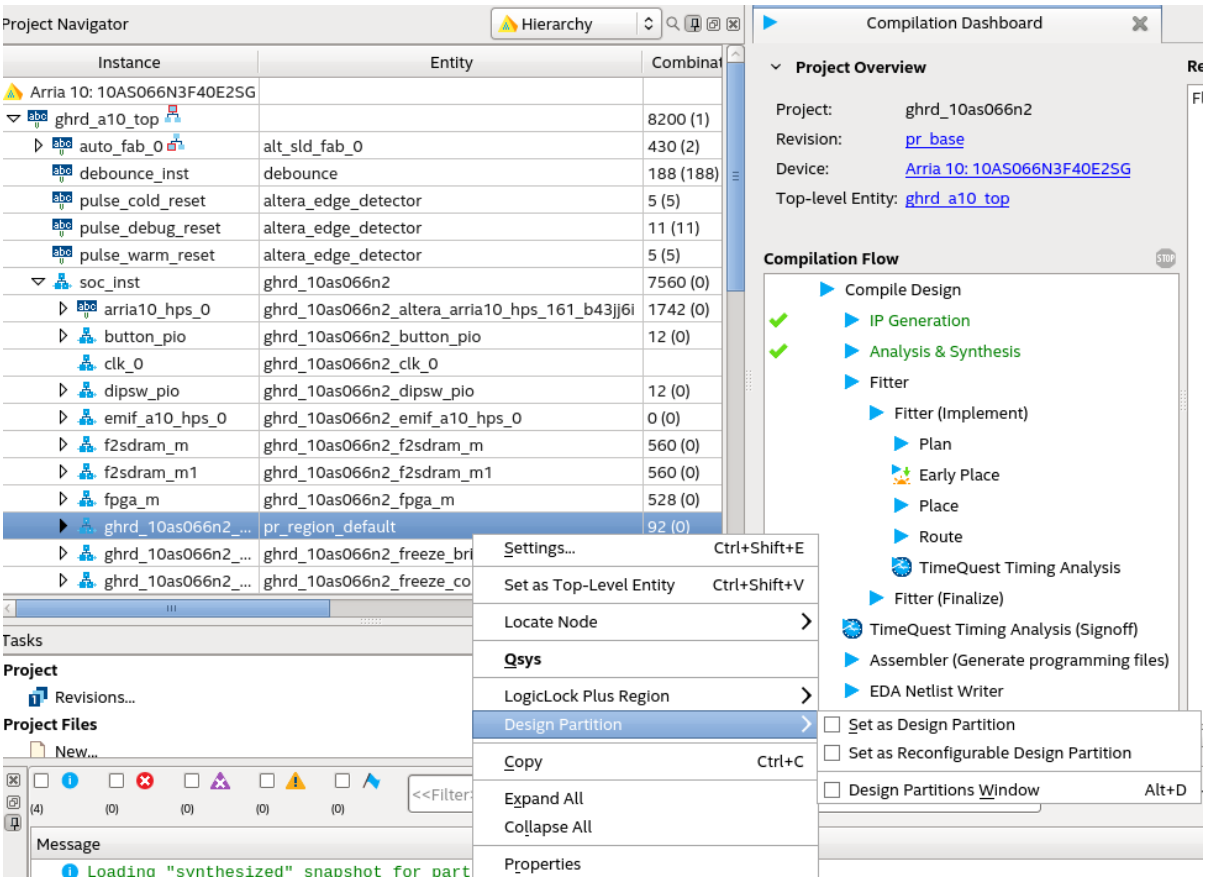
Figure 23: Handshake Signal Connections



Building the Base Revision with the Reconfigurable Design Partition

1. Run the first two stages of the compilation flow (IP Generation and Analysis & Synthesis). This generates the design hierarchy so that the reconfigurable design partition and LogicLock Plus assignments can be added.
2. To create a reconfigurable design partition, right-click the design instance of the PR region, point to **Design Partition**, and click **Set as Reconfigurable Design Partition** as shown below.

Figure 24: Creating a Reconfigurable Design Partition

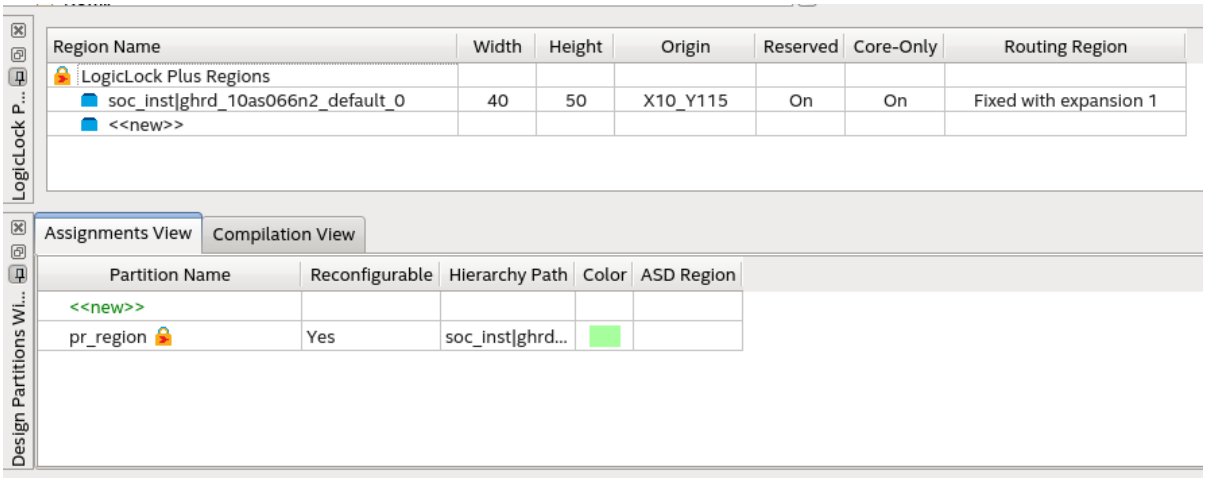


3. To create a new LogicLock Plus region, similarly right-click the design instance, point to **LogicLock Plus Region**, and click **Create New LogicLock Plus Region**.
4. Set the properties of the design partition and LogicLock Plus region using the editor windows.(1) Edit the assignments as shown below. Change the design partition name to something generic, like "pr_region". Make sure to adjust the LogicLock Plus placement, enable the **Reserved** and **Core-Only** attributes, and change the **Routing Region** type to **Fixed with expansion of 1**.

