Partially Reconfiguring a Design on Arria 10 SoC Development Board

AN-770
2016.10.31

Contents

1 Partially Reconfiguring a Design on Arria 10 SoC Development Board

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) 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.

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

Note:

- Implementation of this reference design requires basic familiarity with the Quartus® Prime FPGA implementation flow and knowledge of the primary Quartus Prime project files.

Related Links

- Arria 10 SoC Development Kit User Guide
- Partial Reconfiguration Concepts
- Partial Reconfiguration Design Flow
- Partial Reconfiguration Design Recommendations
- Partial Reconfiguration Design Considerations

1.1 Reference Design Requirements

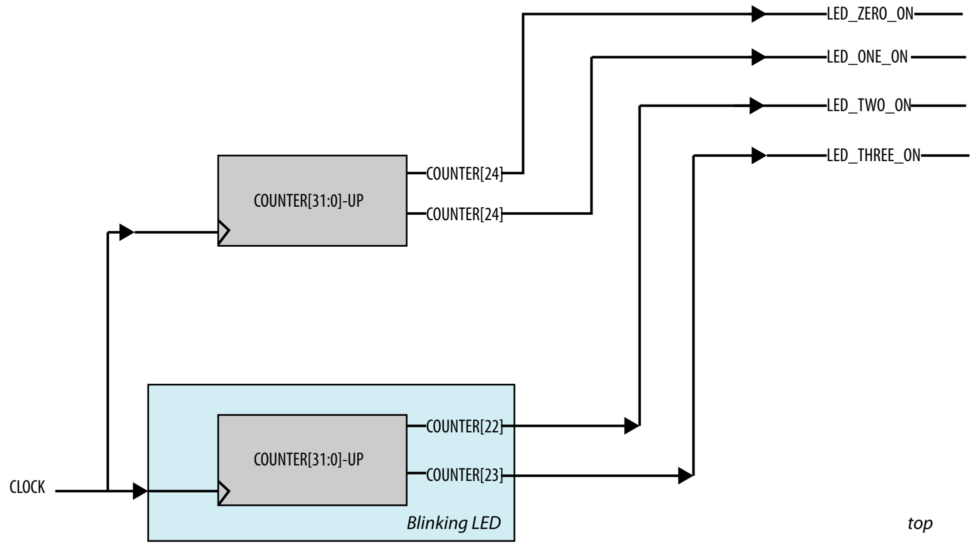
This reference design requires the following:

- Quartus Prime Pro Edition software version 16.1 for the design implementation.
- Arria 10 SoC development kit for the FPGA implementation.

1.2 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 1. Flat Reference Design without PR Partitioning



1.3 Reference Design Files

The partial reconfiguration tutorial is available in the following location:

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

To download the tutorial:

1. Click **Clone or download**.
2. Click **Download ZIP**. Unzip the `design-flows-master.zip` file.
3. 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 1. Reference Design Files

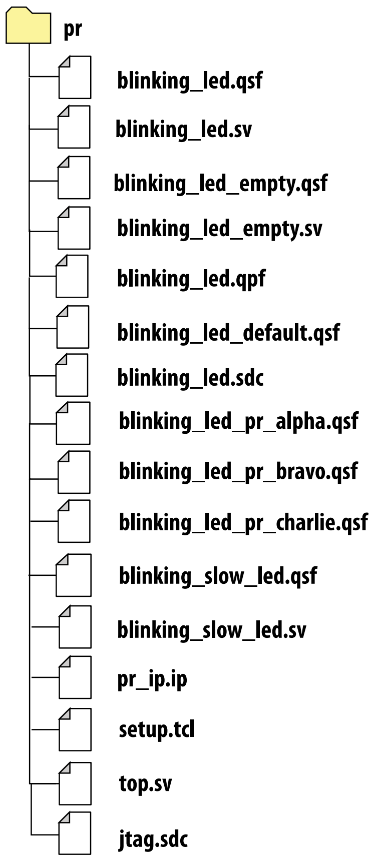
File Name	Description
`top.sv`	Top-level file containing the flat implementation of the design. This module instantiates the `blinking_led` sub-partition. The counter at the top-level controls `LED[0]` and `LED[1]`.
`blinking_led.sdc`	Defines the timing constraints for the project.

continued...

File Name	Description
`blinking_led.sv`	SystemVerilog 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 using this application note. Reference these files at any point during the walkthrough.

