Partially Reconfiguring a Design on Arria 10 SoC Development Board

2016.06.20

AN-770 Subscribe Send Feedback

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

Partial reconfiguration (PR) allows you to reconfigure a portion of the FPGA dynamically, while the rest of the FPGA design is still functioning. You can 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.

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 board space usage

Note: To understand and implement this reference design, you must have basic familiarity with the Quartus® Prime FPGA implementation flow and know how to update the Quartus Prime project files.

Partial Reconfiguration Concepts

The following table details the partial reconfiguration terminology:

Table 1: Partial Reconfiguration Terminology

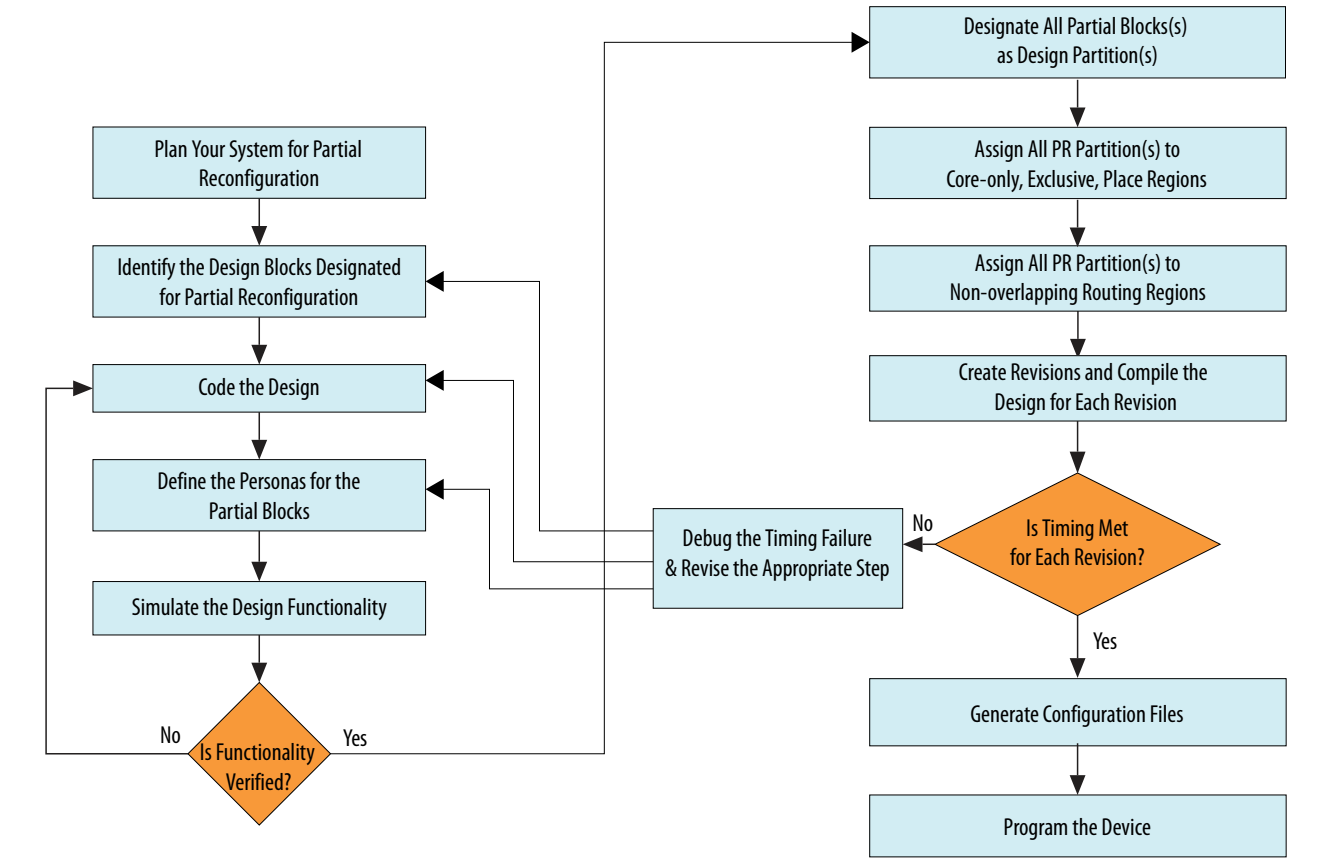
Term	Description
PR partition	Design partition you designate for partial reconfiguration. A PR project can contain one or more partially reconfigurable PR partitions.
PR region	An area in the FPGA that you associate with a partially reconfigurable PR partition. A PR region contains the core locations of the device. A device can contain more than one PR region.
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 the core and periphery locations of the device.

