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

Updated for Intel® Quartus® Prime Design Suite: **17.1**

Contents

(intel®)

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.

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

Reference Design Requirements
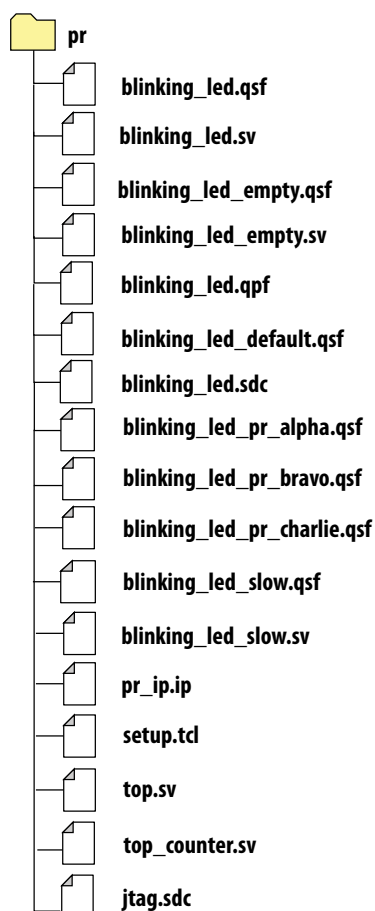
This reference design requires the following:

- Installation and basic familiarity with the Intel Quartus Prime Pro Edition version 17.1 design flow and project files for the design implementation.
- Connection with the Intel Arria 10 GX FPGA development board on the bench.

ISO
9001:2008
Registered

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

```
pr
├── blinking_led.qsf
├── blinking_led.sv
├── blinking_led_empty.qsf
├── blinking_led_empty.sv
├── blinking_led.qpf
├── blinking_led_default.qsf
├── blinking_led.sdc
├── blinking_led_pr_alpha.qsf
├── blinking_led_pr_bravo.qsf
├── blinking_led_pr_charlie.qsf
├── blinking_led_slow.qsf
├── blinking_led_slow.sv
├── pr_ip.ip
├── setup.tcl
├── top.sv
├── top_counter.sv
└── jtag.sdc
```

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

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.
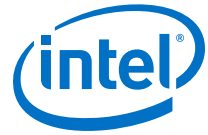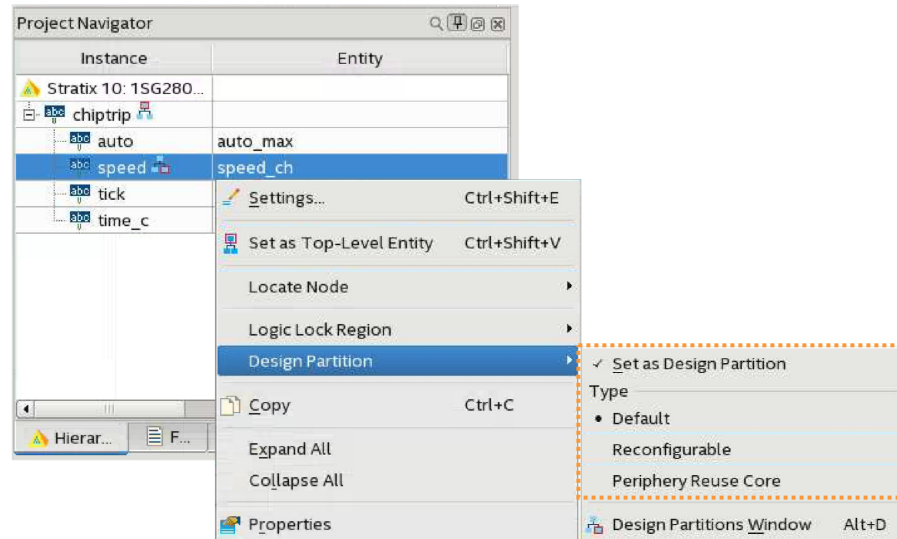
Figure 3. **Creating Design Partitions from Project Navigator**