Figure 2. Reference Files



1.4 Reference Design Walkthrough

The following steps describe the application of partial reconfiguration to a flat design. The tutorial uses the Quartus Prime Pro Edition software for the Arria 10 SoC 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 Partial Reconfiguration IP Core on page 8

- Step 5: Defining Personas on page 11
- Step 6: Creating Revisions on page 12
- Step 7: Generating the Partial Reconfiguration Flow Script on page 16
- Step 8: Running the Partial Reconfiguration Flow Script on page 16
- Step 9: Programming the Board on page 17

1.4.1 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 the downloaded `design-flows-master/partial_reconfig/tutorials/a10_soc_devkit_blinking_led/flat` sub-folder 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**.

1.4.2 Step 2: Creating a Design Partition

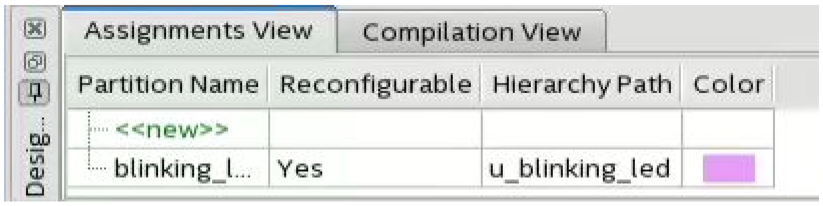
You must create design partitions for each PR region that you want to partially reconfigure. You can create any number of independent partitions or PR regions in your design. This tutorial creates a design partition for the `u_blinking_led` instance.

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 Reconfigurable Design Partition**.

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

Figure 3. Design Partitions Window



Partition Name	Reconfigurable	Hierarchy Path	Color
<<new>>			
blinking_l...	Yes	u_blinking_led	🟪

2. 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 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 Links

Create Design Partitions for Partial Reconfiguration

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

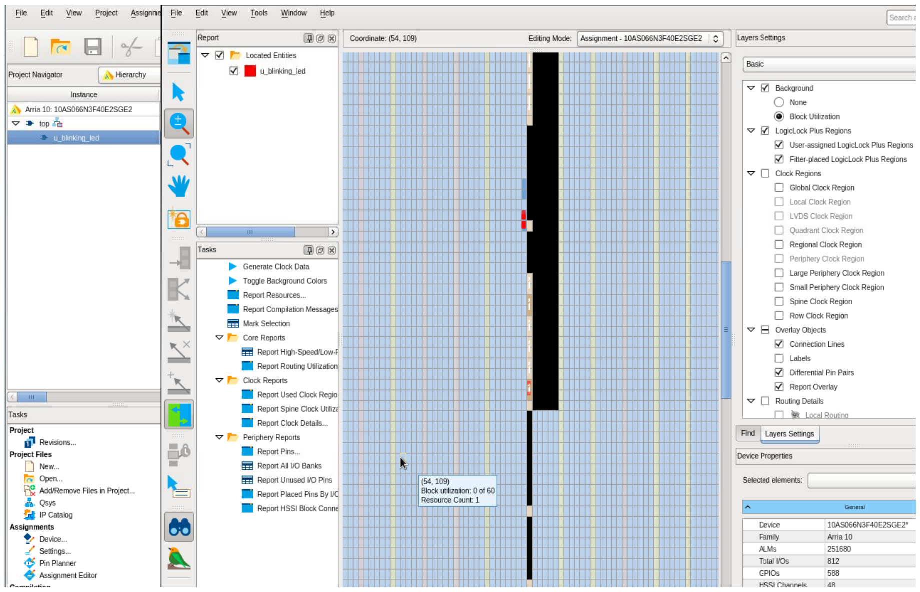
For every base revision you create, the PR design flow uses your PR partition region allocation to place the corresponding persona core in the reserved region. To locate and assign the PR region in the device floorplan for your base revision:

1. Right-click the `u_blinking_led` instance in the **Project Navigator** and click **LogicLock® Plus Region ➤ Create New LogicLock Plus Region**. The region appears on the LogicLock Plus 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 LogicLock Plus Regions Window and click **Locate Node ➤ Locate in Chip Planner**.

   The `u_blinking_led` region is color-coded.