Term	Description
PR persona	A specific PR partition implementation in a PR region. A PR region can contain multiple personas. **Note:** Static regions contain only one persona.
Revision	The set of assignments, settings, and design files for one version of your design. Your Quartus Prime project can contain several revisions. Revision allows you to organize several versions of your design within a single project.

Partial Reconfiguration Design Flow

Partial reconfiguration design flow requires the use of project revisions in the 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 static region and describe the differences in the reconfigurable regions.

Figure 1: Partial Reconfiguration Design Flow



Reference Design Requirements

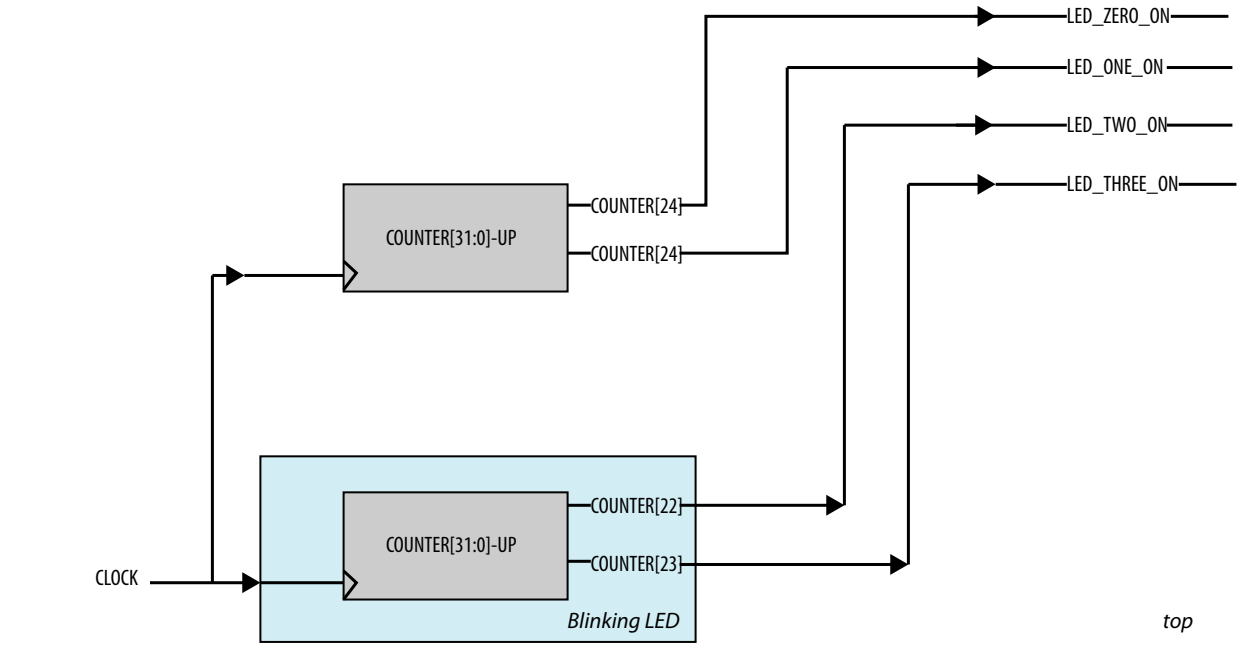
This reference design requires the following:

- Quartus Prime Pro Edition software version 16.0 for the design implementation.
- Arria 10 SoC development kit for the FPGA implementation.

Reference Design Overview

This reference design consists of two 32-bit counters. At the board level, the design connects the clock to a 50MHz source, and connects the output to `USER_LED_FPGA[3:0]`. Selecting the output from the counter bits in a specific sequence causes the LEDs to blink at a specific frequency.

