AN 797: Partially Reconfiguring a Design on Intel® Arria® 10 GX FPGA Development Board

Updated for Intel® Quartus® Prime Design Suite: **18.0**

Subscribe
Send Feedback

AN-797 | 2018.05.07
Latest document on the web: PDF | HTML

Contents

Partially Reconfiguring a Design on Intel® Arria® 10 GX FPGA Development Board

This application note demonstrates transforming a simple design into a partially reconfigurable design, and implementing the design on the Intel® Arria® 10 GX FPGA development board.

Partial reconfiguration (PR) feature allows you to reconfigure a portion of the FPGA dynamically, while the remaining FPGA design continues to function. Create multiple personas for a particular region in your design, without impacting operation in areas outside this region. This methodology is effective in systems where multiple functions time-share the same FPGA device resources. The current version of the software introduces a new and simplified compilation flow for partial reconfiguration.

Partial reconfiguration provides the following advancements to a flat design:

- Allows run-time design reconfiguration
- Increases scalability of the design
- Reduces system down-time
- Supports dynamic time-multiplexing functions in the design
- Lowers cost and power consumption through efficient use of board space

Implementation of this reference design requires basic familiarity with the Intel Quartus® Prime FPGA implementation flow and knowledge of the primary Intel Quartus Prime project files. This tutorial uses the Intel Arria 10 GX FPGA development board on the bench, outside of the PCIe* slot in your workstation.

Related Information

- Intel Arria 10 FPGA Development Kit User Guide
- Partial Reconfiguration Concepts
- Partial Reconfiguration Design Flow
- Partial Reconfiguration Design Considerations
- Partial Reconfiguration Design Guidelines

Reference Design Requirements

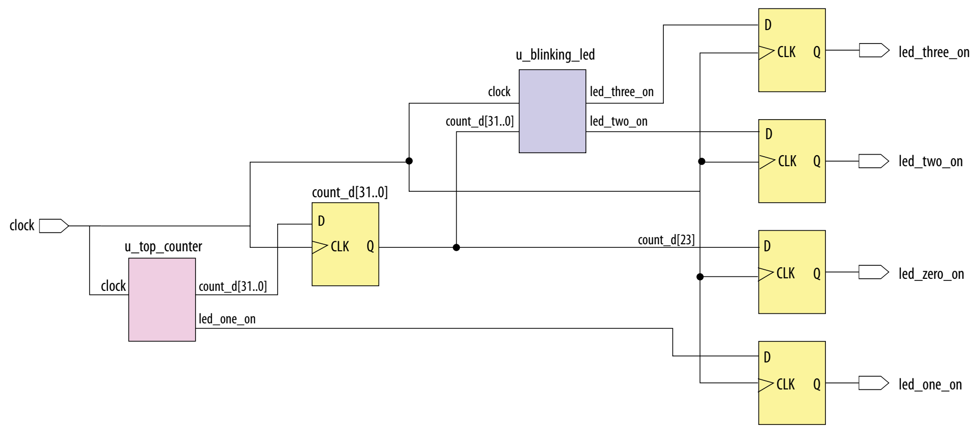
This reference design requires the following:

- Installation and basic familiarity with the Intel Quartus Prime Pro Edition version 18.0 design flow and project files for the design implementation.
- Connection with the Intel Arria 10 GX FPGA development board on the bench.

Reference Design Overview

This reference design consists of one 32-bit counter. At the board level, the design connects the clock to a 50MHz source, and connects the output to four LEDs on the FPGA. Selecting the output from the counter bits in a specific sequence causes the LEDs to blink at a specific frequency.

Figure 1. Flat Reference Design without PR Partitioning



Reference Design Files

The partial reconfiguration tutorial is available in the following location:

https://github.com/intel/fpga-partial-reconfig

To download the tutorial:

1. Click **Clone or download**.
2. Click **Download ZIP**. Unzip the `fpga-partial-reconfig-master.zip` file.
3. Navigate to the `tutorials/a10_pcie_devkit_blinking_led` sub-folder to access the reference design.

The `flat` folder consists of the following files:

Table 1. Reference Design Files

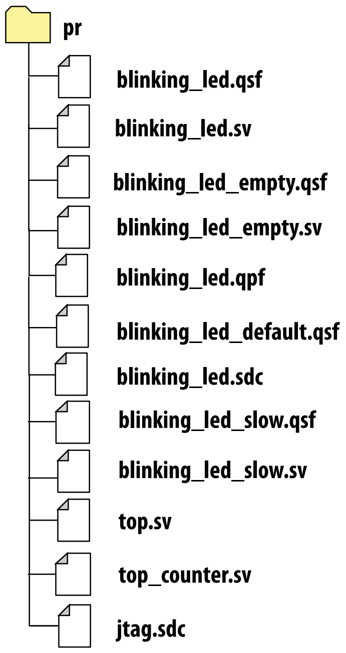
File Name	Description
`top.sv`	Top-level file containing the flat implementation of the design. This module instantiates the `blinking_led` sub-partition and the `top_counter` module.
`top_counter.sv`	Top-level 32-bit counter that controls `LED[1]` directly. The registered output of the counter controls `LED[0]`, and also powers `LED[2]` and `LED[3]` via the `blinking_led` module.
`blinking_led.sdc`	Defines the timing constraints for the project.

continued...

File Name	Description
`blinking_led.sv`	This module acts as the PR partition. The module receives the registered output of `top_counter` module, which controls `LED[2]` and `LED[3]`.
`blinking_led.qpf`	Intel Quartus Prime project file containing the list of all the revisions in the project.
`blinking_led.qsf`	Intel Quartus Prime settings file containing the assignments and settings for the project.

Note: The `pr` folder contains the complete set of files you create using this application note. Reference these files at any point during the walkthrough.

