AN 954: Hierarchical Partial Reconfiguration Tutorial

for the Intel® Agilex® F-Series FPGA Development Board

Updated for Intel® Quartus® Prime Design Suite: **21.1**

Subscribe

Send Feedback

AN-954 | 2021.08.04

Latest document on the web: PDF | HTML

Contents

Hierarchical Partial Reconfiguration Tutorial for the Intel® Agilex® F-Series FPGA Development Board

This application note demonstrates transformation of a simple, flat (non-partitioned) design into a hierarchical partial reconfiguration design and implementation of the design on the Intel® Agilex® F-Series 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. Hierarchical partial reconfiguration (HPR) is an extension of partial reconfiguration (PR) that allows you to contain a child PR partition within another parent PR partition. You can create multiple personas for both the child and parent partitions. You can nest the child partitions within their parent partitions. Reconfiguring a parent partition does not impact the operation in the static region, but replaces the child partitions of the parent region with default child partition personas. 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

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 Agilex F-Series FPGA development board on the bench, outside of the PCIe* slot in your workstation.

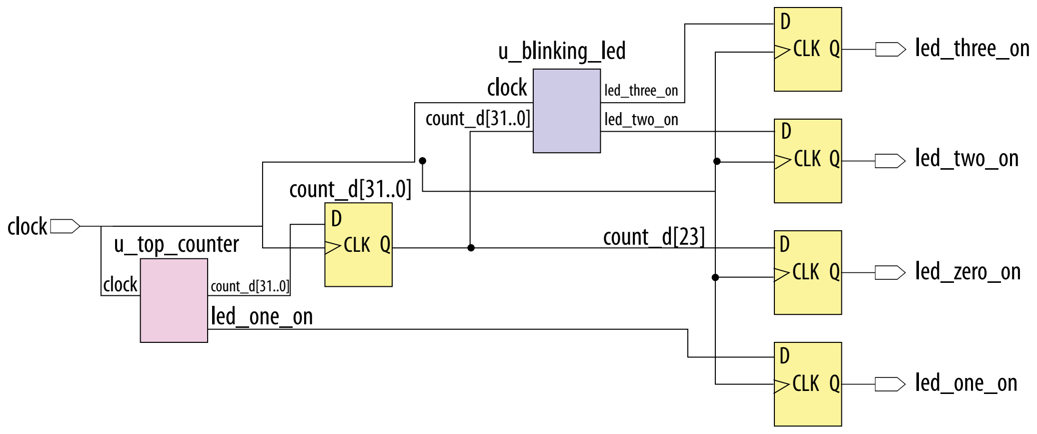
Related Information

- Intel Agilex F-Series FPGA Development Kit User Guide
- Intel Quartus Prime Pro Edition User Guide: Partial Reconfiguration

Reference Design Overview

This reference design used in this tutorial 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 connected to the FPGA. Selecting the output from particular counter bits causes the LEDs to blink at a specific frequency.

Figure 1. Flat Reference Design without PR Partitioning



Reference Design Requirements

This reference design requires the following:

- Intel Quartus Prime Pro Edition software version 21.1 for the design implementation.
- Intel Agilex F-Series FPGA development kit (DK-DEV-AGF014E3ES or DK-DEV-AGF014E2ES) for the FPGA implementation.

Reference Design Files

The files required for this tutorial are available in the following location:

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

To download the files:

1. Click **Code ➤ Download ZIP**.
2. Unzip the `fpga-partial-reconfig-master.zip` file.
3. Navigate to the `tutorials/agilex_pcie_devkit_blinking_led_hpr` sub-folder to access the reference design.