Figure 2: Flat Reference Design without PR Partitioning



Reference Design Files

The partial reconfiguration tutorial is available in the following location:

https://github.com/alterasoftware/design-flows

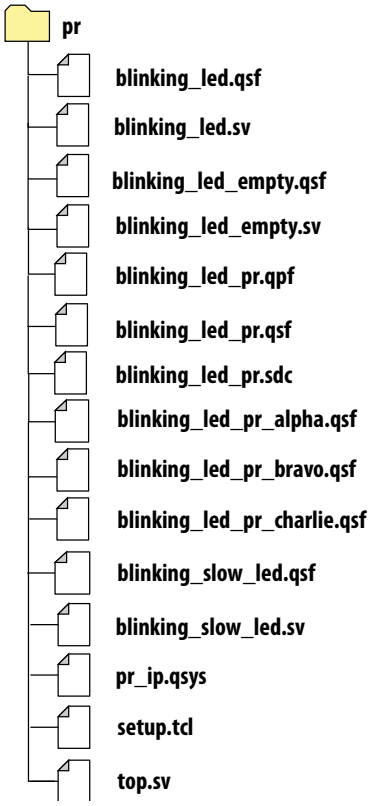
Navigate to the **partial_reconfig/tutorials/a10_soc_devkit_blinking_led** sub-folder to access the reference design. The **flat** folder consists of the following files:

Table 2: 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. The counter at the top-level controls `LED[0]` and `LED[1]`.
blinking_led.sdc	Defines the timing constraints for the project.
blinking_led.sv	System Verilog file that causes the LEDs to blink using a 32-bit counter. The counter controls `LED[2]` and `LED[3]`. This module acts as the PR partition.
blinking_led.qpf	Quartus Prime project file containing the base revision information.
blinking_led.qsf	Quartus Prime settings file containing the assignments and settings for the project.

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

Figure 3: Reference Files



Reference Design Walkthrough

The following steps describe how to apply partial reconfiguration to a flat design using the Quartus Prime Pro Edition software for the Arria 10 SoC development board:

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_soc_devkit_blinking_led_pr**.
2. Copy all the files from **a10_soc_devkit_blinking_led.zip** to the directory, **a10_soc_devkit_blinking_led_pr**.
3. In the Quartus Prime Pro Edition software, click **File** > **Open Project** and select **blinking_led.qpf**.
4. To compile the flat design, click **Processing** > **Start Compilation**.

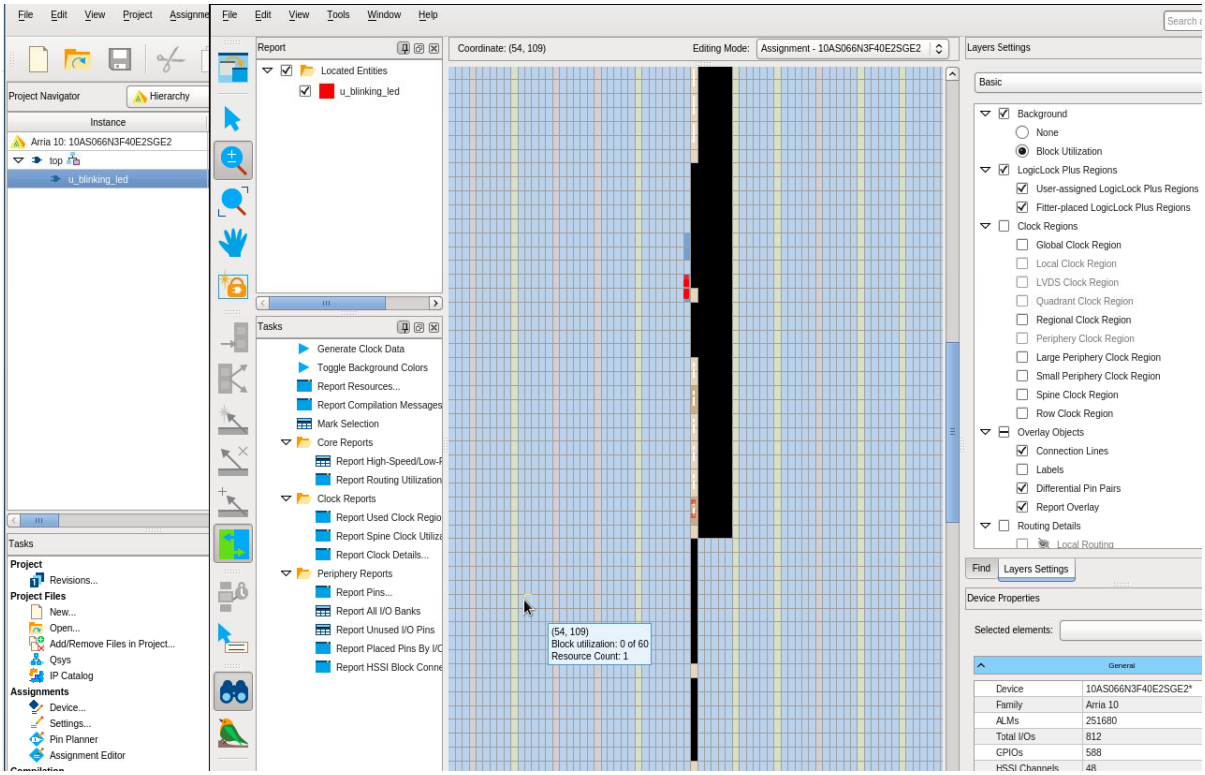
Step 2: Allocating Region for PR Partition