   For detailed information about design partition settings, refer to "Create Design Partitions for Partial Reconfiguration" in the *Creating a Partial Reconfiguration Design* chapter of the *Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation*.

(1) If the editor windows are not visible, use the **Assignments** drop down menu to enable them.

Figure 25: Configure Partition and Region Properties



5. Run a full compilation flow as follows:
   a. If the Compilation Dashboard is not visible, click **Compilation Dashboard** in the Tasks window.
   b. Click **Compile Design**.

   The following critical files are in the `output_files/` directory after the compilation flow finishes:

   - `pr_base.sof`
   - `pr_base.pr_region.pmsf`

   These files contain the FPGA configuration images for the default complete FPGA and the PR region default persona, respectively. These files are converted to RBF images in a later step.

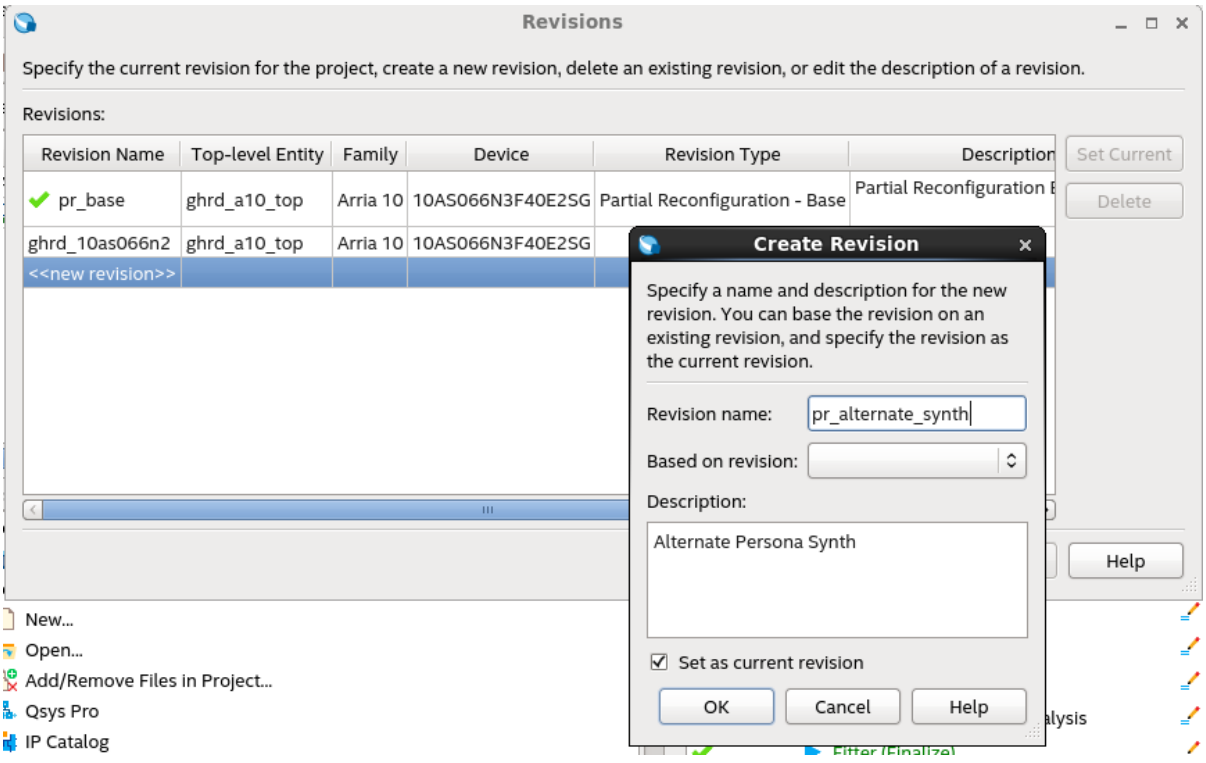
Related Information

- **Create Design Partitions for Partial Reconfiguration**
  For more information about defining physical and logical partitions, refer to "Create Design Partitions for Partial Reconfiguration" in the *Creating a Partial Reconfiguration Design* chapter of the *Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation*.
- **Full Compilation Flow**
  For steps to run a full compilation, refer to "Full Compilation Flow" in the *Design Compilation* chapter of the *Quartus Prime Pro Edition Handbook Volume 1: Design and Compilation*.

Synthesizing an Alternate Persona

1. Create a new design revision for the alternate persona synthesis step. This revision is not based on any existing revision and is created blank with no existing files, settings, or constraints. This revision is only for synthesizing the PR region logic and does not require any of the normal top level settings or constraints, such as pin placement.

