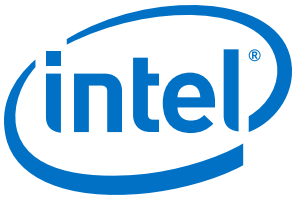


AN 825: Partially Reconfiguring a Design

on Intel® Stratix® 10 GX FPGA Development Board

Updated for Intel® Quartus® Prime Design Suite: **20.3**

Subscribe
Send Feedback

AN-825 | 2020.12.07
Latest document on the web: PDF | HTML

Contents

AN-825 | 2020.12.07

Send Feedback

Partially Reconfiguring a Design on Intel® Stratix® 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® Stratix® 10 GX FPGA development board.

The 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 Intel Quartus® Prime Pro Edition 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

Note: This tutorial uses the Intel Stratix 10 GX FPGA development board on the bench, outside of the PCIe* slot in your workstation.

Related Information

- Intel Stratix 10 GX FPGA Development Kit Web Page
- Intel Stratix 10 GX FPGA Development Kit User Guide
- Partial Reconfiguration Terminology
- Partial Reconfiguration Design Flow
- Partial Reconfiguration Design Considerations
- Partial Reconfiguration Design Guidelines

Reference Design Requirements

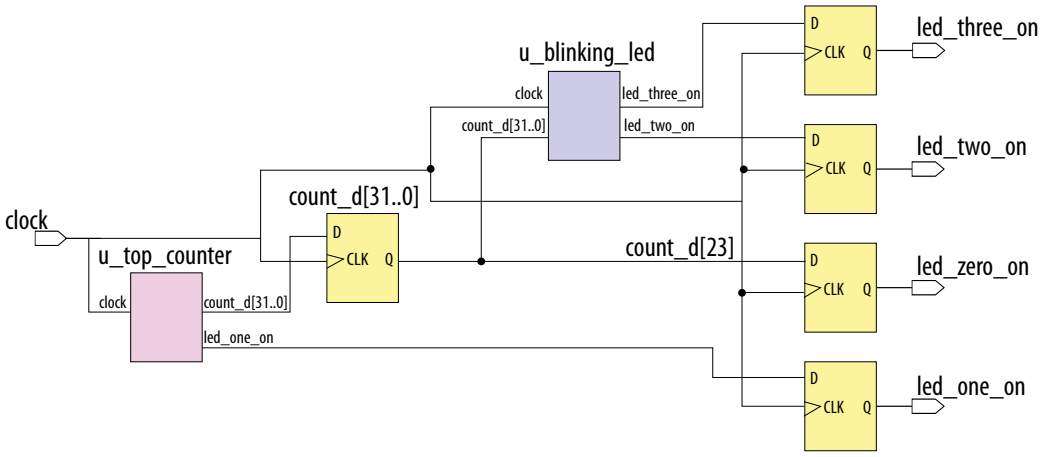
This reference design requires the following:

- Installation of the Intel Quartus Prime Pro Edition version 20.3, and understanding of the basic design flow and project files for the design implementation.
- Connection to the Intel Stratix 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 50 MHz source and connects the output to four LEDs on the board. 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:

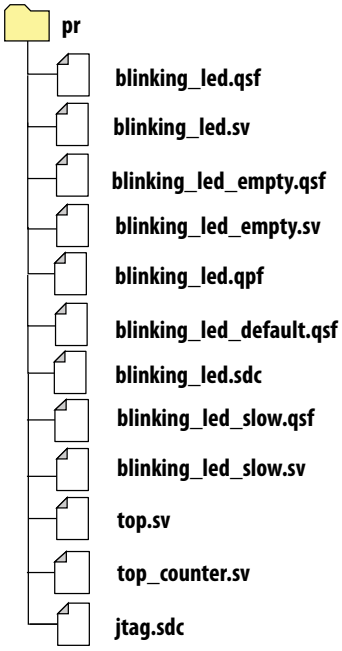
1. Click **Clone or download**.
2. Click **Download ZIP**. Unzip the `fpga-partial-reconfig-master.zip` file.
3. Navigate to the `tutorials/s10_pcie_devkit_blinking_led` sub-folder to access the reference design.