Figure 4. Chip Planner Node Location for `blinking_led`



(54, 109)
Block utilization: 0 of 60
Resource Count: 1

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

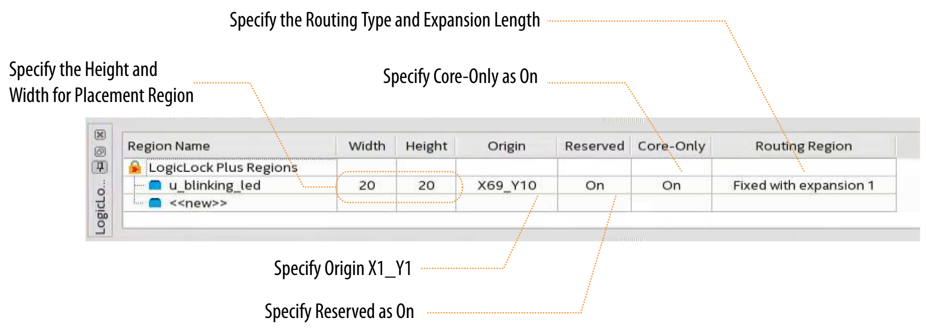
3. Specify the placement region co-ordinates in the **Origin** column. The origin corresponds to the lower-left corner of the region. For example, to set a placement region with (X1 Y1) co-ordinates as (69 10), specify the **Origin** as X69_Y10. The Quartus Prime software automatically calculates the (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 **LogicLock Plus 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 routing engine when the engine routes different personas.

Figure 5. LogicLock Plus Regions Window


Specify the Routing Type and Expansion Length
Specify the Height and Width for Placement Region
Specify Core-Only as On
LogicLo...
Specify Origin X1_Y1
Specify Reserved as On

Region Name	Width	Height	Origin	Reserved	Core-Only	Routing Region
LogicLock Plus Regions						
u_blinking_led	20	20	X69_Y10	On	On	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 Links

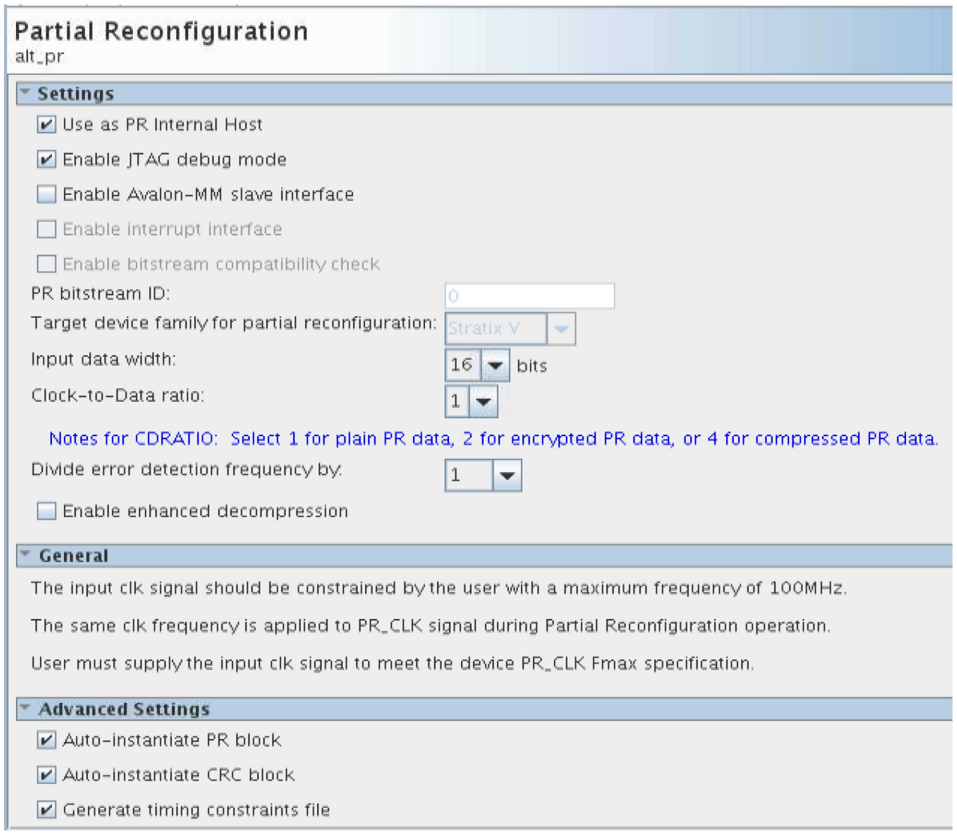
- Floorplan the Partial Reconfiguration Design
- Incrementally Implementing Partial Reconfiguration