The `flat` folder consists of the following files:

Table 1. Reference Design Files

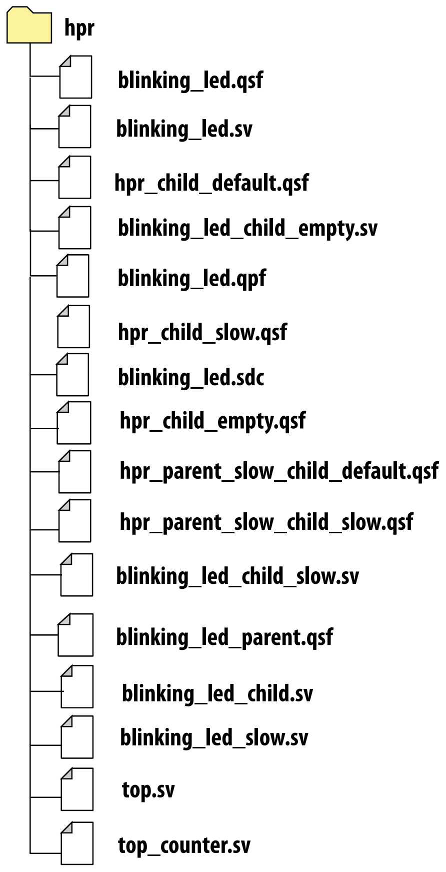
File Name	Description
`top.sv`	Top-level file containing the flat implementation of the design. This module instantiates the `blinking_led` sub-partition and the `top_counter` module.
`top_counter.sv`	Top-level 32-bit counter that controls `LED[1]` directly. The registered output of the counter controls `LED[0]`, and 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`	In this tutorial, you convert this module into a parent 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 `hpr` 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 Agilex F-Series FPGA development board:

- Step 1: Getting Started on page 6
- Step 2: Creating a Child Level Submodule on page 6
- Step 3: Creating Design Partitions on page 7
- Step 4: Allocating Placement and Routing Region for PR Partitions on page 9
- Step 5: Defining Personas on page 10
- Step 6: Creating Revisions on page 11
- Step 7: Compiling the Base Revision on page 14
- Step 8: Preparing the PR Implementation Revisions for Parent PR Partition on page 15
- Step 9: Preparing the PR Implementation Revisions for Child PR Partitions on page 17
- Step 10: Programming the Board on page 20

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, `agilex_pcie_devkit_blinking_led_hpr`.
2. Copy the downloaded `tutorials/agilex_pcie_devkit_blinking_led_hpr/flat` sub-folder to the directory, `agilex_pcie_devkit_blinking_led_hpr`.
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 Child Level Submodule

To convert the flat design into a hierarchical PR design, you create a child submodule (`blinking_led_child.sv`) within the parent submodule (`blinking_led.sv`).

1. Create a new design file, `blinking_led_child.sv`, and add the following lines of code to this file:

```
`timescale 1 ps / 1 ps
`default_nettype none

module blinking_led_child (

   // clock
   input wire clock,
   input wire [31:0] counter,
   // Control signals for the LEDs
   output wire led_three_on

);
   localparam COUNTER_TAP = 23;
   reg led_three_on_r;

   assign led_three_on   = led_three_on_r;

   always_ff @(posedge clock) begin
      led_three_on_r   <= counter[COUNTER_TAP];
   end

endmodule
```

2. Modify `blinking_led.sv` to connect `led_two_on` to bit 23 of the counter from the static region, and instantiate the `blinking_led_child` module. `blinking_led.sv` must appear as follows:

```
`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;
   assign  led_two_on    = led_two_on_r;

   // The counter:
   always_ff @(posedge clock) begin
         led_two_on_r <= counter[COUNTER_TAP];
   end

   blinking_led_child u_blinking_led_child (
         .led_three_on           (led_three_on),
         .counter                (counter),
         .clock                  (clock)
   );

endmodule
```

3. Save all files, retaining the **Add file to current project** option.
4. Click **Processing ➤ Start ➤ Start Analysis & Synthesis**