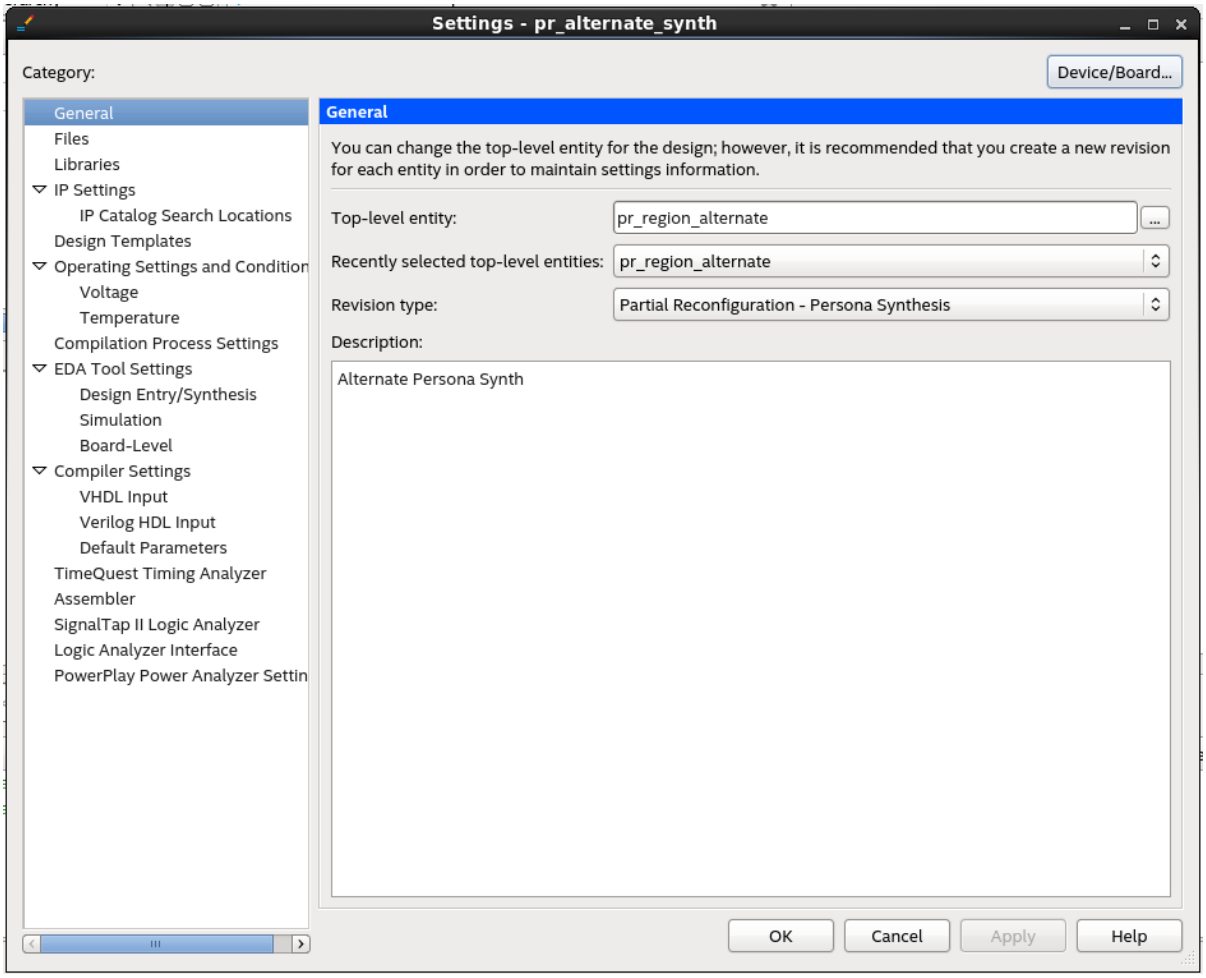
Figure 26: Creating Blank Revision for Alternate Persona



Click **OK** to create the revision and open it as the current revision.

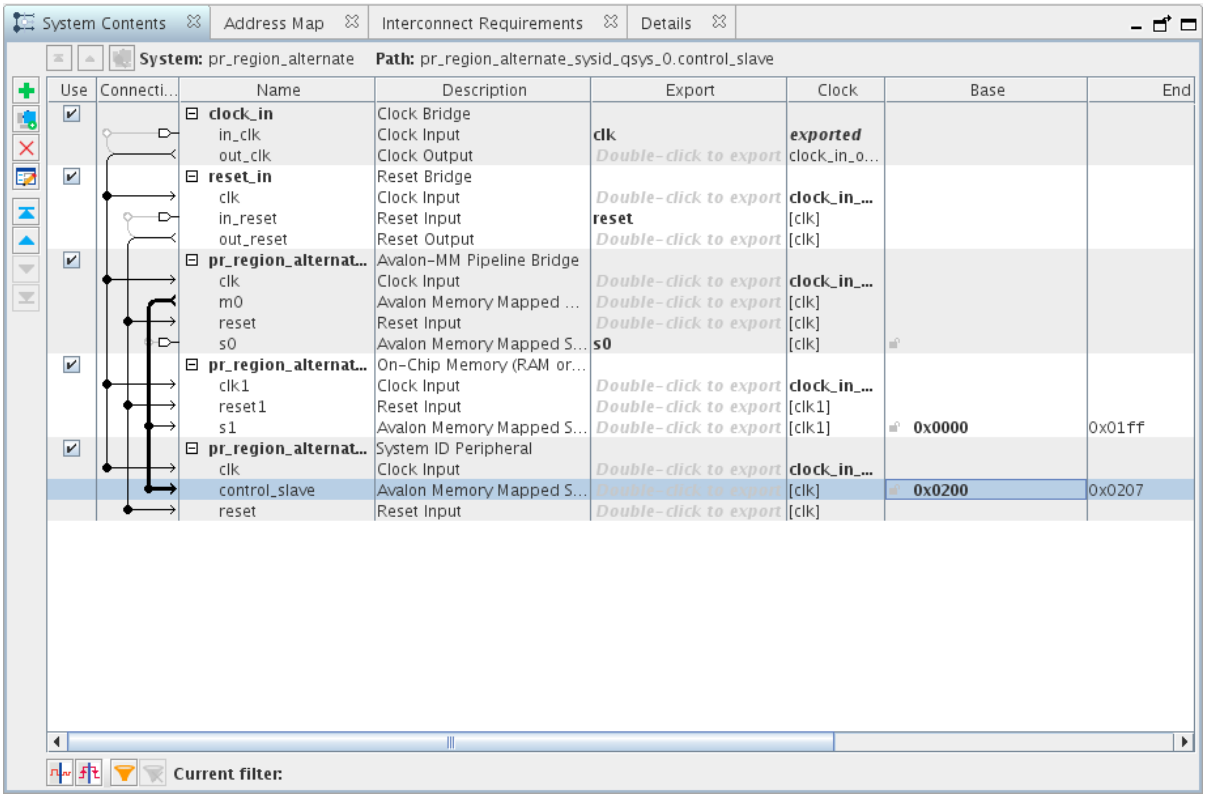
2. Ensure that the new revision's **Device Type** setting matches the other revisions. If it does not, you can change it by clicking **Device** in the Assignments menu, to open the **Device** dialog box.
3. To change the revision type of the new revision, point to **Assignments**, click **Settings**, choose the **General** category, and change the **Revision Type** field to **Partial Reconfiguration – Persona Synthesis** as shown below.