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.
`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 Stratix 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: Defining Personas on page 9
- Step 5: Creating Revisions on page 10
- Step 6: Compiling the Base Revision on page 13
- Step 7: Preparing PR Implementation Revisions on page 13
- Step 8: Programming the Board on page 15

Note: Unlike *AN 797: Partially Reconfiguring a Design on Intel Arria® 10 GX FPGA Development Board*, this tutorial does not require the addition of a Partial Reconfiguration Controller IP core. This difference is because Intel Stratix 10 devices support PR over JTAG using the hard JTAG pins of the FPGA.

Related Information

AN 797: Partially Reconfiguring a Design on 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:

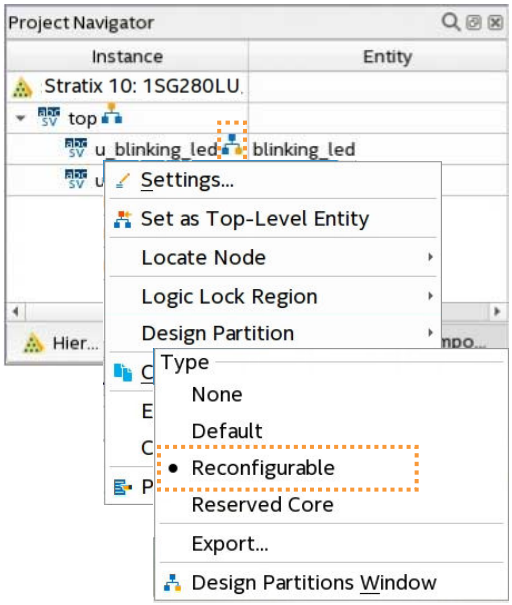
1. Create a directory in your working environment, `s10_pcie_devkit_blinking_led_pr`.
2. Copy the downloaded `tutorials/s10_pcie_devkit_blinking_led/flat` sub-folder to the directory, `s10_pcie_devkit_blinking_led_pr`.
3. In the Intel Quartus Prime Pro Edition software, click **File ➤ Open Project** and select `blinking_led.qpf`.
4. To elaborate the hierarchy of the flat design, click **Processing ➤ Start ➤ Start Analysis & Synthesis**. Alternatively, at the command-line, run the following command:

```
quartus_syn blinking_led -c blinking_led
```

Step 2: Creating a Design Partition

You must create design partitions for each PR region that you want to partially reconfigure. The following steps create a design partition for the `u_blinking_led` instance.

Figure 3. Creating Design Partitions

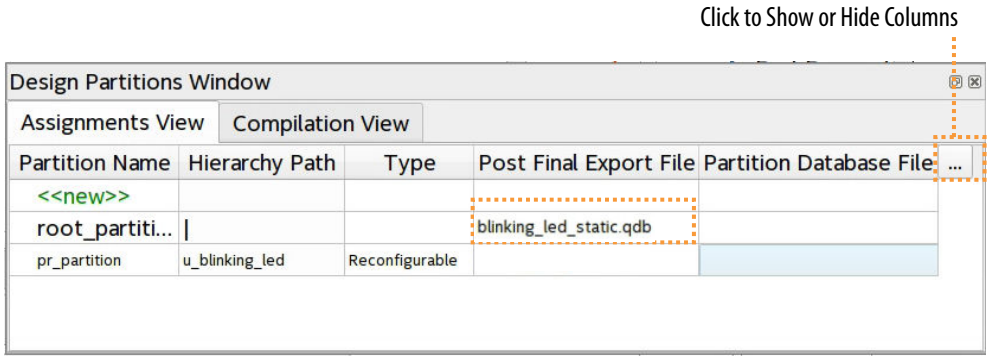


1. Right-click the `u_blinking_led` instance in the **Project Navigator** and click **Design Partition ➤ Reconfigurable**. A design partition icon appears next to each instance that is set as a partition.
2. Click **Assignments ➤ Design Partitions Window**. The window displays all design partitions in the project.
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.

4. To export the finalized static region from the base revision compile, double-click the entry for `root_partition` in the **Post Final Export File** column, and type `blinking_led_static.qdb`.