1.4.4 Step 4: Adding the Partial Reconfiguration IP Core

Use the Partial Reconfiguration IP core to reconfigure the PR partition. This IP core uses JTAG to reconfigure the PR partition. To add the Partial Reconfiguration IP core to your Quartus Prime project:

1. Type `Partial Reconfiguration` in the IP catalog.
2. To launch the IP Parameter Editor Pro window, select the Partial Reconfiguration IP core from the IP library, and click the **Add** button.
3. In the **New IP Variation** dialog box, type `pr_ip` as the **Entity name** and click **OK**. Use the default parameterization for `pr_ip`. Ensure that the **Enable JTAG debug mode** option is turned on.

Figure 6. Partial Reconfiguration IP Core Parameters


Partial Reconfiguration
alt_pr

Settings

- [✓] Use as PR Internal Host
- [✓] Enable JTAG debug mode
- [ ] Enable Avalon-MM slave interface
- [ ] Enable interrupt interface
- [ ] Enable bitstream compatibility check

PR bitstream ID: 0
Target device family for partial reconfiguration: Stratix V
Input data width: 16 bits
Clock-to-Data ratio: 1

Notes for CDRATIO: Select 1 for plain PR data, 2 for encrypted PR data, or 4 for compressed PR data.

Divide error detection frequency by: 1

- [ ] Enable enhanced decompression

General

The input clk signal should be constrained by the user with a maximum frequency of 100MHz.

The same clk frequency is applied to PR_CLK signal during Partial Reconfiguration operation.

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

Advanced Settings

- [✓] Auto-instantiate PR block
- [✓] Auto-instantiate CRC block
- [✓] Generate timing constraints file

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

Notes:

1. If you are copying the `pr_ip.ip` file from the `a10_soc_devkit_blinking_led_pr.zip` folder, manually edit the `blinking_led.qsf` file to include the following line:

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

2. 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 IP core.

Related Links

Partial Reconfiguration IP Core User Guide

1.4.4.1 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     ()
    );
```

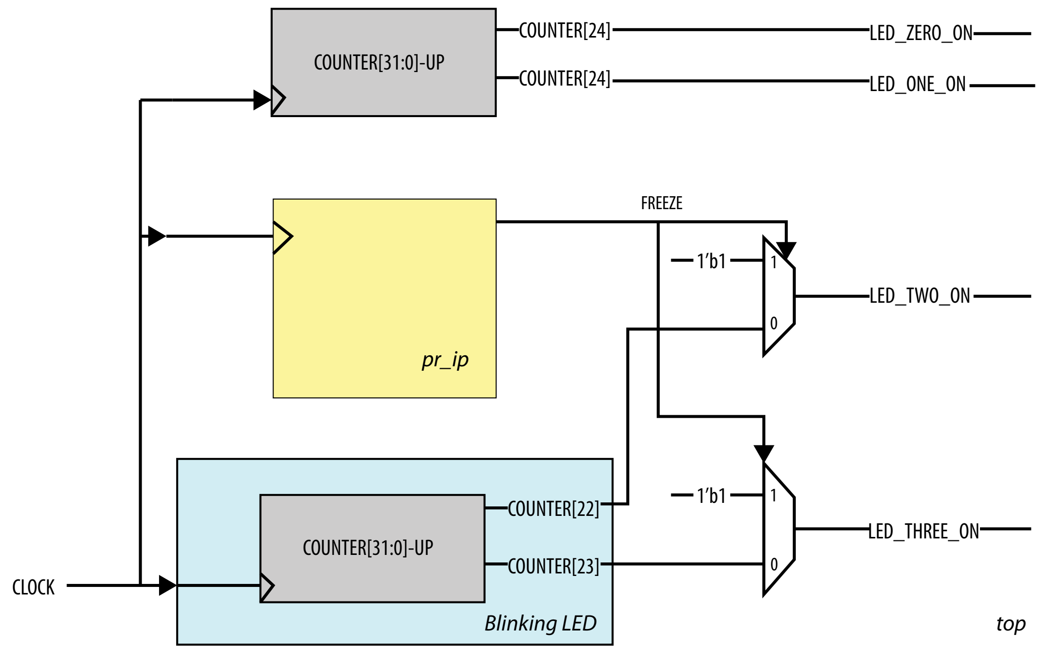
2. To force the output ports to logic 1 during reconfiguration, use the freeze control signal output from PR_IP. Uncomment the following lines of code:

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

Figure 7. Partial Reconfiguration IP Core Integration



1.4.5 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_slow_led.sv`, and `blinking_led_empty.sv` in your working directory for the three personas.

   Note: If you create the SystemVerilog files from the 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 (

		    // 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 = 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)
		```

continued...