Figure 2. Reference Design Files



Reference Design Walkthrough

The following steps describe the application of partial reconfiguration to a flat design. The tutorial uses the Intel Quartus Prime Pro Edition software for the Intel Arria 10 GX FPGA development board:

- Step 1: Getting Started on page 6
- Step 2: Creating a Design Partition on page 6
- Step 3: Allocating Placement and Routing Region for a PR Partition on page 7
- Step 4: Adding the Intel Arria 10 Partial Reconfiguration Controller IP Core on page 9
- Step 5: Defining Personas on page 11
- Step 6: Creating Revisions on page 14

- Step 7: Compiling the Base Revision and Exporting the Static Region on page 15
- Step 8: Preparing PR Implementation Revisions on page 17
- Step 9: Programming the Board on page 19

Step 1: Getting Started

To copy the reference design files to your working environment and compile the `blinking_led` flat design:

1. Create a directory in your working environment, `a10_pcie_devkit_blinking_led_pr`.
2. Copy the downloaded `tutorials/a10_pcie_devkit_blinking_led/flat` sub-folder to the directory, `a10_pcie_devkit_blinking_led_pr`.
3. In the Intel Quartus Prime Pro Edition software, click **File ➤ Open Project** and select `blinking_led.qpf`.
4. To compile the flat design, click **Processing ➤ Start Compilation**.

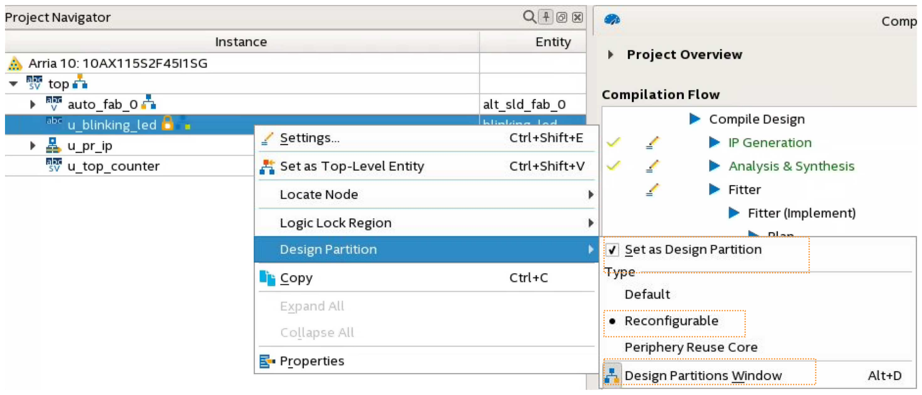
Step 2: Creating a Design Partition

You must create design partitions for each PR region that you want to partially reconfigure. You can create any number of independent partitions or PR regions in your design. This tutorial creates a design partition for the `u_blinking_led` instance.

To create design partition for partial reconfiguration:

1. Right-click the `u_blinking_led` instance in the **Project Navigator** and click **Design Partition ➤ Set as Design Partition**. A design partition icon appears next to each instance that is set as a partition.

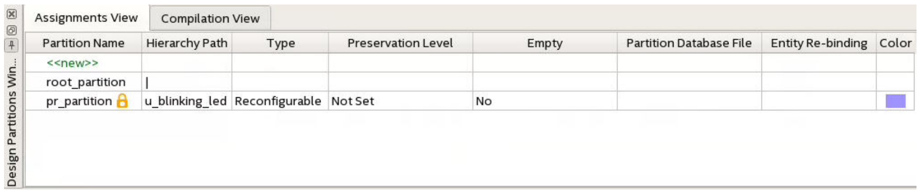
Figure 3. Creating Design Partitions from Project Navigator



2. To define the partition **Type**, right-click the `u_blinking_led` instance in the **Hierarchy** tab, click **Design Partition ➤ Reconfigurable**. You can only define the partition **Type** after setting the instance as a partition.

   The design partition appears on the **Assignments View** tab of the Design Partitions Window.

Figure 4. Design Partitions Window



Partition Name	Hierarchy Path	Type	Preservation Level	Empty	Partition Database File	Entity Re-binding	Color
<<new>>							
root_partition	|						
pr_partition	u_blinking_led	Reconfigurable	Not Set	No			🟪

3. Edit the partition name in the Design Partitions Window by double-clicking the name. For this reference design, rename the partition name to `pr_partition`.

   Note: When you create a partition, the Intel Quartus Prime software automatically generates a partition name, based on the instance name and hierarchy path. This default partition name can vary with each instance.

Verify that the `blinking_led.qsf` contains the following assignments, corresponding to your reconfigurable design partition:

```
set_instance_assignment -name PARTITION pr_partition -to u_blinking_led
set_instance_assignment -name PARTIAL_RECONFIGURATION_PARTITION ON \
        -to u_blinking_led
```

Related Information

Create Design Partitions for Partial Reconfiguration

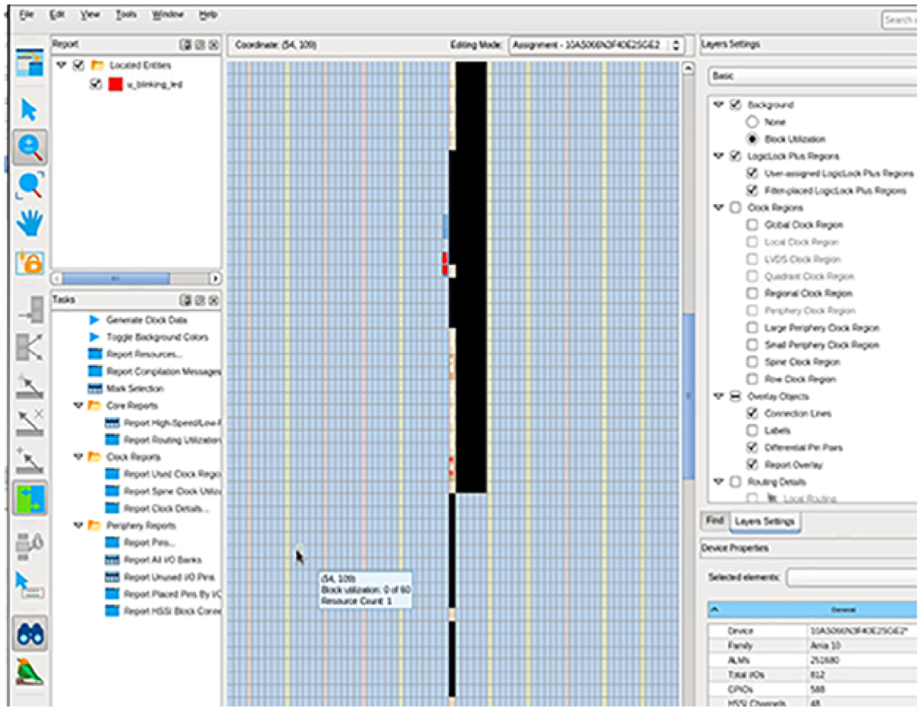
Step 3: Allocating Placement and Routing Region for a PR Partition