Figure 4. Exporting Post Final Snapshot in Design Partitions Window



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

set_instance_assignment -name EXPORT_PARTITION_SNAPSHOT_FINAL \
        blinking_led_static.qdb -to | -entity top
```

Related Information

Create Design Partitions

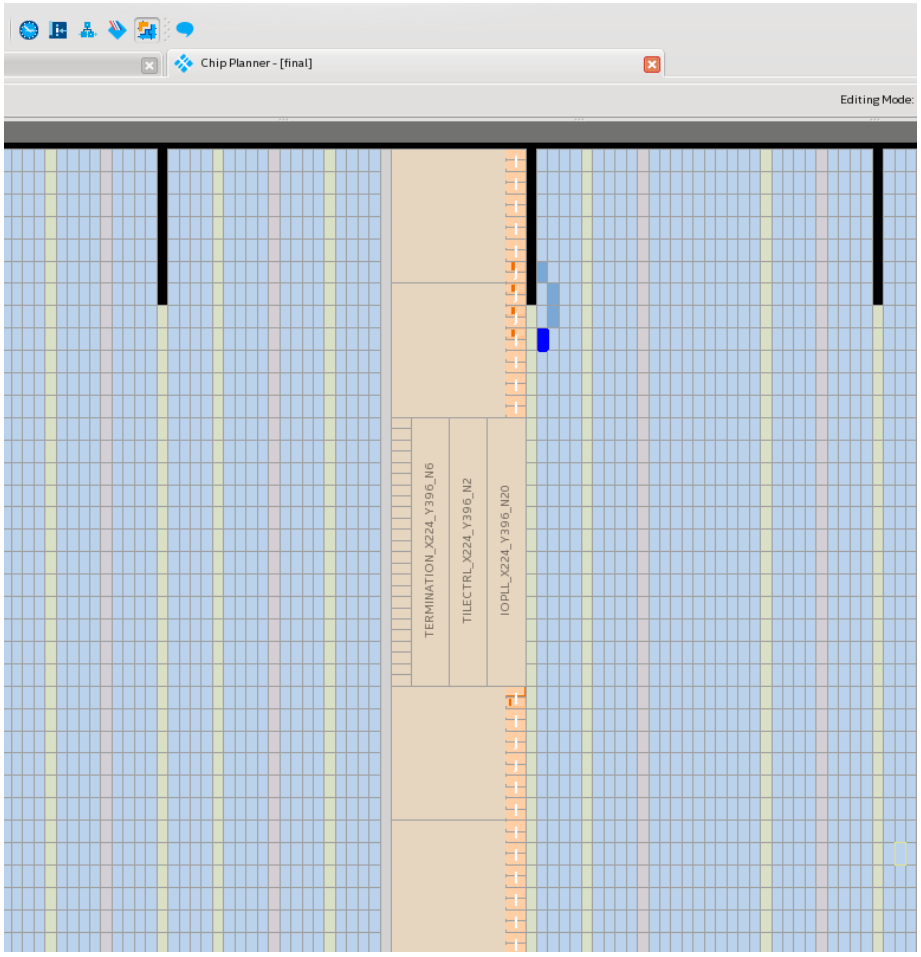
Step 3: Allocating Placement and Routing Region for a PR Partition

For every base revision you create, the PR design flow places the corresponding persona core in your PR partition 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`



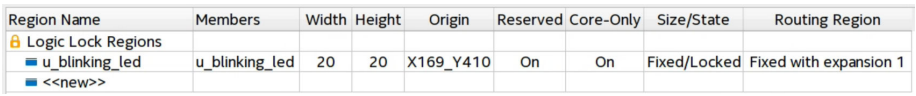
3. In the Logic Lock Regions window, specify the placement region co-ordinates in the **Origin** column. The origin corresponds to the lower-left corner of the region. For example, to set a placement region with (X1 Y1) co-ordinates as (169 410), specify the **Origin** as X169_Y410. The Intel Quartus 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 - (169 410), and a height and width of 20 for the placement region. Define any value for the placement region. Ensure that 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



Region Name	Members	Width	Height	Origin	Reserved	Core-Only	Size/State	Routing Region
Logic Lock Regions								
u_blinking_led	u_blinking_led	20	20	X169_Y410	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 "X169 Y410 X188 Y429" -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 "X168 Y409 X189 Y430" -to \
        u_blinking_led
```

Related Information

- Floorplan the Partial Reconfiguration Design
- Applying Floorplan Constraints Incrementally

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

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

continued...