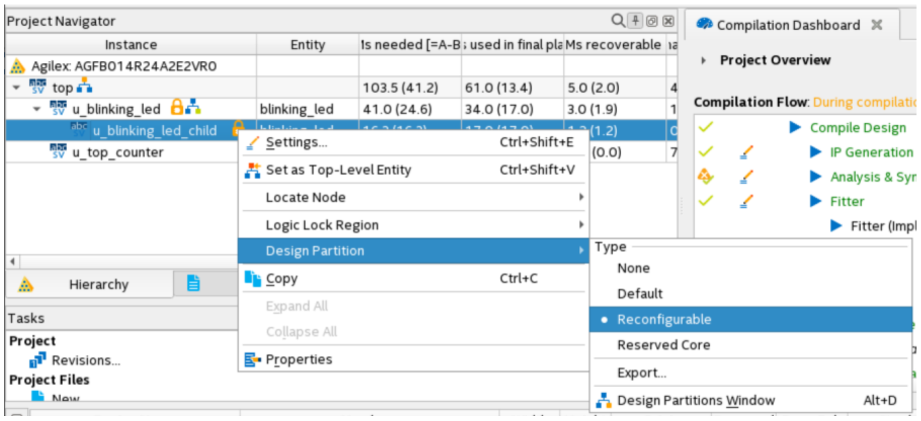
Step 3: Creating Design Partitions

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 two design partitions for the `u_blinking_led_child` and `u_blinking_led` instances.

To create design partitions for hierarchical partial reconfiguration:

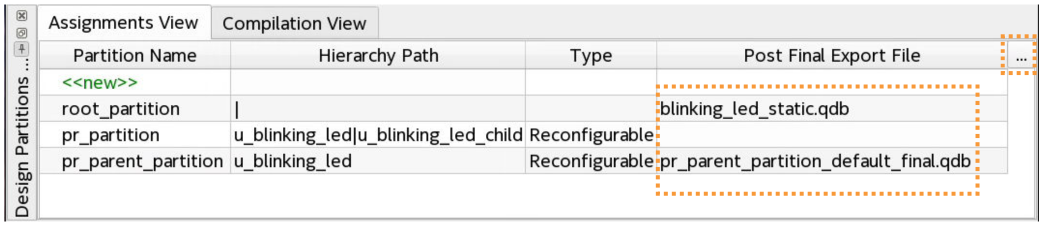
1. Right-click the `u_blinking_led_child` instance in the **Project Navigator** and click **Design Partition ➤ Reconfigurable**.

Figure 3. Create Design Partition



2. Click **Assignments ➤ Design Partitions Window**. The window displays all design partitions in the project.
3. Edit the default partition name in the Design Partitions Window by double-clicking the name. Rename the partition to `pr_partition`.
4. Repeat steps 1 and 2 to assign reconfigurable design partitions to the `u_blinking_led` instance. Rename this partition to `pr_parent_partition`.
5. In the Design Partitions Window, click (**...**) at the farthest right column header and enable the **Post Final Export File** column.
6. 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`. This file is for the subsequent PR implementation compiles.
7. To export the finalized parent PR partition from the base revision compile, double-click the entry for `pr_parent_partition` in **Post Final Export File**, and type `pr_parent_partition_default_final.qdb`. This file is for the subsequent PR implementation compiles.

Figure 4. Design Partitions Window



Partition Name	Hierarchy Path	Type	Post Final Export File
<<new>>			
root_partition	|		blinking_led_static.qdb
pr_partition	u_blinking_led|u_blinking_led_child	Reconfigurable	
pr_parent_partition	u_blinking_led	Reconfigurable	pr_parent_partition_default_final.qdb

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

```
set_instance_assignment -name PARTITION pr_partition -to \
        u_blinking_led|u_blinking_led_child
set_instance_assignment -name PARTIAL_RECONFIGURATION_PARTITION ON -to \
        u_blinking_led|u_blinking_led_child
set_instance_assignment -name PARTITION pr_parent_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
set_instance_assignment -name EXPORT_PARTITION_SNAPSHOT_FINAL \
        pr_parent_partition_default_final.qdb -to u_blinking_led -entity top
```