For every base revision you create, the PR design flow uses your PR partition region allocation to place the corresponding persona core in the reserved region. To locate and assign the PR region in the device floorplan for your base revision:

1. Right-click the `u_blinking_led` instance in the **Project Navigator** and click **Logic Lock Region ➤ Create New Logic Lock Region**. The region appears on the Logic Lock Regions Window.
2. Your placement region must enclose the `blinking_led` logic. Select the placement region by locating the node in Chip Planner. Right-click the `u_blinking_led` region name in the Logic Lock Regions Window and click **Locate Node ➤ Locate in Chip Planner**.

   The `u_blinking_led` region is color-coded.

Figure 5. Chip Planner Node Location for `blinking_led`



Device	10AS066N3F40E2SGE2*
Family	Arria 10
ALMs	251680
Total I/Os	812
GPIOs	588
HSSI Channels	48

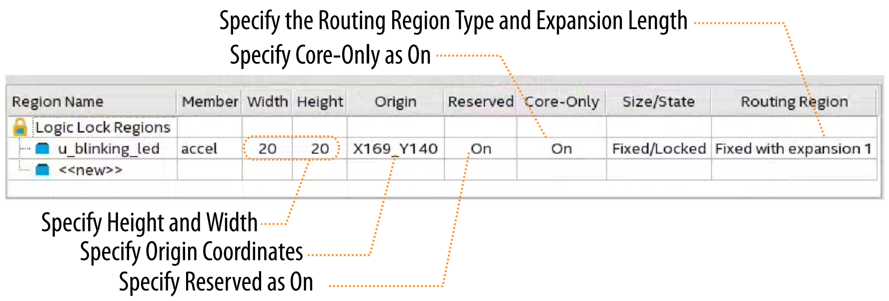
3. In the Logic Lock Regions window, specify the placement region co-ordinates in the **Origin** column. The origin corresponds to the lower-left corner of the region. For example, to set a placement region with (`X1 Y1`) co-ordinates as (`69 10`), specify the **Origin** as `X69_Y10`. The Intel Quartus Prime software automatically calculates the (`X2 Y2`) co-ordinates (top-right) for the placement region, based on the height and width you specify.

   Note: This tutorial uses the (`X1 Y1`) co-ordinates - (`69 10`), and a height and width of 20 for the placement region. Define any value for the placement region, as long as the region covers the `blinking_led` logic.

4. Enable the **Reserved** and **Core-Only** options.
5. Double-click the **Routing Region** option. The **Logic Lock Routing Region Settings** dialog box appears.
6. Select **Fixed with expansion** for the **Routing type**. Selecting this option automatically assigns an expansion length of `1`.

   Note: The routing region must be larger than the placement region, to provide extra flexibility for the Fitter when the engine routes different personas.

Figure 6. Logic Lock Regions Window


Specify the Routing Region Type and Expansion Length
Specify Core-Only as On
Specify Height and Width
Specify Origin Coordinates
Specify Reserved as On

Region Name	Member	Width	Height	Origin	Reserved	Core-Only	Size/State	Routing Region
Logic Lock Regions								
u_blinking_led	accel	20	20	X169_Y140	On	On	Fixed/Locked	Fixed with expansion 1
<<new>>								

Verify that the `blinking_led.qsf` contains the following assignments, corresponding to your floorplanning:

```
set_instance_assignment -name PLACE_REGION "69 10 88 29" -to u_blinking_led
set_instance_assignment -name RESERVE_PLACE_REGION ON -to u_blinking_led
set_instance_assignment -name CORE_ONLY_PLACE_REGION ON -to u_blinking_led
set_instance_assignment -name ROUTE_REGION "68 9 89 30" -to u_blinking_led
```

Related Information

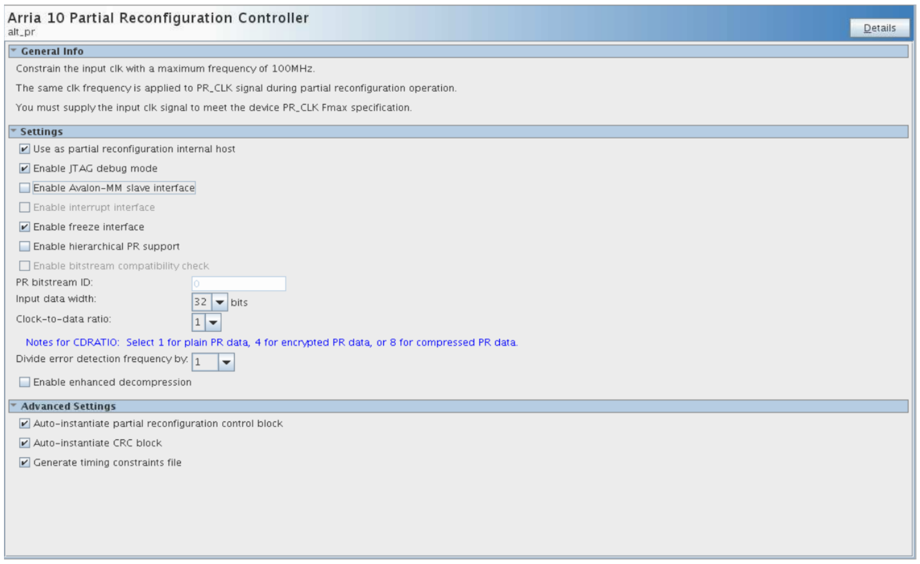
- Floorplan the Partial Reconfiguration Design
- Applying Floorplan Constraints Incrementally