For every 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:

1. Right-click the `u_blinking_led` instance in the **Project Navigator** and click **Locate Node** > **Locate in Chip Planner**.

   Note: The `u_blinking_led` region is color-coded.

Figure 4: Chip Planner Node Location for `blinking_led`



2. To define the PR region, add the following lines of code to the **blinking_led.qsf** file:

   a. To assign the pr_partition name to the `u_blinking_led` PR partition:

   ```
   set_instance_assignment -name PARTITION pr_partition -to u_blinking_led
   ```

   b. To assign floorplan of the partition:

   ```
   set_instance_assignment -name PLACE_REGION "54 109 104 151" -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
   ```

   Note: The order for the box co-ordinates are lower-left corner (`X1 Y1`), followed by upper-right corner (`X2 Y2`).

   Ensure that your placing region encloses the PR partition region.

   c. To assign a routing region for the blocked section:

   ```
   set_instance_assignment -name ROUTE_REGION "53 108 105 152" -to u_blinking_led
   ```

   Note: The design routing region assignments contain a slightly larger value than the placing region to provide extra flexibility for the routing engine when the engine routes different personas.

   d. To set `u_blinking_led` hierarchy as a PR partition, add the following line:

   ```
   set_instance_assignment -name PARTIAL_RECONFIGURATION_PARTITION ON -to
   u_blinking_led
   ```

Note: Manually editing the **.qsf** file is a requirement only in the Quartus Prime Pro Edition software version 16.0.

Figure 5: blinking_led.qsf Updated with PR Partition Region Assignments

```
set_global_assignment -name SDC_FILE blinking_led_pr.sdc
set_global_assignment -name PROJECT_OUTPUT_DIRECTORY output_files
# ------------------------------------------------------------------------ #
# exclusive region for PR:
set_instance_assignment -name PARTITION pr_partition -to u_blinking_led
set_instance_assignment -name PLACE_REGION "54 109 104 151" -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

# define route region slightly larger than place region
set_instance_assignment -name ROUTE_REGION "53 108 105 152" -to u_blinking_led
set_instance_assignment -name PARTIAL_RECONFIGURATION_PARTITION ON -to u_blinking_led
# ------------------------------------------------------------------------ #
```

Step 3: Adding Altera Partial Reconfiguration Control Block IP Core

This reference design uses the Altera Partial Reconfiguration IP core to reconfigure the PR partition. This IP core uses JTAG to reconfigure the PR partition from an external host PR control block. To add the Altera Partial Reconfiguration IP core from the Quartus Prime IP catalog:

1. Type `PR` in the IP catalog.
2. To launch the parameter editor, select the Partial Reconfiguration IP core from the IP library, and click the **Add** button.
3. In the New IP Variation window, specify the **Entity name** as `pr_ip` and click **OK**.