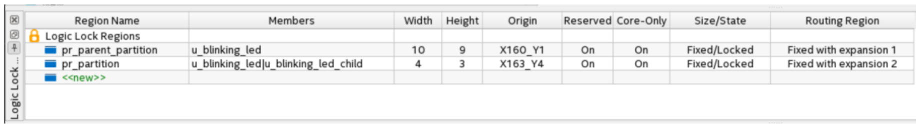
Step 4: Allocating Placement and Routing Region for PR Partitions

When you create the base revision, 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_child` 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. Specify a region **Width** of 4 and **Height** of 3.
3. Specify the placement region coordinates for `blinking_led_child` in the **Origin** column. The origin corresponds to the lower-left corner of the region. Specify the **Origin** as X163_Y4. The Compiler calculates (X166_Y6) as the top-right coordinate.

   Note: This tutorial uses the (`X1 Y1`) co-ordinates - (`163 4`), a height of 3 and a width of 4 for the placement region. Define any value for the placement region. Ensure that the region covers the `blinking_led_child` 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 2. The routing region must be larger than the placement region to accommodate the routing of different personas.
7. Repeat steps 1-6 for the `u_blinking_led` instance. The parent-level placement region must fully enclose the corresponding child-level placement and routing regions, while allowing sufficient space for the parent-level logic placement. Specify a **Height** of `9`, and **Width** of `10`. Specify the **Origin** as `X160 Y1`. The Compiler calculates the top-right coordinate.

Figure 5. Logic Lock Regions Window



Region Name	Members	Width	Height	Origin	Reserved	Core-Only	Size/State	Routing Region
Logic Lock Regions								
pr_parent_partition	u_blinking_led	10	9	X160_Y1	On	On	Fixed/Locked	Fixed with expansion 1
pr_partition	u_blinking_led|u_blinking_led_child	4	3	X163_Y4	On	On	Fixed/Locked	Fixed with expansion 2
<<new>>								

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

```
set_instance_assignment -name PLACE_REGION "X163 Y4 X166 Y6" -to \
     u_blinking_led|u_blinking_led_child
set_instance_assignment -name RESERVE_PLACE_REGION ON \
     -to u_blinking_led|u_blinking_led_child
set_instance_assignment -name CORE_ONLY_PLACE_REGION ON -to \
     u_blinking_led|u_blinking_led_child
set_instance_assignment -name REGION_NAME u_blinking_led_child -to \
     u_blinking_led|u_blinking_led_child
set_instance_assignment -name ROUTE_REGION "X161 Y2 X168 Y8" -to \
     u_blinking_led|u_blinking_led_child
set_instance_assignment -name RESERVE_ROUTE_REGION OFF -to \
     u_blinking_led|u_blinking_led_child
set_instance_assignment -name PLACE_REGION "X160 Y1 X169 Y9" \
     -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 REGION_NAME u_blinking_led -to \
     u_blinking_led
set_instance_assignment -name ROUTE_REGION "X159 Y0 X170 Y10" -to \
     u_blinking_led
```

Related Information

- Floorplan the Partial Reconfiguration Design
- Applying Floorplan Constraints Incrementally

Step 5: Defining Personas

This reference design defines five separate personas for the parent and child PR partitions. To define and include the personas in your project:

1. Create four SystemVerilog files, `blinking_led_child.sv`, `blinking_led_child_slow.sv`, `blinking_led_child_empty.sv`, and `blinking_led_slow.sv` in your working directory for the five 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_child.sv	Default persona for the child-level design	
		```
		\`timescale 1 ps / 1 ps
		\`default_nettype none

		module blinking_led_child (

		   // clock
		   input wire clock,
		   input wire [31:0] counter,

		   // Control signals for the LEDs
		   output wire led_three_on

		);
		   localparam COUNTER_TAP = 23;
		   reg led_three_on_r;

		   assign led_three_on   = led_three_on_r;

		   always_ff @(posedge clock) begin
		      led_three_on_r   <= counter[COUNTER_TAP];
		   end

		endmodule
		```

blinking_led_child_slow.sv	The LED_THREE blinks slower	
		```
		\`timescale 1 ps / 1 ps
		\`default_nettype none

		module blinking_led_child_slow (

		   // clock
		   input wire clock,
		   input wire [31:0] counter,

		   // Control signals for the LEDs
		   output wire led_three_on
		);

		   localparam COUNTER_TAP = 27;
		   reg led_three_on_r;
		```

continued...