Step 4: Adding the Intel Arria 10 Partial Reconfiguration Controller IP Core

The Intel Arria 10 Partial Reconfiguration Controller IP core enables reconfiguration of the PR partition. This IP core uses JTAG to reconfigure the PR partition. To add the Intel Arria 10 Partial Reconfiguration Controller IP core to your Intel Quartus Prime project:

1. Type `Partial Reconfiguration` in the IP Catalog (**Tools ➤ IP Catalog**).
2. Double-click the Intel Arria 10 Partial Reconfiguration Controller IP core.
3. In the **Create IP Variant** dialog box, type `pr_ip` as the file name, and then click **Create**. Use the default parameterization for `pr_ip`. Ensure that the **Enable JTAG debug mode** and **Enable freeze interface** options are turned on, and **Enable Avalon-MM slave interface** option is turned off.

Figure 7. Intel Arria 10 Partial Reconfiguration Controller IP Core Parameters



Arria 10 Partial Reconfiguration Controller
alt_pr

General Info

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

The same clk frequency is applied to PR_CLK signal during partial reconfiguration operation.

You must supply the input clk signal to meet the device PR_CLK Fmax specification.

Settings

- ✓ Use as partial reconfiguration internal host
- ✓ Enable JTAG debug mode
- ☐ Enable Avalon-MM slave interface
- ☐ Enable interrupt interface
- ✓ Enable freeze interface
- ☐ Enable hierarchical PR support
- ☐ Enable bitstream compatibility check

PR bitstream ID: 0
Input data width: 32 bits
Clock-to-data ratio: 1

Notes for CDRATIO: Select 1 for plain PR data, 4 for encrypted PR data, or 8 for compressed PR data.

Divide error detection frequency by: 1

- ☐ Enable enhanced decompression

Advanced Settings

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

4. Click **Finish**, and exit the parameter editor without generating the system. The parameter editor generates the `pr_ip.ip` IP variation file and adds the file to the `blinking_led` project.

   Note: a. If you are copying the `pr_ip.ip` file from the `pr` folder, manually edit the `blinking_led.qsf` file to include the following line:

   ```
   set_global_assignment -name IP_FILE pr_ip.ip
   ```

   b. Place the `IP_FILE` assignment after the `SDC_FILE` assignments (`jtag.sdc` and `blinking_led.sdc`) in your `blinking_led.qsf` file. This ordering ensures appropriate constraining of the Partial Reconfiguration Controller IP core.

   Note: To detect the clocks, the `.sdc` file for the PR IP must follow any `.sdc` that creates the clocks that the IP core uses. You facilitate this order by ensuring the `.ip` file for the PR IP core comes after any `.ip` files or `.sdc` files that you use to create these clocks in the `.qsf` file for your Intel Quartus Prime project revision. For more information, refer to the *Partial Reconfiguration IP Solutions User Guide*.

Related Information

Partial Reconfiguration IP Solutions User Guide
For information on all Partial Reconfiguration IP cores.

Updating the Top-Level Design

To update the `top.sv` file with the PR_IP instance:

1. To add the `pr_ip` instance to the top-level design, uncomment the following code block in `top.sv` file:

```
pr_ip u_pr_ip
    (
        .clk           (clock),
        .nreset        (1'b1),
        .freeze        (freeze),
        .pr_start      (1'b0),            // ignored for JTAG
        .status        (pr_ip_status),
        .data          (16'b0),
        .data_valid    (1'b0),
        .data_ready    ()
    );
```

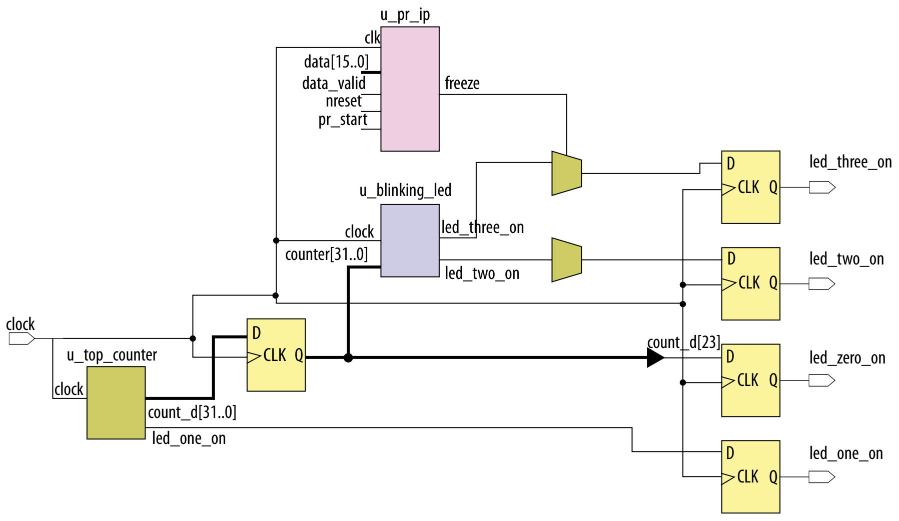
2. To force the output ports to logic 1 during reconfiguration, use the freeze control signal output from PR_IP. Uncomment the following lines of code:

```
assign led_two_on_w    = freeze ? 1'b1 : pr_led_two_on;
assign led_three_on_w  = freeze ? 1'b1 : pr_led_three_on;
```

3. To assign an instance of the default persona (`blinking_led`), update the `top.sv` file with the following block of code:

```
blinking_led u_blinking_led
    (
        .led_two_on    (pr_led_two_on),
        .led_three_on (pr_led_three_on),
        .clock    (clock),
        .counter (count_d)
    );
```

Figure 8. Partial Reconfiguration IP Core Integration



Step 5: Defining Personas

This reference design defines three separate personas for the single PR partition. To define and include the personas in your project:

1. Create three SystemVerilog files, `blinking_led.sv`, `blinking_led_slow.sv`, and `blinking_led_empty.sv` in your working directory for the three personas.