   Note: Use the default parameterization for `pr_ip`.
4. To save the system, click **Finish**, and exit the parameter editor without generating the system. Quartus Prime software creates the **pr_ip.qsys** IP variation file, and adds the file to the **blinking_led** project.

Note: If you are directly copying the **pr_ip.qsys** file from the **a10_soc_devkit_blinking_led_pr.zip** folder, you must manually edit the **blinking_led.qsf** file to include the following line:

```
set_global_assignment -name QSYS_FILE pr_ip.qsys
```

Related Information

Partial Reconfiguration IP Core User Guide

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),
        .double_pr      (1'b0),             // ignored for JTAG
        .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 by using the freeze control signal output from PR_IP, uncomment the following lines of code:

```
assign led_two_on    = freeze ? 1'b1 : pr_led_two_on;
assign led_three_on  = freeze ? 1'b1 : pr_led_three_on;
```

3. To assign 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)
    );
```

Figure 6: top.sv File Updated with PR_IP Instance

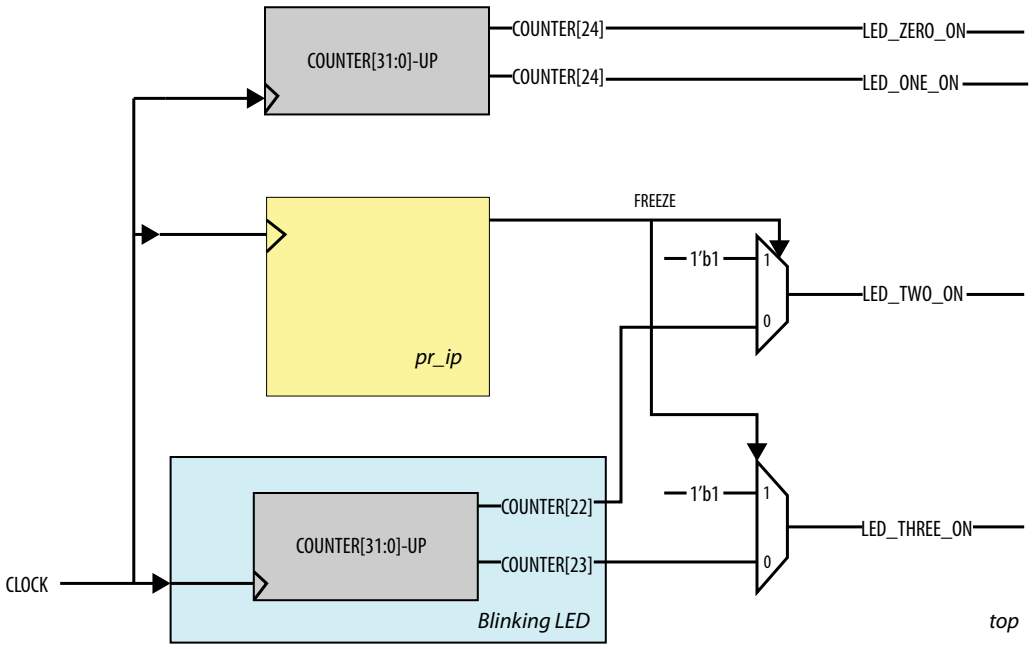
```
//////////////////////////////////////////////////////////////////
// The output ports driven by PR logic need to stablized
// during PR reprograming.  Using "freeze" control signal,
// from PR IP, to drive these output to logic 1 during
// reconfiguration
//////////////////////////////////////////////////////////////////
assign led_two_on    = freeze ? 1'b1 : pr_led_two_on;
assign led_three_on  = freeze ? 1'b1 : pr_led_three_on;

//////////////////////////////////////////////////////////////////
// instance of the default persona
//////////////////////////////////////////////////////////////////
blinking_led u_blinking_led
(
    .led_two_on    (pr_led_two_on),
    .led_three_on  (pr_led_three_on),

    .clock         (clock)
);

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

Figure 7: Integration with Altera Partial Reconfiguration IP Core



Step 4: 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 System Verilog files, **blinking_led.sv**, **blinking_slow_led.sv**, and **blinking_led_empty.sv** in your working directory for the three personas:

Table 3: 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 (// Control signals for the LEDs led_two_on, led_three_on, // clock clock); // assuming single bit control signal to turn LED 'on' output wire led_two_on; output wire led_three_on; // clock input wire clock;

File Name	Description	Code

		`// the 32-bit counter`
		`reg [31:0] count;`
		
		`localparam COUNTER_TAP = 23;`
		
		`// The counter:`
		`always_ff @(posedge clock)`
		`begin`
		`count <= count + 1;`
		`end`
		
		`assign led_two_on = count[COUNTER_TAP];`
		`assign led_three_on = ~count[COUNTER_TAP];`
		
		`endmodule`

blinking_slow_led.sv	LEDs blink slower.	`` `timescale 1 ps / 1 ps ``
		`` `default_nettype none ``
		
		`module blinking_slow_led (`
		
		`// Control signals for the LEDs`
		`led_two_on,`
		`led_three_on,`
		
		`// clock`
		`clock`
		`);`
		
		`// assuming single bit control signal to turn LED 'on'`
		`output wire led_two_on;`
		`output wire led_three_on;`
		
		`// clock`
		`input wire clock;`
		
		`// the 32-bit counter`
		`reg [31:0] count;`
		
		`localparam COUNTER_TAP = 27;`
		
		`// The counter:`
		`always_ff @(posedge clock)`
		`begin`
		`count <= count + 1;`
		`end`
		
		`assign led_two_on = count[COUNTER_TAP];`
		`assign led_three_on = ~count[COUNTER_TAP];`
		
		`endmodule`

blinking_led_empty.sv	LEDs stay ON.	`` `timescale 1 ps / 1 ps ``
		`` `default_nettype none ``
		
		`module blinking_led_empty (`
		
		`// Control signals for the LEDs`
		`led_two_on,`
		`led_three_on,`

File Name	Description	Code

		`// clock`
		`clock`
		`);`
		
		`// Control signal to turn LED 'on'`
		`output wire led_two_on;`
		`output wire led_three_on;`
		
		`// clock`
		`input wire clock;`
		
		`// LED is active low`
		`assign led_two_on = 1'b0;`
		`assign led_three_on = 1'b0;`
		
		`endmodule`

2. To include the personas in the project, add the below lines to the **blinking_led.qsf** file:

```
set_global_assignment -name SYSTEMVERILOG_FILE blinking_slow_led.sv
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led_empty.sv
```

Step 5: Creating Revisions

To compile a PR design, you must create a PR implementation revision and synthesis revision for each persona. In this reference design, the three personas contain three separate revisions:

Table 4: Revisions for the Three Personas

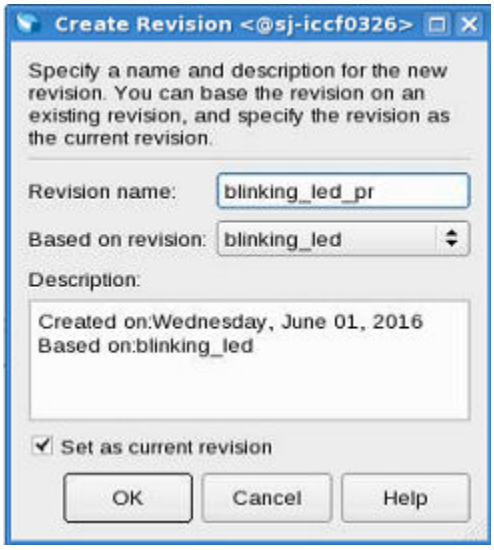
Persona	Revision
`blinking_led`	`blinking_led_pr_alpha`
`blinking_slow_led`	`blinking_led_pr_bravo`
`blinking_led_empty`	`blinking_led_pr_charlie`

To create the PR implementation revisions from the Quartus Prime Pro Edition software:

1. To open the Revisions window, click **Project** > **Revisions**.
2. To create a new revision, double-click **<<new revision>>**.
3. Specify the **Revision name** as `blinking_led_pr` and select `blinking_led` for **Based on Revision**.
4. Enable **Set as Current Revision** and click **OK**.
5. Similarly, create `blinking_led_pr_alpha`, `blinking_led_pr_bravo`, and `blinking_led_pr_charlie` revisions based on the `blinking_led_pr` revision.