Figure 27: Setting the Revision Type



4. Change the **Top-level entity** field to `pr_region_alternate`. This is the name of a new Qsys design that is generated in a later step.
5. Create a new Qsys Pro design called `pr_region_alternate.qsys` as shown below.

Figure 28: New Design for Alternate Persona



6. Add an Avalon-MM pipeline bridge, an on-chip memory, and a system ID peripheral, as described in "Creating the Partial Reconfiguration Design". However, configure a different value in the 32-bit system ID field of the system ID peripheral, and set the base address of the system ID peripheral to 0x0200. Leave the on-chip memory base address at 0x0000.

   At runtime, software will read the system ID from each design revision. For each PR region, it will read a unique value from a unique address. This will demonstrate that the PR region has changed and Linux was updated properly.
7. Click **Generate RTL** and then **Finish**. This closes the Qsys Pro design.
8. Run the first two stages of the compilation flow (IP generation and analysis and synthesis), as follows:
   a. If the Compilation Dashboard is not visible, click **Compilation Dashboard** in the Tasks window.
   b. Click **IP Generation**.
   c. When IP generation is complete, click **Analysis & Synthesis**.

   No further compilation is needed for the alternate persona synthesis step. Running further compilation steps on this type of revision is prohibited by the tools.

Related Information

Creating the Partial Reconfiguration Design on page 12
Adding components to the PR design: bridge, memory, and system ID

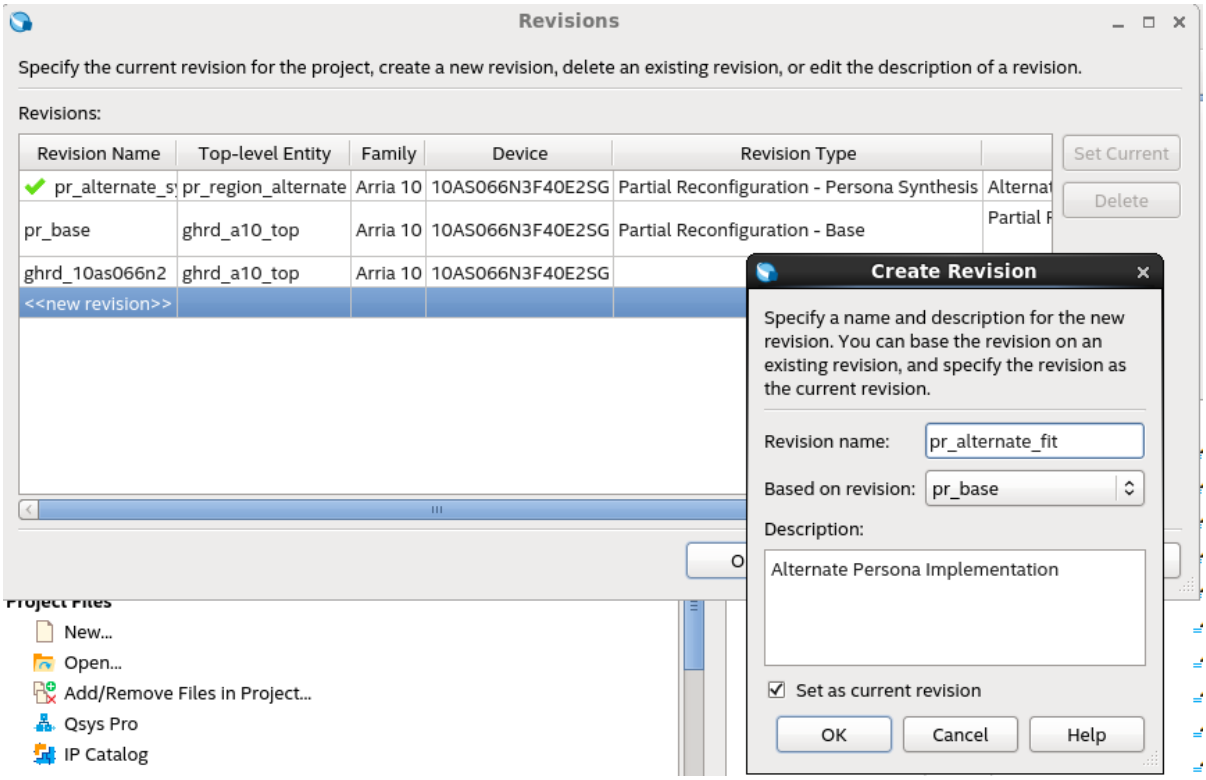
Implementing the Alternate Persona

In the following steps, you export the implemented base region and synthesized alternate persona regions as snapshots, and merge them into a complete PR design revision.