Note:
- `blinking_led.sv` is already available as part of the files you copy from the `flat/` sub-directory. You can simply reuse this file.
- If you create the SystemVerilog files from the Intel Quartus Prime Text Editor, disable the **Add file to current project** option, when saving the files.

Table 2. Reference Design Personas

File Name	Description	Code

`blinking_led.sv`	Default persona with same design as the flat implementation	`` `timescale 1 ps / 1 ps``
		`` `default_nettype none``
		
		`module blinking_led (`
		    `// clock`
		    `input wire clock,`
		    `input wire [31:0] counter,`
		
		    `// Control signals for the LEDs`
		    `output wire led_two_on,`
		    `output wire led_three_on`
		`);`
		
		    `localparam COUNTER_TAP = 23;`
		
		    `reg led_two_on_r;`
		    `reg led_three_on_r;`
		
		    `assign led_two_on = led_two_on_r;`
		    `assign led_three_on = led_three_on_r;`
		
		    `always_ff @(posedge clock)`
		    `begin`
		        `led_two_on_r <= counter[COUNTER_TAP];`
		        `led_three_on_r <= counter[COUNTER_TAP];`
		    `end`
		
		`endmodule`

`blinking_led_slow.sv`	LEDs blink slower	`` `timescale 1 ps / 1 ps``
		`` `default_nettype none``
		
		`module blinking_led_slow (`
		    `// clock`
		    `input wire clock,`
		    `input wire [31:0] counter,`
		
		    `// Control signals for the LEDs`
		    `output wire led_two_on,`
		    `output wire led_three_on`
		`);`
		
		    `localparam COUNTER_TAP = 27;`
		
		    `reg led_two_on_r;`
		    `reg led_three_on_r;`
		
		    `assign led_two_on = led_two_on_r;`
		    `assign led_three_on = led_three_on_r;`
		
		    `always_ff @(posedge clock)`
		    `begin`
		       `led_two_on_r <= counter[COUNTER_TAP];`
		       `led_three_on_r <= counter[COUNTER_TAP];`
		    `end`
		
		`endmodule`

`blinking_led_empty.sv`	LEDs stay ON	`` `timescale 1 ps / 1 ps``
		`` `default_nettype none``
		
		`module blinking_led_empty(`
		    `// clock`
		    `input wire clock,`
		    `input wire [31:0] counter,`
		
		    `// Control signals for the LEDs`
		    `output wire led_two_on,`
		    `output wire led_three_on`
		`);`
		
		    `// LED is active low`
		    `assign led_two_on = 1'b0;`
		    `assign led_three_on = 1'b0;`
		
		`endmodule`

Related Information

Step 2: Creating a Design Partition on page 6

Step 6: Creating Revisions

The PR design flow uses the project revisions feature in the Intel Quartus Prime software. Your initial design is the base revision, where you define the static region boundaries and reconfigurable regions on the FPGA.

From the base revision, you create multiple revisions. These revisions contain the different implementations for the PR regions. However, all PR implementation revisions use the same top-level placement and routing results from the base revision.

To compile a PR design, you must create a PR implementation revision for each persona. In addition, you must assign revision types for each of the revisions. There are the following revision types:

- Partial Reconfiguration - Base
- Partial Reconfiguration - Persona Implementation

The following table lists the revision name and the revision type for each of the revisions:

Table 3. Revision Names and Types

Revision Name	Revision Type
`blinking_led.qsf`	Partial Reconfiguration - Base
`blinking_led_default.qsf`	Partial Reconfiguration - Persona Implementation
`blinking_led_slow.qsf`	Partial Reconfiguration - Persona Implementation
`blinking_led_empty.qsf`	Partial Reconfiguration - Persona Implementation

Setting the Base Revision Type

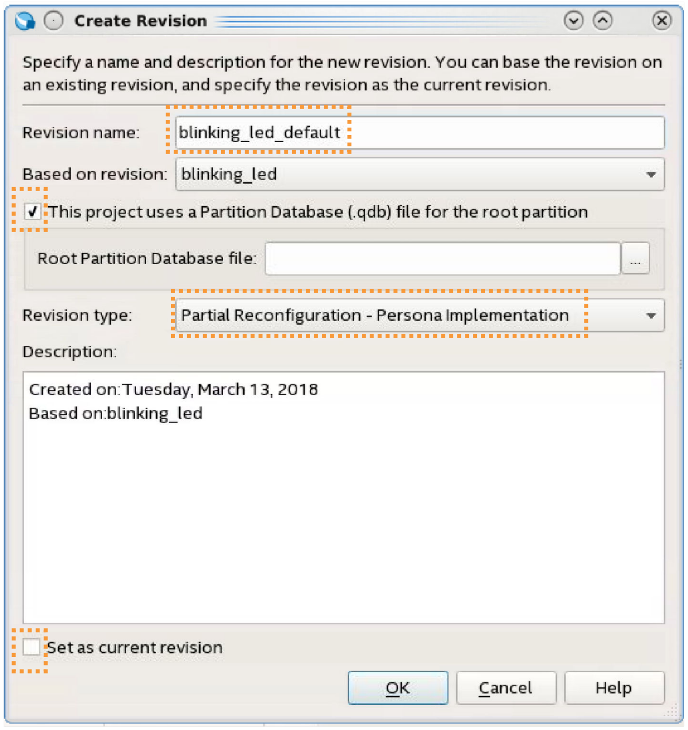
1. Click **Project ➤ Revisions**.
2. In **Revision Name**, select the **blinking_led** revision, and then click **Set Current**.
3. Click **Apply**. The `blinking_led` revision displays as the current revision.
4. To set the **Revision Type** for `blinking_led`, click **Assignments ➤ Settings ➤ General**.
5. For **Revision Type**, select **Partial Reconfiguration - Base**, and then click **OK**.
6. Verify that the `blinking_led.qsf` now contains the following assignment:

```
##blinking_led.qsf
set_global_assignment -name REVISION_TYPE PR_BASE
```

Creating Implementation Revisions

1. To open the **Revisions** dialog box, click **Project ➤ Revisions**.
2. To create a new revision, double-click **<<new revision>>**.
3. In **Revision name**, specify `blinking_led_default` and select **blinking_led** for **Based on revision**.
4. For the **Revision type**, select **Partial Reconfiguration - Persona Implementation**.
5. Enable **This project uses a Partition Database (.qdb) file for the root partition**. You do not need to specify the **Root Partition Database file** at this point. You can input this name at a later stage from the **Design Partitions Window**.