File Name	Description	Code
		```
		   assign led_three_on = led_three_on_r;

		   always_ff @(posedge clock) begin
		      led_three_on_r   <= counter[COUNTER_TAP];
		   end

		endmodule
		```

blinking_led_child_empty.sv	The LED_THREE stays ON	
		```
		`timescale 1 ps / 1 ps
		`default_nettype none

		module blinking_led_child_empty (

		   // clock
		   input wire clock,
		   input wire [31:0] counter,

		   // Control signals for the LEDs
		   output wire led_three_on

		);

		   // LED is active low
		   assign  led_three_on  = 1'b0;

		endmodule
		```

blinking_led_slow.sv	The LED_TWO blinks 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;
		   assign  led_two_on    = led_two_on_r;

		   // The counter:
		   always_ff @(posedge clock) begin
		         led_two_on_r <= counter[COUNTER_TAP];
		   end


		   blinking_led_child u_blinking_led_child(
		         .led_three_on           (led_three_on),
		         .counter             (counter),
		         .clock                  (clock)
		   );

		endmodule
		```

Related Information

Step 3: Creating Design Partitions on page 7

Step 6: Creating Revisions

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

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

To compile a PR design, you must create a PR implementation revision for each persona. In addition, you must specify the **Partial Reconfiguration - Base** or **Partial Reconfiguration - Persona Implementation** revision type for each of the revisions.

The following table lists the revision name and type for each revision you create:

Table 3. Revision Names and Types

Revision Name	Revision Type
`blinking_led.qsf`	Partial Reconfiguration - Base
`hpr_child_default.qsf`	Partial Reconfiguration - Persona Implementation
`hpr_child_slow.qsf`	Partial Reconfiguration - Persona Implementation
`hpr_child_empty.qsf`	Partial Reconfiguration - Persona Implementation
`hpr_parent_slow_child_default.qsf`	Partial Reconfiguration - Persona Implementation
`hpr_parent_slow_child_slow.qsf`	Partial Reconfiguration - Persona Implementation