The snapshots are merged in the alternate persona implementation revision. The static snapshot is the scaffolding: the placed and routed static region, with an unused area set aside for the PR region. The persona is fitted and routed into that region. The persona snapshot is an unplaced and unrouted design, which is subsequently placed and routed into the PR region.

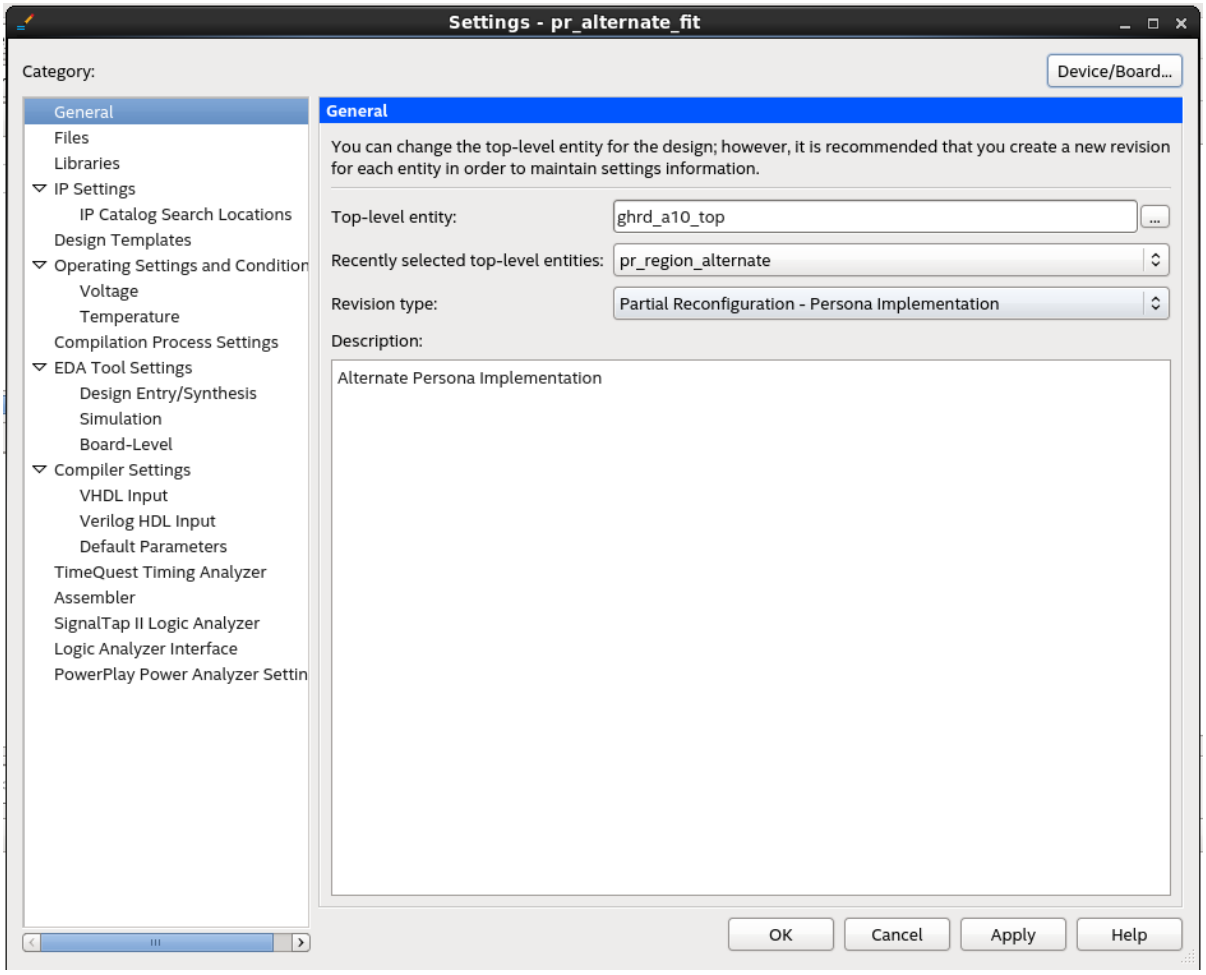
1. In the main Quartus window, point to **Project**, and click **Revisions** from the drop down menu. Create a new revision for the PR alternate persona implementation configuration. Double click on **<< new revision >>** and enter the revision information as shown below. Note this revision is based on the existing pr_base revision.

Figure 29: Creating an Alternate Persona Revision



2. Click **OK** to create the revision and open it as the current revision.
3. To change the revision type of the new revision, point to **Assignments**, click **Settings**, choose the **General** category, and change the **Revision Type** field to **Partial Reconfiguration – Persona Implementation** as shown below.

Figure 30: Setting the Alternate Persona Revision Type



Because this is a Partial Reconfiguration – Persona Implementation type revision, the Quartus Pro tools do not allow the compilation stages before the fitter to run. Furthermore, some of the required compilation steps for this type of revision are not available in the Quartus Pro GUI window. You must compile this revision type from the command line.