Figure 9. Creating Revisions



Specify a name and description for the new revision. You can base the revision on an existing revision, and specify the revision as the current revision.

Revision name: blinking_led_default
Based on revision: blinking_led
[✓] This project uses a Partition Database (.qdb) file for the root partition
Root Partition Database file:
Revision type: Partial Reconfiguration - Persona Implementation
Description:
Created on: Tuesday, March 13, 2018
Based on: blinking_led
[ ] Set as current revision

6. Similarly, set the **Revision type** for the other revisions:
   - `blinking_led_slow`
   - `blinking_led_empty`
7. Verify that each `.qsf` file now contains the following assignment:

```
set_global_assignment -name REVISION_TYPE PR_IMPL
```

Step 7: Compiling the Base Revision and Exporting the Static Region

Before you begin:

1. Set `blinking_led` as the **Current Revision**.
2. Ensure the `blinking_led.qsf` contains the following assignments:

```
set_global_assignment -name GENERATE_PR_RBF_FILE ON
set_global_assignment -name ON_CHIP_BITSTREAM_DECOMPRESSION OFF
```

   These assignments allow the assembler to automatically generate the required PR bitstreams.

To compile the base revision and export the static region:

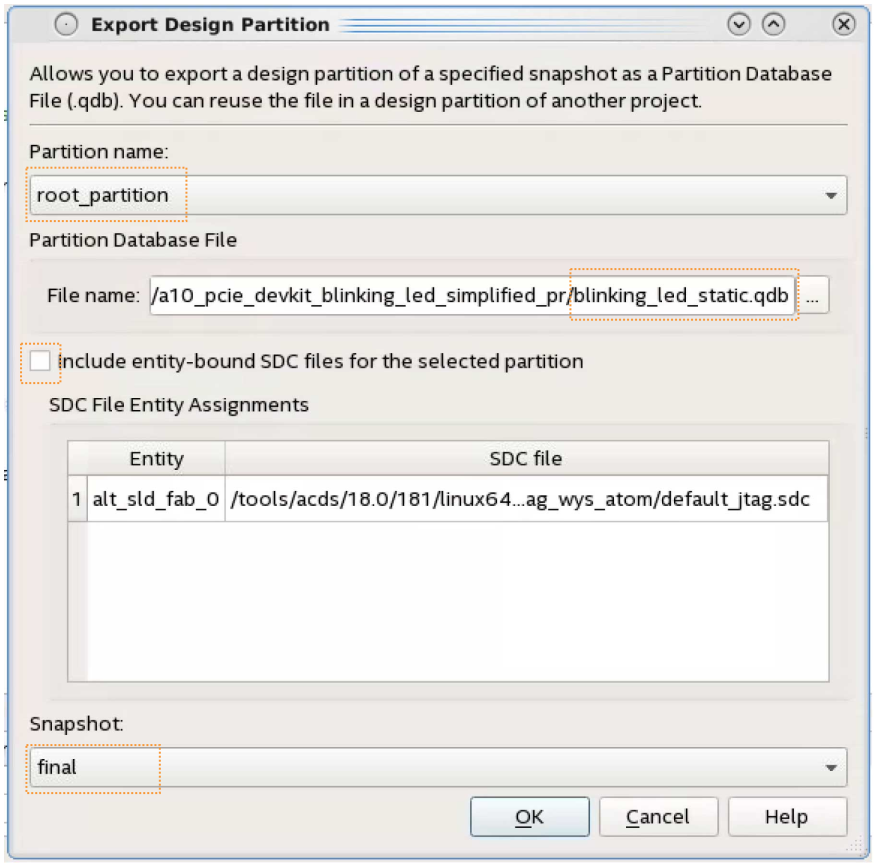
1. To compile the base revision, click **Processing ➤ Start Compilation**.
   Alternatively, the following command compiles the base revision:

```
quartus_sh --flow compile  blinking_led -c blinking_led
```

2. To export the root partition, click **Project ➤ Export Design Partition**, and then specify the following options for the partition:

Option	Setting
Partition name	**root_partition**
Partition database file	`<project>/blinking_led_static.qdb`
Include entity-bound SDC files	Enable
Snapshot	**Final**

Figure 10. Exporting the Static Region


Export Design Partition

Allows you to export a design partition of a specified snapshot as a Partition Database File (.qdb). You can reuse the file in a design partition of another project.

Partition name: root_partition

Partition Database File

File name: /a10_pcie_devkit_blinking_led_simplified_pr/blinking_led_static.qdb

☐ Include entity-bound SDC files for the selected partition

SDC File Entity Assignments

	Entity	SDC file
1	alt_sld_fab_0	/tools/acds/18.0/181/linux64...ag_wys_atom/default_jtag.sdc

Snapshot: final

Alternatively, the following command exports the root partition:

```
quartus_cdb -r  blinking_led -c blinking led --export_block \
        root_partition --snapshot final --file blinking_led_static.qdb
```

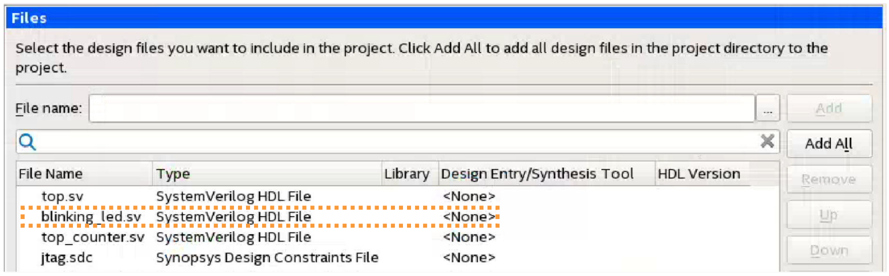
Related Information

- Floorplan the Partial Reconfiguration Design
- Applying Floorplan Constraints Incrementally

Step 8: Preparing PR Implementation Revisions

You must prepare the PR implementation revisions before you can compile and generate the PR bitstream for device programming. This setup includes adding the static region .qdb file as the source file for each implementation revision. In addition, you must specify the corresponding entity of the PR region.