File Name	Description	Code

		`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,`
		
		`    // 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`

1.4.6 Step 6: 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 a base revision, three separate synthesis revisions, and three separate implementation revisions:

Table 3. Revisions for the Three Personas

Synthesis Revision	Implementation Revision
blinking_led_default	blinking_led_pr_alpha
blinking_slow_led	blinking_led_pr_bravo
blinking_led_empty	blinking_led_pr_charlie

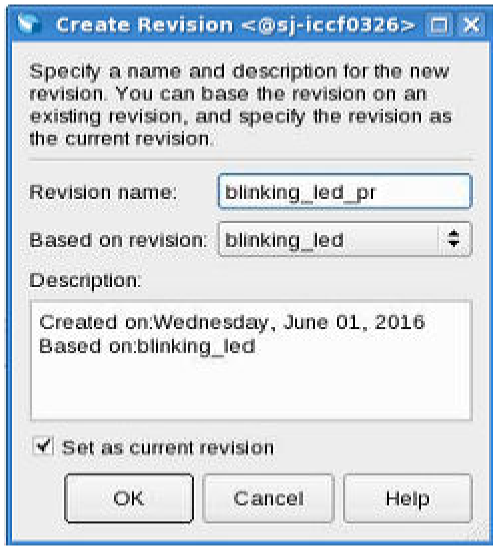
1.4.6.1 Creating Implementation Revisions

To create the PR implementation revisions:

1. To open the **Revisions** dialog box, click **Project ➤ Revisions**.
2. To create a new revision, double-click **<<new revision>>**.
3. Specify the **Revision name** as `blinking_led_pr_alpha` and select `blinking_led` for **Based on Revision**.
4. Similarly, create `blinking_led_pr_bravo` and `blinking_led_pr_charlie` revisions, based on the `blinking_led` revision.

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

Figure 8. Creating Revisions



Create Revision <@sj-iccf0326>

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_pr

Based on revision: blinking_led

Description:

Created on:Wednesday, June 01, 2016
Based on:blinking_led

✓ Set as current revision

1.4.6.2 Creating Synthesis-Only Revisions

To create synthesis-only revisions for the personas, you must assign the top-level entity and corresponding SystemVerilog file for each of the personas:

1. In the Quartus Prime software, click **Project ➤ Revisions**.
2. Create `blinking_led_default` revision based on `blinking_led` revision. Do not set this revision as current revision.
3. Modify `blinking_led_default.qsf` file to include the following assignments:

```
set_global_assignment -name TOP_LEVEL_ENTITY blinking_led
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led.sv
```

4. Similarly, create `blinking_led_empty` and `blinking_slow_led` revisions based on `blinking_led` revision. Do not set these revisions as current revision.

5. 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 TOP_LEVEL_ENTITY blinking_slow_led
set_global_assignment -name SYSTEMVERILOG_FILE blinking_slow_led.sv
```

```
##blinking_led_empty.qsf
set_global_assignment -name TOP_LEVEL_ENTITY blinking_led_empty
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led_empty.sv
```

6. To avoid synthesis errors, ensure that the synthesis revision files do not contain any design partition or LogicLock Plus region assignments. Comment out these assignments, if any, in the `blinking_led_default.qsf`, `blinking_slow_led.qsf`, and `blinking_led_empty.qsf` files:

```
#set_instance_assignment -name PARTITION pr_partition -to u_blinking_led
#set_instance_assignment -name PARTITION_COLOUR 4294935709 -to u_blinking_led
#set_instance_assignment -name PARTIAL_RECONFIGURATION_PARTITION ON -to
u_blinking_led
#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 PARTITION_COLOUR 4288610222 -to top
#set_instance_assignment -name ROUTE_REGION "68 9 89 30" -to u_blinking_led
```

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

```
PROJECT_REVISION = "blinking_led"
PROJECT_REVISION = "blinking_led_pr_alpha"
PROJECT_REVISION = "blinking_led_pr_bravo"
PROJECT_REVISION = "blinking_led_pr_charlie"
PROJECT_REVISION = "blinking_led_default"
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, manually update the `blinking_led.qpf` file with the above lines of code.