File Name	Description	Code
`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

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.

Related Information

Step 2: Creating a Design Partition on page 6

Step 5: 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. The available revision types are:

- 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

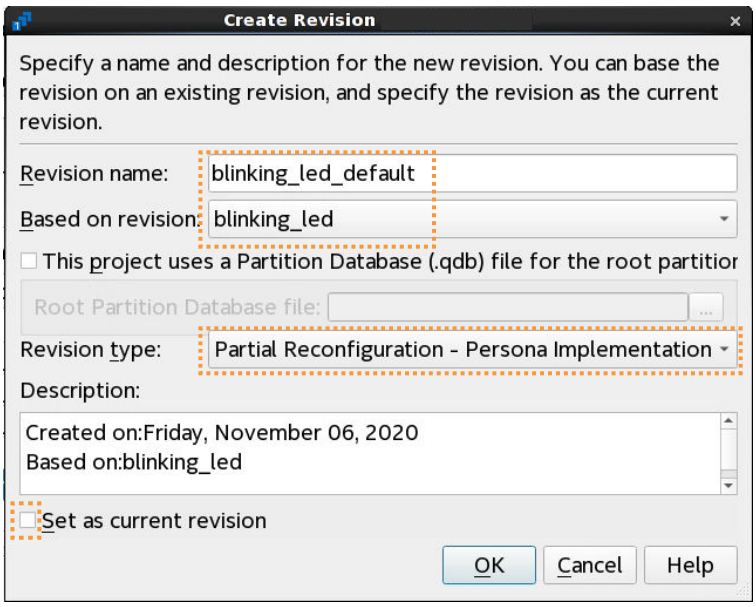
1. Click **Project ➤ Revisions**.
2. In **Revision Name**, select the **blinking_led** revision, and then click **Set Current**.
3. Click **Apply**. The `blinking_led` revision displays as the current revision.
4. To set the **Revision Type** for `blinking_led`, click **Assignments ➤ Settings ➤ General**.
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**.

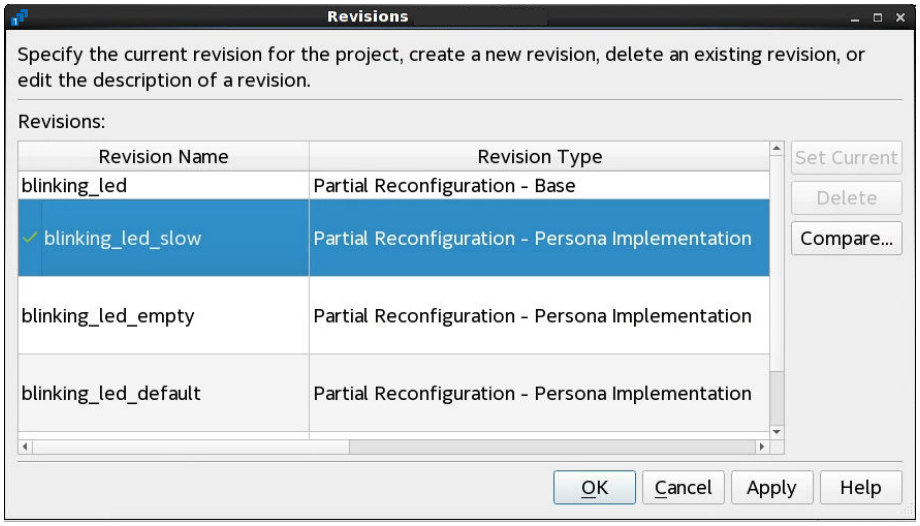
Figure 7. Creating Revisions



Note: You can add the static region `.qdb` file by turning on **This project uses a Partition Database (.qdb) file for the root partition**, and then specifying the static region `.qdb` file name.