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**

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 Links

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`**
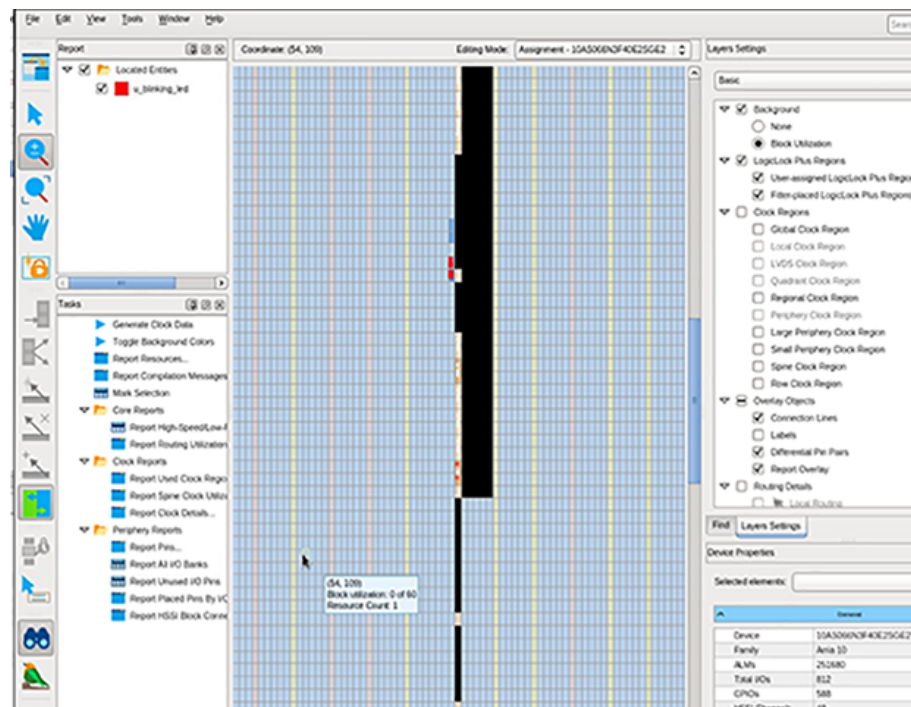
3. In the Logic Lock Regions window, specify the placement region co-ordinates in the **Origin** column. The origin corresponds to the lower-left corner of the region. For example, to set a placement region with (`X1 Y1`) co-ordinates as (`69 10`), specify the **Origin** as `X69_Y10`. The Intel Quartus 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**