1. To set the current revision, click **Project ➤ Revisions**, select **blinking_default** as the **Revision name**, and then click **Set Current**.
2. To verify the correct source for each implementation revision, click **Project ➤ Add/Remove Files in Project**. The `blinking_led.sv` file appears in the file list.


Files

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project.

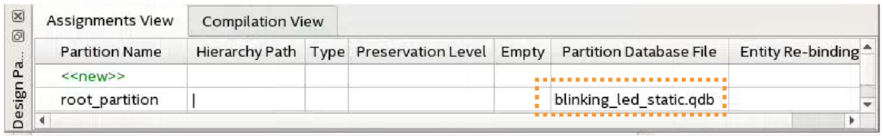
File name:

File Name	Type	Library	Design Entry/Synthesis Tool	HDL Version
top.sv	SystemVerilog HDL File		<None>	
blinking_led.sv	SystemVerilog HDL File		<None>	
top_counter.sv	SystemVerilog HDL File		<None>	
jtag.sdc	Synopsys Design Constraints File		<None>	

3. Repeat steps 1 through 2 to verify the other implementation revision source files:

Implementation Revision Name	Source File
blinking_led_default	`blinking_led.sv`
blinking_led_empty	`blinking_led_empty.sv`
blinking_led_slow	`blinking_led_slow.sv`

4. To verify the `.qdb` file associated with the root partition, click **Assignments ➤ Design Partitions Window**. Confirm that the **Partition Database File** specifies the `blinking_led_static.qdb` file, or double-click the **Partition Database File** cell to specify this file.

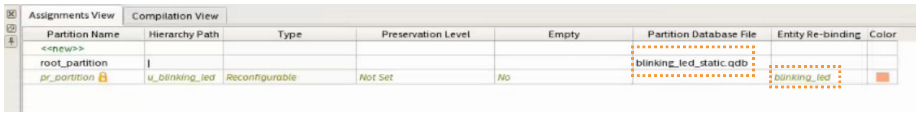

Partition Name	Hierarchy Path	Type	Preservation Level	Empty	Partition Database File	Entity Re-binding
<<new>>						
root_partition	|				blinking_led_static.qdb	

Alternatively, the following command assigns this file:

```
set_instance_assignment -name QDB_FILE_PARTITION \
      blinking_led_static.qdb -to |
```

5. In the **Entity Re-binding** cell, specify the entity name of each PR partition that you change in the implementation revision. For the `blinking_led_default` implementation revision, the entity name is `blinking_led`. In this tutorial, you overwrite the `u_blinking_led` instance from the base revision compile with the new `blinking_led` entity.

Implementation Revision Name	Entity Re-binding
`blinking_led_default`	`blinking_led`
`blinking_led_slow`	`blinking_led_slow`
`blinking_led_empty`	`blinking_led_empty`


Partition Name	Hierarchy Path	Type	Preservation Level	Empty	Partition Database File	Entity Re-binding	Color
<<new>>							
root_partition	|				blinking_led_static.qdb		
pr_partition	u_blinking_led	Reconfigurable	Not Set	No		blinking_led	🟧

Verify that the following line now exists in the `.qsf`:

```
##blinking_led_default.qsf
set_instance_assignment -name ENTITY_REBINDING blinking_led \
       -to u_blinking_led
```

```
##blinking_led_slow.qsf
set_instance_assignment -name ENTITY_REBINDING blinking_led_slow \
        -to u_blinking_led

##blinking_led_empty.qsf
set_instance_assignment -name ENTITY_REBINDING blinking_led_empty \
        -to u_blinking_led
```

6. Before compiling the implementation revision, ensure the `blinking_led_default.qsf` contains the following assignments:

   ```
   set_global_assignment -name GENERATE_PR_RBF_FILE ON
   set_global_assignment -name ON_CHIP_BITSTREAM_DECOMPRESSION OFF
   ```

   These assignments allow the assembler to automatically generate the required PR bitstreams.

7. To compile the design, click **Processing ➤ Start Compilation**. Alternatively, the following command compiles this project:

   ```
   quartus_sh --flow compile blinking_led –c blinking_led_default
   ```

8. Repeat steps 1 through 7 to prepare `blinking_led_slow` and `blinking_led_empty` implementation revisions.

   Note: You can specify any Fitter specific settings that you want to apply during the PR implementation compilation. Fitter specific settings impact only the fit of the persona, without affecting the imported static region.

Step 9: Programming the Board

Before you begin:

1. Connect the power supply to the Intel Arria 10 GX FPGA development board.
2. Connect the USB Blaster cable between your PC USB port and the USB Blaster port on the development board.