4. To compile the persona implementation, type the following commands:

```
$ quartus_cdb ghrd_10as066n2 -c pr_base --export_pr_static_block \
root_partition --snapshot final --file pr_base_static.qdb
```

The step above exports a snapshot (`pr_base_static.qdb`) of the PR static region design from the base revision. This design has been compiled, placed, and routed, and is used as a scaffolding to build the alternate personas.

```
$ quartus_cdb ghrd_10as066n2 -c pr_alternate_synth --export_block \
root_partition --snapshot synthesized --file pr_alternate_synth.qdb
```

The step above exports a snapshot (`pr_alternate_synth.qdb`) of the synthesized logic for the PR region alternate persona. This design is placed and routed using the scaffolding generated above.

```
$ quartus_cdb ghrd_10as066n2 -c pr_alternate_fit --import_block \
root_partition --file pr_base_static.qdb
$ quartus_cdb ghrd_10as066n2 -c pr_alternate_fit --import_block \
pr_region --file pr_alternate_synth.qdb
```

The steps above import the snapshots of the static place and route design, plus the synthesized alternate personal logic, into the current revision.

```
$ quartus_fit ghrd_10as066n2 -c pr_alternate_fit
$ quartus_sta ghrd_10as066n2 -c pr_alternate_fit
$ quartus_asm ghrd_10as066n2 -c pr_alternate_fit
```

The last steps complete the normal compilation flow starting with the fitter.
After successful completion of the assembler stage, new files with the prefix `pr_alternate_fit` can be found in the `output_files/` directory. These files contain reports and FPGA configuration images for the alternate persona.

Related Information

Important Partial Reconfiguration Terminology on page 42

Generating the RBF FPGA Image Files

Convert the FPGA configuration images to RBF files using the following commands. The host software developer needs RBF files to configure the FPGA.

```
$ quartus_cpf --hps -c output_files/pr_base.sof output_files/pr_base.rbf
$ quartus_cpf -c output_files/pr_base.pr_region.pmsf \
output_files/pr_region_default.rbf
$ quartus_cpf -c output_files/pr_alternate_fit.pr_region.pmsf \
output_files/pr_region_alt.rbf
```

Design Handoff to Software Developer

The FPGA designer must provide following information to the host software developer.

- The RBF images for the complete default FPGA and each PR region persona. In this example, three RBF files must be provided:
  - `pr_base.rbf`
  - `pr_region_default.rbf`
  - `pr_region_alt.rbf`
- The base address of the freeze controller logic
- The base address of the freeze bridge address window

Related Information

- **Generating the RBF FPGA Image Files** on page 35
  Generating the RBF files

- **Adding the PR Region** on page 18
  Determining the freeze controller and freeze bridge base addresses

Generating the Example Software Image

Overview of Software Image Creation

The SoC FPGA GSRD provided on **rocketboards.org** is not initially set up for PR. The Linux kernel is compiled with the required configuration options for PR, and the preferred command line tool for managing device tree overlays is provided, but the device tree loaded at boot (the "live tree") is not laid out properly for PR.

The Linux kernel build configuration options required for PR are shown in the following list. These options must be set in the `socfpga_defconfig` file when rebuilding the Linux kernel as described on **rocketboards.org**. The Linux kernel provided with the GSRD was built with these options enabled.

```
CONFIG_FPGA=y
CONFIG_FPGA_REGION=y
CONFIG_FPGA_MGR_SOCFPGA=y
CONFIG_FPGA_MGR_SOCFPGA_A10=y
CONFIG_FPGA_BRIDGE=y
CONFIG_SOCFPGA_FPGA_BRIDGE=y
CONFIG_ALTERA_FREEZE_BRIGE=y
CONFIG_OF=y
CONFIG_OF_OVERLAY=y
CONFIG_OF_CONFIGFS=y
CONFIG_CONFIGFS_FS=y
```

The command line tool used to manage device tree overlays is called **dtbt**, an open source Python script. The source code is provided at **altera-opensource/dtbt** on GitHub. The software tool is provided in the `/sbin/` directory of the GSRD SD card image. Use `dtbt -h` for more information.

The default device tree provided with the GSRD is not set up for PR. To support PR, the device tree loaded at boot must contain a definition of the FPGA regions linking the HPS FPGA manager and the bridge controllers to the top level FPGA region. The default tree provided on the GSRD SD card image from **rocketboards.org** does not provide this linkage.

After a proper device tree is loaded at Linux boot time, the **dtbt** tool is used to load and unload device tree overlays to modify the live tree.

The instructions provided in this document describe how to build a minimal boot device tree to support PR. Instructions are also provided on how to build the required device tree overlays, where to place them within the Linux directory structure on the SD card, and how to load and unload them using the **dtbt** command line tool.

To create and run the software image, you carry out these steps:

1. Rebuild the Linux kernel with build configuration options for PR
2. Create a Linux device tree describing the FPGA manager and the bridge controllers in relation to the top-level configuration region
3. Create device tree overlays for personas in each PR region
4. Place the device trees in the correct location in the Linux directory structure
5. Boot the Linux system, loading the default device tree and the base revision overlay
6. Use the `dtbt` tool to load and unload device tree overlays for various personas

Related Information

altera-opensource/dtbt on GitHub

Generating a New Boot Device Tree

The device tree loaded at boot by the GSRD default SD card image does not contain the required information to support PR. However, the **Linux development repository for socfpga** on the Altera Opensource GitHub provides Linux kernel source code with a device tree configured to support PR.

Related Information

Linux development repository for socfpga on GitHub

Critical Information in the Default Device Tree

Listed below is the critical information provided in the default device tree, which is located with the Linux kernel source. `/soc/base_fpga_region` defines an empty FPGA region that is filled in by loading a device tree overlay after boot. This entry links to the FPGA manager, which is used to configure this FPGA region. In the example in this document, you add a PR FPGA region as a child node of `/soc/base_fpga_region`. This region inherits the link to the FPGA manager created in this boot device tree.