   Note: Do not set the above three revisions as current revision.

Figure 8: Creating Revisions



Creating Synthesis-Only Revisions

To create synthesis-only revisions for the personas:

1. Modify **blinking_led.qsf** file to include only the following assignments:

```
set_global_assignment -name FAMILY "Arria 10"
set_global_assignment -name DEVICE 10AS066N3F40E2SGE2
set_global_assignment -name TOP_LEVEL_ENTITY blinking_led
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led.sv
```

2. In the Quartus Prime software, click **Project** > **Revisions**.
3. Create `blinking_led_empty` and `blinking_slow_led` revisions based on `blinking_led` revision.

   Note: Do not set the above revisions as current revision.
4. Update the **blinking_slow_led.qsf** and **blinking_led_empty.qsf** files with their corresponding TOP_LEVEL_ENTITY and SYSTEMVERILOG_FILE assignments:

```
##blinking_slow_led.qsf
set_global_assignment -name FAMILY "Arria 10"
set_global_assignment -name DEVICE 10AS066N3F40E2SGE2
set_global_assignment -name TOP_LEVEL_ENTITY blinking_slow_led
set_global_assignment -name SYSTEMVERILOG_FILE blinking_slow_led.sv


##blinking_led_empty.qsf
set_global_assignment -name FAMILY "Arria 10"
set_global_assignment -name DEVICE 10AS066N3F40E2SGE2
set_global_assignment -name TOP_LEVEL_ENTITY blinking_led_empty
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led_empty.sv
```

5. Verify that the **blinking_led.qpf** file contains the following revisions, in no particular order:

```
PROJECT_REVISION = "blinking_led_pr"
PROJECT_REVISION = "blinking_led_pr_alpha"
PROJECT_REVISION = "blinking_led_pr_bravo"
```

```
PROJECT_REVISION = "blinking_led_pr_charlie"
PROJECT_REVISION = "blinking_led"
PROJECT_REVISION = "blinking_slow_led"
PROJECT_REVISION = "blinking_led_empty"
```

Note: If you are copying the revision files from **a10_soc_devkit_blinking_led.zip** folder, you must manually update the **blinking_led.qpf** file with the above lines of code.

Step 6: Generating the Partial Reconfiguration Flow Script

To generate the partial reconfiguration flow script:

1. From the Quartus Prime command shell, create a flow template by running the following command:

```
quartus_sh --write_flow_template -flow a10_partial_reconfig
```

Quartus Prime generates the **a10_partial_reconfig.tcl** file. Do not modify this file.

2. Define a **setup.tcl** file with the following configuration that overrides the variable settings in the **a10_partial_reconfig.tcl** file:
   a. Define the name of the project:

```
define_project blinking_led_pr
```

   b. Define the base revision:

```
define_base_revision blinking_led_pr
```

   c. Define each of the partial reconfiguration implementation revisions, along with the PR partition names and the synthesis revision that implements the revisions:

```
define_pr_revision -impl_rev_name blinking_led_pr_alpha \
        -impl_block [list pr_partition blinking_led]

define_pr_revision -impl_rev_name blinking_led_pr_bravo \
        -impl_block [list pr_partition blinking_slow_led]

define_pr_revision -impl_rev_name blinking_led_pr_charlie \
        -impl_block [list pr_partition blinking_led_empty]
```

Step 7: Running the Partial Reconfiguration Flow Script

Before you begin

Exit the Quartus Prime software before running the synthesis from Quartus Prime shell.