5. Similarly, set the **Revision type** for `blinking_led_slow` and `blinking_led_empty` revisions.

Figure 8. Project Revisions



6. Verify that each `.qsf` file now contains the following assignment:

```
set_global_assignment -name REVISION_TYPE PR_IMPL
set_instance_assignment -name ENTITY_REBINDING \
       place_holder -to u_blinking_led
```

where, `place_holder` is the default entity name for the newly created PR implementation revision.

Step 6: Compiling the Base Revision

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. Inspect the bitstream files that generate in the `output_files` directory.

Table 4. Generated Files

Name	Type	Description
`blinking_led.sof`	Base programming file	Used for full-chip base configuration
`blinking_led.pr_partition.rbf`	PR bitstream file for base persona	Used for partial reconfiguration of base persona.
`blinking_led_static.qdb`	`.qdb` database file	Finalized database file used to import the static region.

Related Information

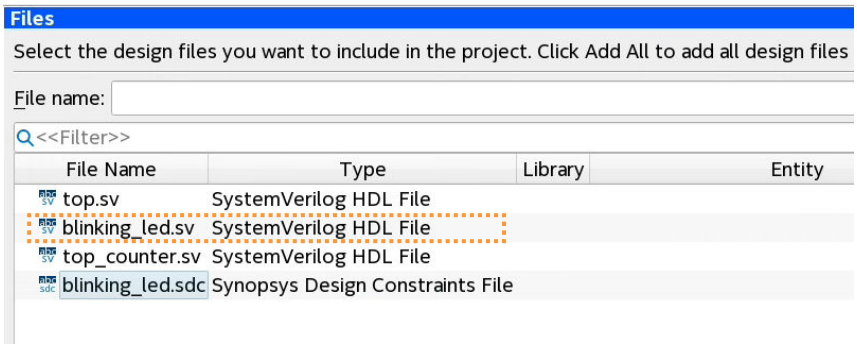
- Floorplan the Partial Reconfiguration Design
- Applying Floorplan Constraints Incrementally

Step 7: 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_led_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.

Figure 9. Files Page



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.

   Alternatively, the following command assigns this file:

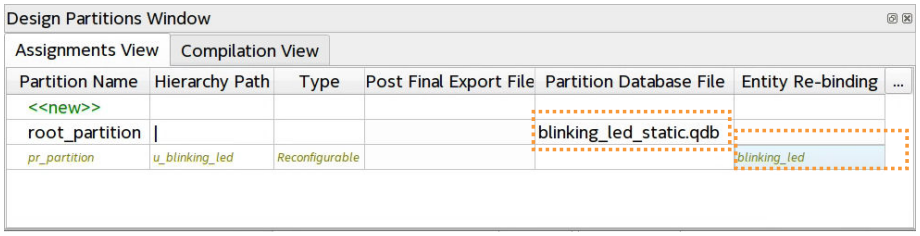
```
set_instance_assignment -name QDB_FILE_PARTITION \
    blinking_led_static.qdb -to |
```

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.

   Note: A placeholder entity rebinding assignment is added to the implementation revision automatically. However, you must change the default entity name in the assignment to an appropriate entity name for your design.