Table 4. Parent and Child Persona Revisions

Revision Name	Parent Persona Behavior	Child Persona Behavior
`hpr_child_default.qsf`	Fast Blinking	Fast Blinking
`hpr_child_slow.qsf`	Fast Blinking	Slow Blinking
`hpr_child_empty.qsf`	Fast Blinking	No Blinking (Always ON)
`hpr_parent_slow_child_default.qsf`	Slow Blinking	Fast Blinking
`hpr_parent_slow_child_slow.qsf`	Slow Blinking	Slow Blinking

Setting the Base Revision Type

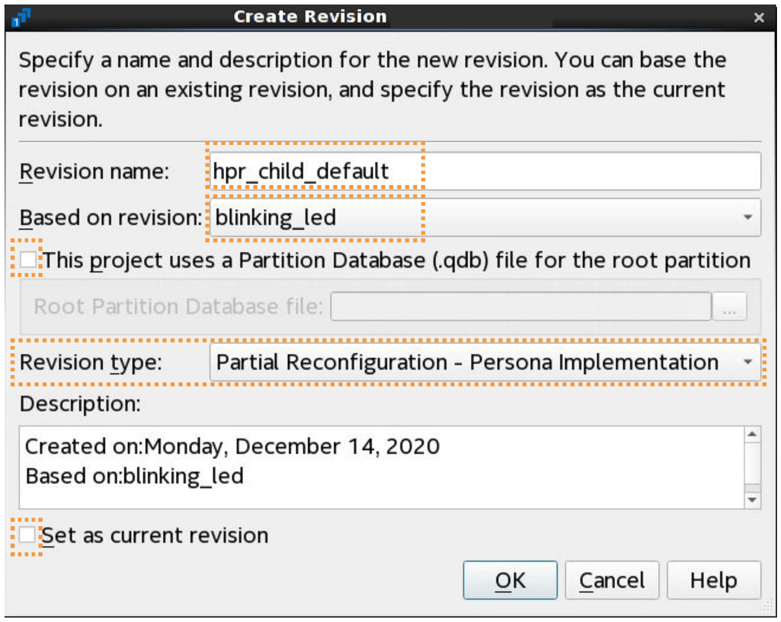
1. Click **Project ➤ Revisions**.
2. In **Revision Name**, select the **blinking_led** revision.
3. For **Revision Type**, select **Partial Reconfiguration - Base**, and then click **OK**.
4. Verify that the `blinking_led.qsf` now contains the following assignment:

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

Creating Implementation Revisions

1. Double-click **<<new revision>>**.
2. For **Revision name**, specify `hpr_child_default` and select **blinking_led** for **Based on revision**.
3. Select **Partial Reconfiguration - Persona Implementation** for **Revision type**.

Figure 6. Creating Revisions


Create Revision

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: hpr_child_default

Based on revision: blinking_led

☐ This project uses a Partition Database (.qdb) file for the root partition

Root Partition Database file:

Revision type: Partial Reconfiguration - Persona Implementation

Description:

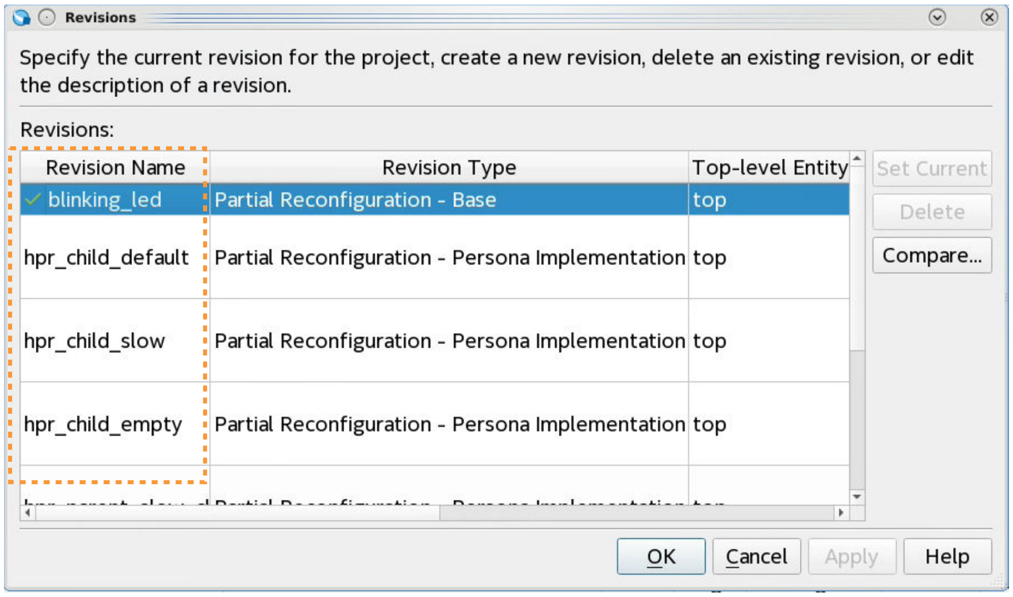
Created on:Monday, December 14, 2020
Based on:blinking_led

☐ Set as current revision

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

4. Turn off **Set as current revision**.
5. Repeat steps 1 through 4 to create these implementation revisions:
   - `hpr_child_slow`
   - `hpr_child_empty`
   - `hpr_parent_slow_child_default`
   - `hpr_parent_slow_child_slow`

Figure 7. New Implementation Revisions



Specify the current revision for the project, create a new revision, delete an existing revision, or edit the description of a revision.

Revisions:

Revision Name	Revision Type	Top-level Entity