socfpga-linux/arch/arm/boot/dts/socfpga_arria10.dtsi

```
/ {
    #address-cells = <1>;
    #size-cells = <1>;

        soc {
        #address-cells = <1>;
        #size-cells = <1>;
        compatible = "simple-bus";
        device_type = "soc";
        interrupt-parent = <&intc>;
        ranges;

        base_fpga_region {
            compatible = "fpga-region";
            fpga-mgr = <&fpga_mgr>;

            #address-cells = <0x1>;
            #size-cells = <0x1>;

            /*
             * NOTE: This node will be the parent for
             *       the PR FPGA Region that is managed
             *       using device tree overlays. The child
             *       FPGA Regions will inherit the link to
             *       the FPGA Manger created here.
             */
```

```
        };

        fpga_mgr: fpga-mgr@ffd03000 {
            compatible = "altr,socfpga-a10-fpga-mgr";
            reg = <0xffd03000 0x100
                   0xffcfe400 0x20>;
            clocks = <&l4_mp_clk>;
            resets = <&rst FPGAMGR_RESET>;
            reset-names = "fpgamgr";
        };

        };
};
```

Steps to Create a New Device Tree

In this example, to create a new device tree, you clone the source code repository and make the default device tree for the SoC FPGA with SD card boot. To make the device tree, type the following commands.

1. `git clone https://github.com/altera-opensource/socfpga-linux`
2. `cd socfpga-linux`
3. `git checkout -b socfpga-4.1.33-ltsi origin/socfpga-4.1.33-ltsi`[(2)]
4. `~/intelFPGA/16.1/embedded/embedded_command_shell.sh`
5. `export ARCH=arm`
6. `export CROSS_COMPILE=arm-altera-eabi-`
7. `make socfpga_defconfig`
8. `make socfpga_arria10_socdk_sdmmc.dtb`
9. `cp arch/arm/boot/dts/socfpga_arria10_socdk_sdmmc.dtb ./`

Creating the Device Tree Overlays

Three device tree overlay files are needed for this example. The first overlay updates the system description to add the new node containing the reconfigurable region. The other two overlays are used to describe the two PR region personas.

1. Create a new file named `base_static.dtso` with the following contents. Note that the freeze Ccontroller type and base address must be entered into the appropriate fields.

```
/dts-v1//plugin/;

/ {

  fragment@0 {
    target-path = "/soc/base_fpga_region";
    #address-cells = <0x1>;
    #size-cells = <0x1>;

    __overlay__ {
      ranges = <0x0 0x0 0xc0000000 0x40000 0x1 0x0 0xff200000 0x1000>;
      external-fpga-config;
      #address-cells = <0x2>;
      #size-cells = <0x1>;

      fpga_pr_region0 {
        compatible = "fpga-region";
        fpga-bridges = <&freeze_controller_0>;
```

(2) This step ensures that you are using the correct kernel version.

```
        ranges;
      };

      freeze_controller_0: freeze_controller@0x100000450 {
        compatible = "altr,freeze_controller-16.1", \
          "altr,freeze-bridge-controller";
        reg = <0x1 0x450 0x10>;
      };
    };
  };
};
```

2. Create a file named `pr_region_default.dtso` with the following contents. Note that the system ID peripheral base address is required.

```
/dts-v1//plugin/;

/ {

  fragment@1 {
    target-path = "/soc/base_fpga_region/fpga_pr_region0";
    #address-cells = <0x1>;
    #size-cells = <0x1>;

    __overlay__ {
      partial-fpga-config;
      firmware-name = "pr_region_default.rbf";
      #address-cells = <0x2>;
      #size-cells = <0x1>;
      region-unfreeze-timeout-us = <0x4>;
      region-freeze-timeout-us = <0x4>;

      sysid@0x100000800 {
        compatible = "altr,sysid-16.1", "altr,sysid-1.0";
        reg = <0x1 0x800 0x8>;
      };
    };
  };
};
```

3. Create a file named `pr_region_alt.dtso` with the following contents. Note that the system ID peripheral base address is different from the default persona.

```
/dts-v1//plugin/;

/ {

  fragment@1 {
    target-path = "/soc/base_fpga_region/fpga_pr_region0";
    #address-cells = <0x1>;
    #size-cells = <0x1>;

    __overlay__ {
      partial-fpga-config;
      firmware-name = "pr_region_alt.rbf";
      #address-cells = <0x2>;
      #size-cells = <0x1>;
      region-unfreeze-timeout-us = <0x4>;
      region-freeze-timeout-us = <0x4>;

      sysid@0x100000A00 {
        compatible = "altr,sysid-16.1", "altr,sysid-1.0";
```

```
            reg = <0x1 0xA00 0x8>;
          };
        };
      };
    };
```

4. Load the `embedded_command_shell.sh` environment provided by the SoC EDS tools. This environment provides the device tree compiler command `dtc`. Use `dtc` to compile the newly created Device Tree Source (DTS) files to Device Tree Blob (DTB) files as shown below.

```
$ dtc -@ -I dts -O dtb -o base_static.dtbo base_static.dtso
$ dtc -@ -I dts -O dtb -o pr_region_default.dtbo pr_region_default.dtso
$ dtc -@ -I dts -O dtb -o pr_region_alt.dtbo pr_region_alt.dtso
```

Loading Partial Reconfiguration Designs Using Linux

To perform partial reconfiguration, software must carry out the following steps:

1. For components in the region to be reprogrammed, shut down all driver instances.
2. Use the PR controller to freeze the interfaces to the PR region.
3. Use the FPGA manager to load the persona bit stream.
4. Release bridges surrounding PR region from reset.
5. Instantiate drivers for components inside the PR region.

This example shows how to modify a GSRD image for an SD card. The GSRD image, including the HPS bootloader and Linux, loads the new default complete FPGA configuration image and supports loading the two PR region personas created in earlier steps.

When adding overlays to the GSRD, you must also update the base device tree, used by the Linux kernel to boot. The base device tree is in the kernel source.

Related Information

- **Building the Base Revision with the Reconfigurable Design Partition** on page 27
  Creating the default PR region persona
- **Implementing the Alternate Persona** on page 33
  Creating the alternate PR region persona

Installing Files on the SD Card

The following steps show how to install your software on the SD card (created in "Hardware Prerequisites").