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
```
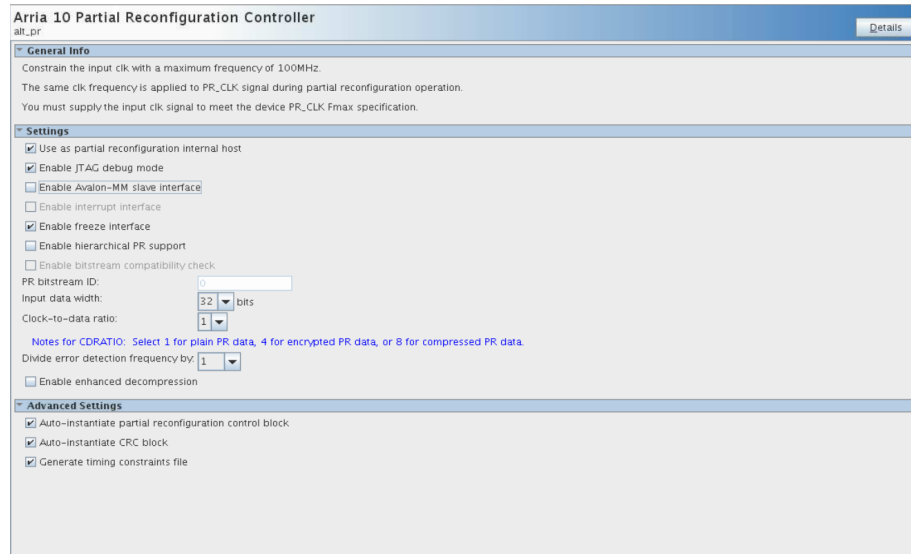
Related Links

- Floorplan the Partial Reconfiguration Design
- Incrementally Implementing Partial Reconfiguration

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**

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 Links

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
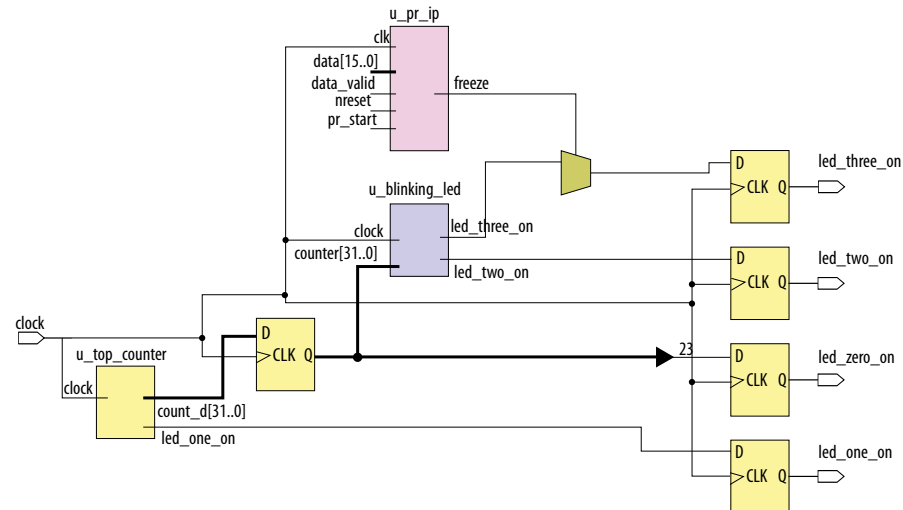
Step 5: Defining Personas

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: 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	<pre>`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</pre>
blinking_led_slow.sv	LEDs blink slower	<pre>`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</pre>
blinking_led_empty.sv	LEDs stay ON	<pre>`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</pre>

Related Links

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 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_led_slow`	`blinking_led_pr_bravo`
`blinking_led_empty`	`blinking_led_pr_charlie`

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. Disable the **Set as current revision** option and click **OK**.

5. Similarly, create `blinking_led_pr_bravo` and `blinking_led_pr_charlie` revisions, based on the `blinking_led` revision.

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

Figure 9.   Creating Revisions
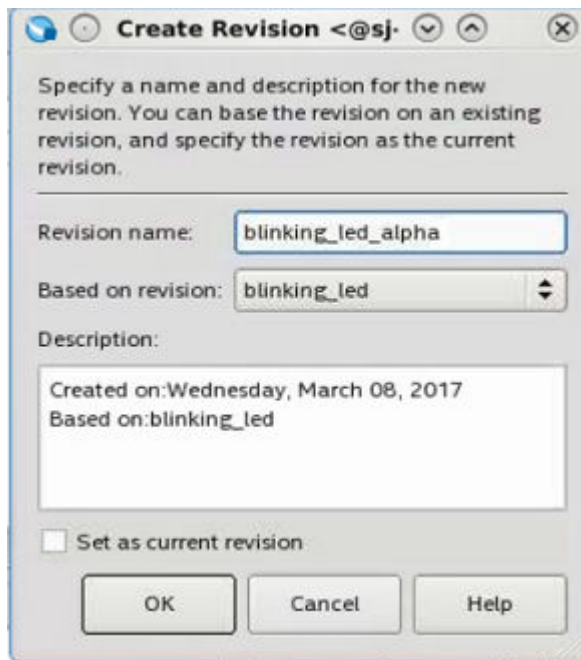
Creating Synthesis-Only Revisions

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 Intel 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_led_slow` revisions based on `blinking_led` revision. Do not set these revisions as current revision.

5. Update the `blinking_led_slow.qsf` and `blinking_led_empty.qsf` files with their corresponding TOP_LEVEL_ENTITY and SYSTEMVERILOG_FILE assignments:

```
##blinking_led_slow.qsf
set_global_assignment -name TOP_LEVEL_ENTITY blinking_led_slow
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led_slow.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, Logic Lock region assignments, or pin assignments. Remove these assignments, if any, in the `blinking_led_default.qsf`, `blinking_led_slow.qsf`, and `blinking_led_empty.qsf` files:

```
set_instance_assignment -name PARTITION pr_partition -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 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_led_slow"
PROJECT_REVISION = "blinking_led_empty"
```

Note: If you are copying the revision files from `pr` folder, manually update the `blinking_led.qpf` file with the above lines of code.

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_led_slow.qsf`	Partial Reconfiguration - Persona Synthesis
	continued...

Revision Name	Revision Type
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 each `.qsf` file 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_led_slow.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

Step 7: Generating the Partial Reconfiguration Flow Script

To generate the partial reconfiguration flow script:

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

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

   Intel 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_led_slow
   ```

   Note: All the revision projects must be in the same directory as `blinking_led.qpf`. Otherwise, update the flow script accordingly.

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. Intel 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 Intel 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

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_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 10.   Programming the Intel Arria 10 GX FPGA Development Board
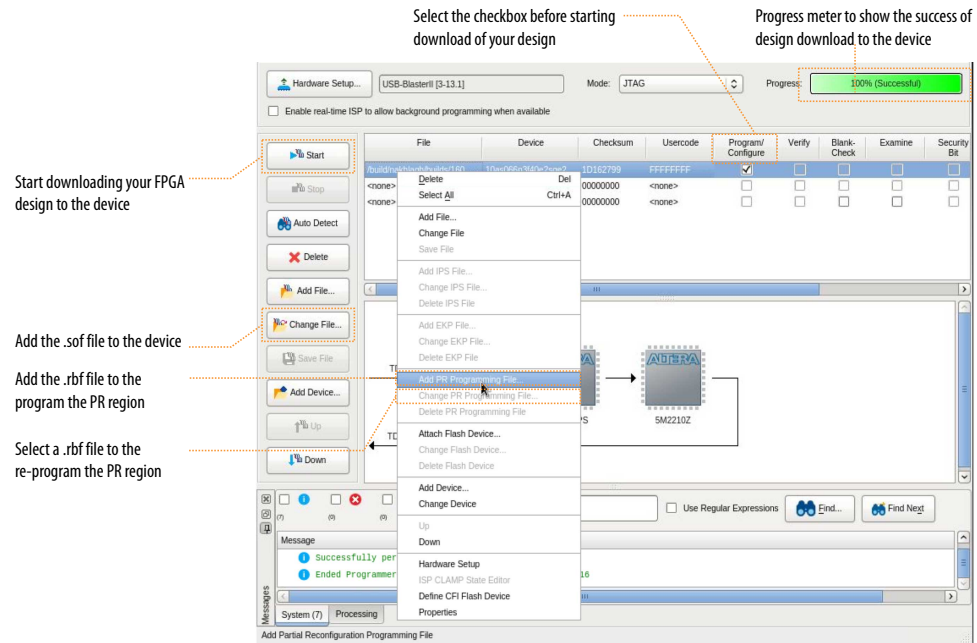
Modifying an Existing Persona

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. 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_led_slow` 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 19 to program the resulting RBF file into the FPGA.

    Note: This command does not recompile the base revision.

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 15.

   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 16.

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 Intel Arria 10 devices, refer to *Creating a Partial Reconfiguration Design* in Volume 1 of the *Intel Quartus Prime Pro Edition Handbook*.

Related Links

- Creating a Partial Reconfiguration Design
- Partial Reconfiguration Online Training

Document Revision History

Table 5.   Document Revision History

Date	Version	Changes
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 • 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