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

Figure 10. Entity Rebinding



Verify that each of the following lines now exists in the appropriate `.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. 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
```

7. Repeat the above steps to prepare `blinking_led_slow` and `blinking_led_empty` revisions:

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

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

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 8: Programming the Board

Before you begin:

1. Connect the power supply to the Intel Stratix 10 GX FPGA development board.
2. Connect the Intel FPGA Download Cable between your PC USB port and the Intel FPGA Download Cable port on the development board.

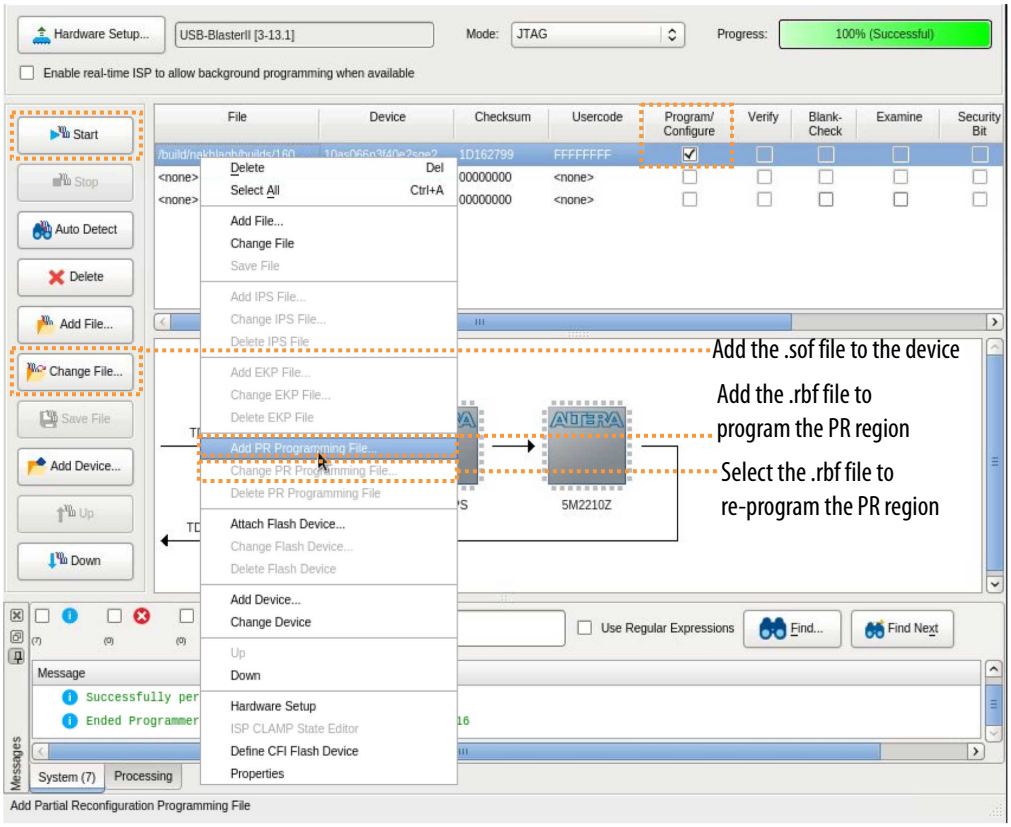
Note: This tutorial utilizes the Intel Stratix 10 GX FPGA development board on the bench, outside of the PCIe slot in your host machine.

To run the design on the Intel Stratix 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, **1SG280LU5S1**.
4. Click **OK**. The Intel Quartus Prime software detects and updates the Programmer with the three FPGA devices on the board.
5. Select the 1SG280LU5S1 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_slow.rbf` file.
11. Disable **Program/Configure** for `blinking_led_default.sof` file.

12. Enable **Program/Configure** for `blinking_led_slow.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 reprogram 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.rbf` file causes the LEDs to blink at a specific frequency, and loading the `blinking_led_empty.rbf` file causes the LEDs to stay ON.

Figure 11. Programming the Intel Stratix 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 *Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration* 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 11.

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

Related Information

- Creating a Partial Reconfiguration Design
- Partial Reconfiguration Online Training

Document Revision History for AN 825: Partially Reconfiguring a Design on Intel Stratix 10 GX FPGA Development Board

Document Version	Intel Quartus Prime Version	Changes
2020.12.07	20.3	• Updated Intel Quartus Prime software version number in "Reference Design Requirements" topic. • Updated figures in "Creating Implementation Revisions" topic. • Corrected typo in "Step 7: Preparing PR Implementation Revisions" topic.
2019.08.06	19.1	• Corrected error in "Exporting Post Final Snapshot in Design Partitions Window" figure.
		continued...

Document Version	Intel Quartus Prime Version	Changes
2019.07.15	19.1	• Changed default file export location from output_files to project directory. • Described new reserved core partition type and related GUI. • Updated Design Partition Window descriptions and screenshots for column display button and new partition properties.
2018.09.24	18.1	• Updated sections - *Step 2: Creating a Design Partition*, *Step 6: Compiling the Base Revision*, and *Step 7: Preparing PR Implementation Revisions* with the new PR flow that eliminates the need for manual export of finalized snapshot of the static region. • Other minor text edits and image updates.
2018.05.07	18.0	• Compilation flow change • Other minor text edits
2017.11.06	17.1	Initial release of the document.