1.4.6.3 Specifying Revision Type

You must assign revision type for each of your revisions. There are three revision types:

- Partial Reconfiguration - Base
- Partial Reconfiguration - Persona Synthesis
- Partial Reconfiguration - Persona Implementation

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

Table 4. Revision Types

Revision Name	Revision Type
`blinking_led.qsf`	Partial Reconfiguration - Base
`blinking_led_default.qsf`	Partial Reconfiguration - Persona Synthesis
`blinking_led_empty.qsf`	Partial Reconfiguration - Persona Synthesis
`blinking_slow_led.qsf`	Partial Reconfiguration - Persona Synthesis
`blinking_led_pr_alpha.qsf`	Partial Reconfiguration - Persona Implementation
`blinking_led_pr_bravo.qsf`	Partial Reconfiguration - Persona Implementation
`blinking_led_pr_charlie.qsf`	Partial Reconfiguration - Persona Implementation

To specify the revision type:

1. Click **Project ➤ Revisions**. The **Revisions** dialog box appears.
2. Select `blinking_led` in the Revision Name column, and click **Set Current**.
3. Click **Apply**. The `blinking_led` revision opens.
4. To set the revision type for `blinking_led`, click **Assignments ➤ Settings**.
5. Select the **Revision Type** as **Partial Reconfiguration - Base**.
6. Similarly, set the revision types for the other six revisions, as listed in the above table.

   Note: You must set each revision as the current revision before assigning the revision type.

Verify that the `.qsf` file for each of the revisions contains the following assignment:

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

##blinking_led_default.qsf
set_global_assignment -name REVISION_TYPE PR_SYN

##blinking_slow_led.qsf
set_global_assignment -name REVISION_TYPE PR_SYN

##blinking_led_empty.qsf
set_global_assignment -name REVISION_TYPE PR_SYN

##blinking_led_pr_alpha.qsf
set_global_assignment -name REVISION_TYPE PR_IMPL

##blinking_led_pr_bravo.qsf
set_global_assignment -name REVISION_TYPE PR_IMPL

##blinking_led_pr_charlie.qsf
set_global_assignment -name REVISION_TYPE PR_IMPL
```

Note: Add any Fitter specific settings that you wish to use in the PR implementation compile to the persona implementation revisions. The Fitter specific settings affect the fit of the persona, but do not affect the imported static region. You can also add any synthesis specific settings to individual persona synthesis revisions.

Related Links

Create Revisions for Personas

1.4.7 Step 7: 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/flow.tcl` file.

2. Rename the generated `a10_partial_reconfig/setup.tcl.example` to `a10_partial_reconfig/setup.tcl`, and modify the script to specify your partial reconfiguration project details:

   a. To define the name of the project, update the following line:

   ```
   define_project blinking_led
   ```

   b. To define the base revision, update the following line:

   ```
   define_base_revision blinking_led
   ```

   c. To define each of the partial reconfiguration implementation revisions, along with the PR partition names and the synthesis revision that implements the revisions, update the following lines:

   ```
   define_pr_impl_partition -impl_rev_name blinking_led_pr_alpha \
       -partition_name pr_partition\
       -source_rev_name blinking_led_default

   define_pr_impl_partition -impl_rev_name blinking_led_pr_charlie \
       -partition_name pr_partition\
       -source_rev_name blinking_led_empty