To run the partial reconfiguration flow script:

1. From the Quartus Prime command shell, run the Arria 10 partial reconfiguration flow script:

```
quartus_sh -t a10_partial_reconfig.tcl -setup_script setup.tcl
```

This command runs the synthesis for the three personas. Quartus Prime generates a SRAM Object File (**.sof**) for each design, along with a Raw Binary File (**.rbf**) for each of the personas

Step 8: Programming the Board

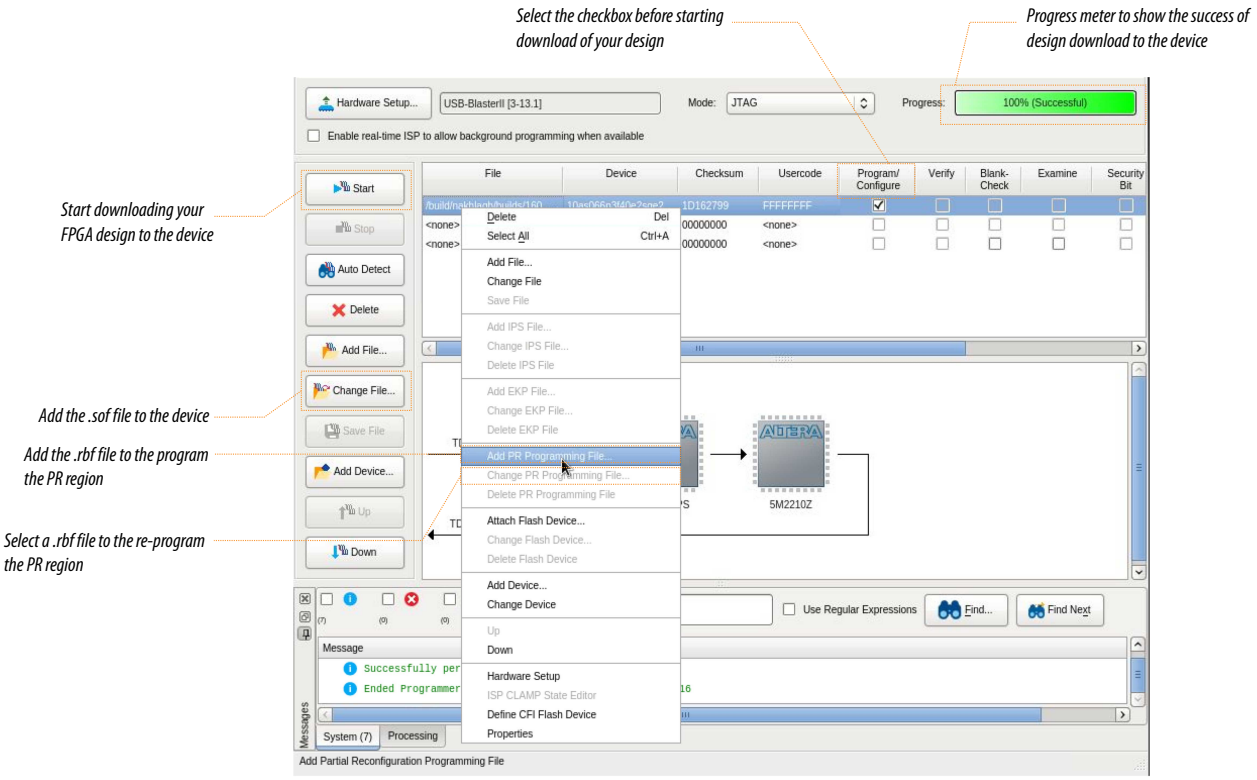
Before you begin

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

To run the design on the Arria 10 SoC development board:

1. Open the 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, **10AS066N3E2**.
4. Click **OK**. The Quartus Prime software detects and updates the Programmer with the three FPGA chips on the board.
5. Select the 10AS066N3E2 device, click **Change File** and load the **blinking_led_pr_alpha.sof** file.
6. Enable **Program/Configure** for **blinking_led_pr_alpha.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_pr_alpha.sof** file in the Programmer and click **Add PR Programming File**.
10. Select the **blinking_led_pr_bravo.rbf** file.
11. Disable **Program/Configure** for **blinking_led_pr_alpha.sof** file.
12. Enable **Program/Configure** for **blinking_led_pr_bravo.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_pr_alpha.rbf** file causes the LEDs to blink at a specific frequency, and loading the **blinking_led_pr_charlie.rbf** file causes the LEDs to stay ON.

Figure 9: Programming the Arria 10 SoC development board



Related Information

- Design Planning for Partial Reconfiguration
- Partial Reconfiguration Online Training

Document Revision History

Table 5: Document Revision History

Date	Version	Changes
2016.07.07	16.0.0	Initial release of the document.