1. Copy the `pr_base.core.rbf` and `pr_base.periph.rbf` files to the FAT partition of the SD card, overwriting the `ghrd_10as066n2.core.rbf` and `ghrd_10as066n2.periph.rbf` files that already exist at the directory root. This replaces the GHRD image provided by the GSRD SD card image with the default complete FPGA configuration image, which you created in the preceding steps.
2. Copy the `socfpga_arria10_socdk_sdmmc.dtb` file to the FAT partition of the SD card, overwriting the `socfpga_arria10_socdk_sdmmc.dtb` file that already exists at the directory root.

This replaces the GSRD boot device tree provided by the GSRD SD card image with the default PR enabled device tree from the Linux kernel source tree, which you created in the preceding steps.

3. Copy the `pr_region_default.rbf` and `pr_region_alt.rbf` files to the `/lib/firmware/` directory on the EXT3 partition of the SD card. When loading new firmware images using the device tree overlay, Linux searches this directory to find any named firmware files.
4. Copy the `base_static.dtbo`, `pr_region_default.dtbo` and `pr_region_alt.dtbo` files to the `/boot/` directory on the EXT3 partition of the SD card. This is the recommended location for PR DTB files.

Related Information

Hardware Prerequisites on page 3

Loading and Unloading the Device Tree Overlays

1. Boot the SoC Development Kit using the updated SD card.
2. From the `/boot/` directory, run the following commands to load the base static region device tree overlay. This creates the device tree node that can be modified to load and unload the example PR region personas.

```
root@arria10:~# dtbt -a base_static.dtbo -p /boot
Set dtbo search path to /boot
Applying dtbo: base_static.dtbo
```

3. Check the list of applied overlays using the following command.

```
root@arria10:~# dtbt -l
1 fpga_static_region.dtbo applied /sys/kernel/config/device-tree/overlays/1-
base_static.dtbo
```

4. Load the default PR region persona using the following command.

```
root@arria10:~# dtbt -a pr_region_default.dtbo -p /boot
Set dtbo search path to /boot
Applying dtbo: pr_region_default.dtbo
[ 78.934742] fpga_manager fpga0: writing pr_region_default.rbf to SoCFPGA
Arria10 FPGA Manager
```

5. Check the system ID value using the following command.

```
root@arria10:~# cat /sys/bus/platform/drivers/altera_sysid/ \
    ff200800.sysid/sysid/id
```

6. Unload the default PR region persona and load the alternate persona using the following commands.

```
root@arria10:~# dtbt -r pr_region_default.dtbo -p /boot
Set dtbo search path to /boot
Removing dtbo: 2-pr_region_default.dtbo

root@arria10:~# dtbt -a pr_region_alt.dtbo -p /boot
Set dtbo search path to /boot
Applying dtbo: pr_region_alt.dtbo
```

```
[ 437.905424] fpga_manager fpga0: writing pr_region_alt.rbf to SoCFPGA Arria10
FPGA Manager
```

7. Check the system ID value of the newly loaded alternate persona with the following command.

```
root@arria10:~# cat \
/sys/bus/platform/drivers/altera_sysid/ff200a00.sysid/sysid/id
```

8. Finally, remove the PR region alternate persona using the following command.

```
root@arria10:~# dtbt -r pr_region_alt.dtbo -p /boot
Set dtbo search path to /boot
Removing dtbo: 2-pr_region_alt.dtbo
```

Related Information

altera-opensource/dtbt on GitHub

Important Partial Reconfiguration Terminology

Implementing a partial reconfiguration (PR) design requires understanding of the FPGA device capabilities and the Quartus Prime Pro compilation flow. This section defines common terminology that you need to fully understand PR.

Core logic: Logic resources on the device which have no direct off-chip connections, such as LABs, embedded memory blocks, and DSP blocks.

Floorplan: The layout of physical resources on the device. Creating a design floorplan, or floorplanning, is the process of mapping logical design hierarchy to physical regions in the device.

Periphery logic: Logic resources on the device which include offchip I/O connections, such as transceivers, external memory interfaces, GPIOs, I/O receivers, and the HPS.

PR control block: A dedicated FPGA block. The PR control block processes the PR requests, handshake protocols, and verifies the cyclic redundancy check (CRC).

PR host: The system for coordinating PR. The PR host communicates with the PR control block. Implement the PR host within the FPGA (internal PR host) or in a chip or microprocessor (external PR host).

PR IP Core: The Intel partial reconfiguration IP core that you instantiate in the static region of your design. This IP core interfaces with the PR control block to manage the bitstream source.

PR partition or reconfigurable design partition: A logical division in the source code hierarchy that you designate for PR. A PR project can contain one or more partially reconfigurable PR partitions.

PR persona: A specific configuration of a PR partition in a PR region. A PR region can support multiple personas. Static regions support only one persona.

PR region or LogicLock Plus region: A physical portion of the device that you choose to reconfigure. You can configure a device with more than one PR region. A PR region can only include core components such as LAB, RAM, or DSP. You associate a PR region with a PR partition in your design.

Revision: A collection of settings and constraints for one version of your project. A Quartus Prime Settings File (.qsf) preserves each revision of your project. Your Quartus Prime project can contain several

revisions. A PR project includes a base revision, defining region boundaries and base logic, and one or more revisions defining different PR region implementations.

Snapshot: The output of a Compiler stage. The Quartus Prime Pro Edition Compiler generates a snapshot of the compiled database after each stage. The snapshot preserves the compilation database.

Static region: All areas outside the PR regions in your project. You associate the static region with the top-level partition of the design. The static region can contain both core and periphery locations in the device.

Revision History

Date	Version	Changes
January 2017	2017.01.25	Initial release