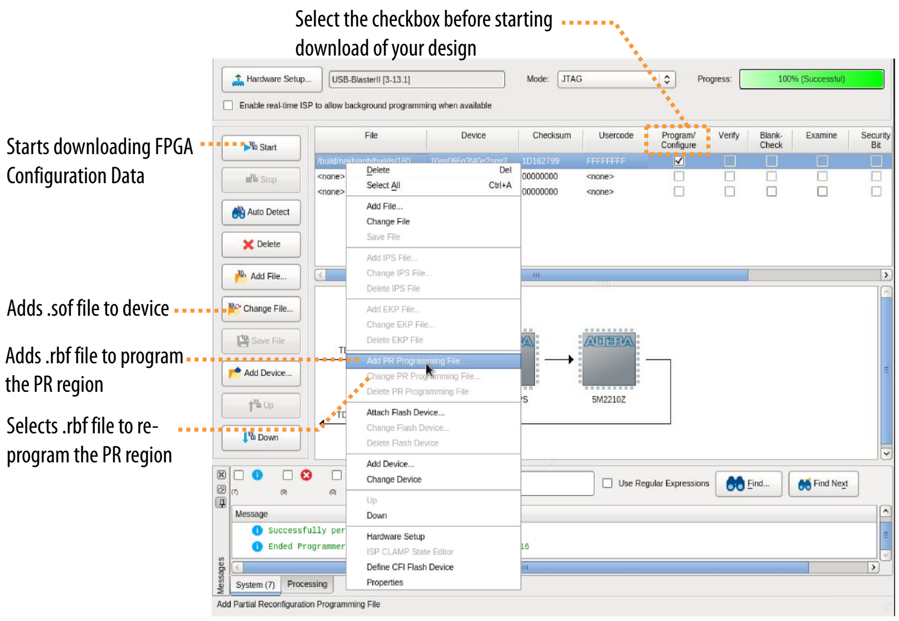
Note: This tutorial utilizes the Intel Arria 10 GX FPGA development board on the bench, outside of the PCIe slot in your host machine.

To run the design on the Intel Arria 10 GX FPGA development board:

1. Open the Intel Quartus Prime software and click **Tools ➤ Programmer**.
2. In the Programmer, click **Hardware Setup** and select **USB-Blaster**.
3. Click **Auto Detect** and select the device, **10AX115S2**.
4. Click **OK**. The Intel Quartus Prime software detects and updates the Programmer with the three FPGA chips on the board.
5. Select the 10AX115S2 device, click **Change File** and load the `blinking_led_default.sof` file.
6. Enable **Program/Configure** for `blinking_led_default.sof` file.
7. Click **Start** and wait for the progress bar to reach 100%.
8. Observe the LEDs on the board blinking at the same frequency as the original flat design.

9. To program only the PR region, right-click the `blinking_led_default.sof` file in the Programmer and click **Add PR Programming File**.
10. Select the `blinking_led_default.pr_partition.rbf` file.
11. Disable **Program/Configure** for `blinking_led_default.sof` file.
12. Enable **Program/Configure** for `blinking_led_slow.pr_partition.rbf` file and click **Start**. On the board, observe `LED[0]` and `LED[1]` continuing to blink. When the progress bar reaches 100%, `LED[2]` and `LED[3]` blink slower.
13. To re-program the PR region, right-click the `.rbf` file in the Programmer and click **Change PR Programing File**.
14. Select the `.rbf` files for the other two personas to observe the behavior on the board. Loading the `blinking_led_default.pr_partition.rbf` file causes the LEDs to blink at a specific frequency, and loading the `blinking_led_empty.pr_partition.rbf` file causes the LEDs to stay ON.

Figure 11. Programming the Intel Arria 10 GX FPGA Development Board



Troubleshooting PR Programming Errors

Ensuring proper setup of the Intel Quartus Prime Programmer and connected hardware helps to avoid any errors during PR programming.

If you face any PR programming errors, refer to *Troubleshooting PR Programming Errors* in the *Partial Reconfiguration User Guide* for step-by-step troubleshooting tips.

Related Information

Troubleshooting PR Programming Errors

Modifying an Existing Persona

You can change an existing persona, even after fully compiling the base revision.

For example, to cause the `blinking_led_slow` persona to blink even slower:

1. In the `blinking_led_slow.sv` file, modify the `COUNTER_TAP` parameter from 27 to 28.
2. Recompile only the `blinking_led_slow` revision. There is no requirement to modify or recompile the other revisions.

Adding a New Persona to the Design

After fully compiling your base revisions, you can still add new personas and individually compile these personas.

For example, to define a new persona that keeps one LED on and the other LED off:

1. Copy `blinking_led_empty.sv` to `blinking_led_wink.sv`.
2. In the `blinking_led_wink.sv` file, modify the assignment, `assign led_three_on = 1'b0;` to `assign led_three_on = 1'b1;`.
3. Create a new implementation revision, `blinking_led_wink`, by following the steps in Creating Implementation Revisions on page 15.

   Note: The `blinking_led_wink` revision must use the `blinking_led_wink.sv` file, and use the `blinking_led_wink` in the entity rebinding assignment.
4. Compile the revision by clicking **Processing ➤ Start Compilation**.

For complete information on partially reconfiguring your design for Intel Arria 10 devices, refer to *Creating a Partial Reconfiguration Design* in Volume 1 of the *Intel Quartus Prime Pro Edition Handbook*.

Related Information

- Creating a Partial Reconfiguration Design
- Partial Reconfiguration Online Training

Document Revision History for AN 797: Partially Reconfiguring a Design on Intel Arria 10 GX FPGA Development Board

Date	Intel Quartus Prime Version	Changes
2018.05.07	18.0.0	• Compilation flow change • Other minor text edits
2017.11.06	17.1.0	• Updated the *Reference Design Requirements* section with software version • Updated the *Flat Reference Design without PR Partitioning* figure with design block changes • Updated the *Reference Design Files* table with information on the `Top_counter.sv` module

continued...

Date	Intel Quartus Prime Version	Changes
		• Updated the *Partial Reconfiguration IP Core Integration* figure with design block changes • Updated the figures - *Design Partitions Window* and *Logic Lock Regions Window* to reflect the new GUI • Text edits
2017.05.08	17.0.0	• Updated software version in *Reference Design Requirements* section • Added information about enable freeze interface option in *Step 4: Adding the Partial Reconfiguration IP Core* section • Added information on the importance of SDC ordering in *Step 4: Adding the Partial Reconfiguration IP Core* section • Added an overview on base, synthesis, and implementation revisions in *Step 6: Creating Revisions* section • Text edits
2016.12.21	16.1.0	Initial release of the document