   define_pr_impl_partition -impl_rev_name blinking_led_pr_bravo \
       -partition_name pr_partition\
       -source_rev_name blinking_slow_led
   ```

1.4.8 Step 8: Running the Partial Reconfiguration Flow Script

To run the partial reconfiguration flow script:

1. Click **Tools ➤ Tcl Scripts**. The **Tcl Scripts** dialog box appears.
2. Click **Add to Project**, browse and select the `a10_partial_reconfig/flow.tcl`.
3. Select the `a10_partial_reconfig/flow.tcl` in the Libraries pane, and click **Run**.
   This script runs the synthesis for the three personas. Quartus Prime generates a SRAM Object File (`.sof`), a Partial-Masked SRAM Object File (`.pmsf`), and a Raw Binary File (`.rbf`) for each of the personas.

Note: To run the script from the Quartus Prime command shell, type the following command:

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

Related Links

- Compile the Partial Reconfiguration Design
- Using the Partial Reconfiguration Flow Script
- Configuring the Partial Reconfiguration Flow Script
- Generate Programming Files

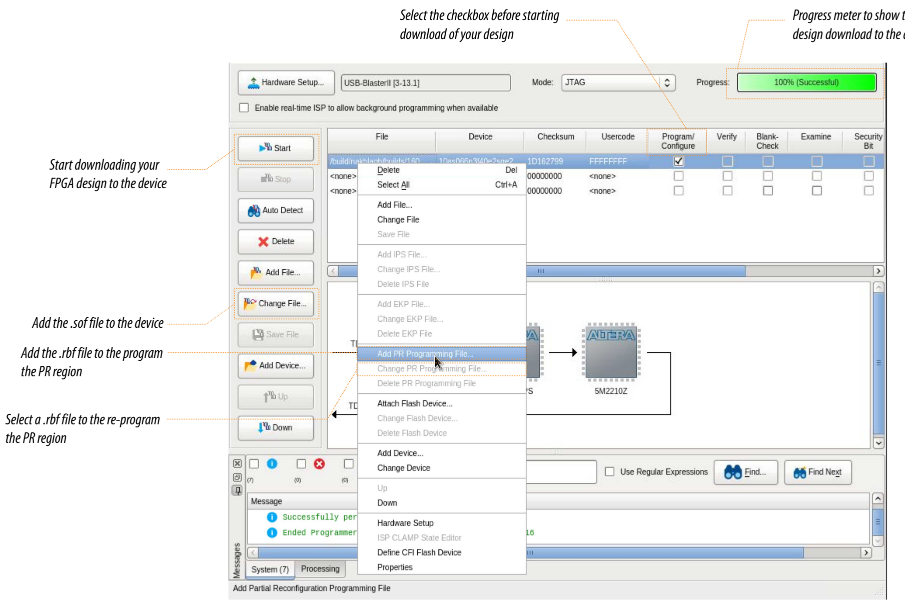
1.4.9 Step 9: Programming the Board

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, **10AS066N3**.
4. Click **OK**. The Quartus Prime software detects and updates the Programmer with the three FPGA chips on the board.
5. Select the 10AS066N3 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



1.4.10 Modifying an Existing Persona

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

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

1. In the `blinking_slow_led.sv` file, modify the `COUNTER_TAP` parameter from 27 to 28.
2. To re-synthesize and re-implement just this persona, run the following command:

```
quartus_sh -t a10_partial_reconfig/flow.tcl -setup_script
a10_partial_reconfig/setup.tcl -impl blinking_led_pr_bravo
```

   This command re-synthesizes the `blinking_slow_led` synthesis revision, and then runs the PR implementation compile using `blinking_led_pr_bravo`. Follow the steps in Step 9: Programming the Board on page 17 to program the resulting RBF file into the FPGA.

   Note: This command does not recompile the base revision.

1.4.11 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 synthesis revision, `blinking_led_wink`, by following the steps in Creating Synthesis-Only Revisions on page 13.

Note: The `blinking_led_wink` revision must use the `blinking_led_wink.sv` file.

4. Create a new implementation revision, `blinking_led_pr_delta`, by following the steps in Specifying Revision Type on page 14.
5. Update the `a10_partial_reconfig/setup.tcl` file to define the new PR implementation:

```
define_pr_impl_partition -impl_rev_name blinking_led_pr_delta \
        -partition_name pr_partition \
        -source_rev_name blinking_led_wink
```

6. Compile just this new revision by running the following command:

```
quartus_sh –t a10_partial_reconfig/flow.tcl –setup_script
a10_partial_reconfig/setup.tcl –impl blinking_led_pr_delta
```

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

Related Links

- Creating a Partial Reconfiguration Design
- Partial Reconfiguration Online Training

1.5 Document Revision History

Table 5. Document Revision History

Date	Version	Changes
2016.10.31	16.1.0	• Updated flow with 16.1 PR specific GUI features: — Design Partitions Window updates — LogicLock Plus region updates — Revision and Revision Types updates • New topic added for modifying an existing persona • New topic added for including a persona in the design
2016.07.07	16.0.0	Initial release of the document