✓ blinking_led	Partial Reconfiguration - Base	top
hpr_child_default	Partial Reconfiguration - Persona Implementation	top
hpr_child_slow	Partial Reconfiguration - Persona Implementation	top
hpr_child_empty	Partial Reconfiguration - Persona Implementation	top
hpr_parent_slow_c	Partial Reconfiguration - Persona Implementation	top

6. Verify that the `.qsf` file for each revision 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 7: 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 generated to the `output_files` directory:

Table 5. Generated Files

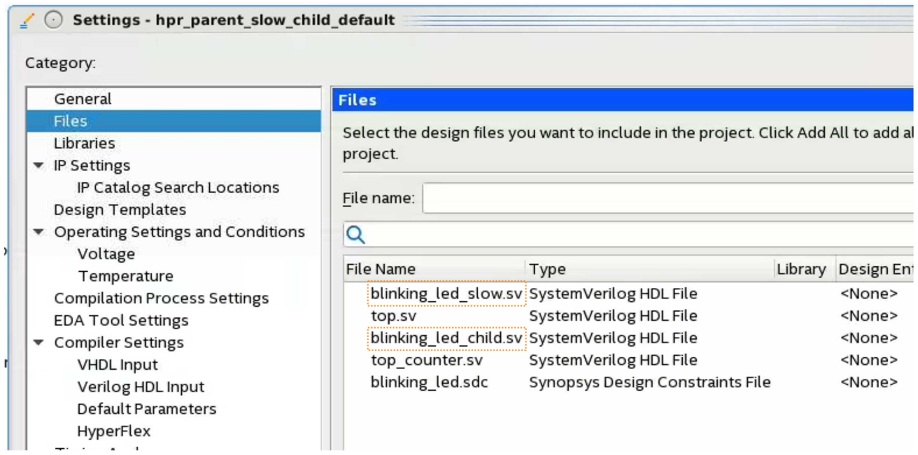
Name	Type	Description
`blinking_led.sof`	Base programming file	For programming the FPGA with the static logic, along with the default personas for the parent and child PR regions.
`blinking_led.pr_parent_partition.rbf`	PR bitstream file for parent PR partition	For programming the default persona of the parent PR region.
`blinking_led.pr_parent_partition.pr_partition.rbf`	PR bitstream file for child PR partition	For programming the default persona of the child PR region.
`blinking_led_static.qdb`	`.qdb` database file	Finalized database file for importing the static region.
`pr_parent_partition_default_final.qdb`	`.qdb` database file	Finalized database file for importing the default parent PR partition.

Step 8: Preparing the PR Implementation Revisions for Parent PR Partition

You must prepare the parent and child PR implementation revisions before you can generate the PR bitstream for device programming. This setup includes mapping the new PR logic to the preexisting parent PR partition.

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

Figure 8. Add/Remove Files in Project



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

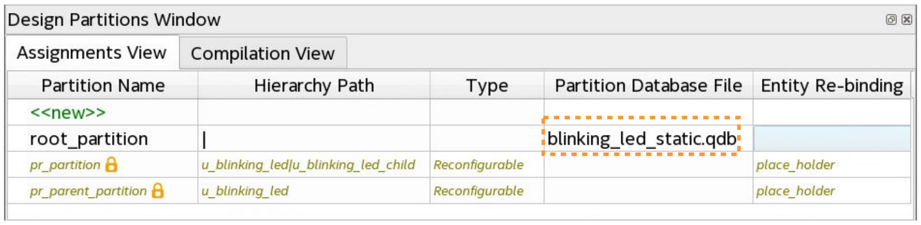
File Name	Type	Library	Design En
blinking_led_slow.sv	SystemVerilog HDL File		<None>
top.sv	SystemVerilog HDL File		<None>
blinking_led_child.sv	SystemVerilog HDL File		<None>
top_counter.sv	SystemVerilog HDL File		<None>
blinking_led.sdc	Synopsys Design Constraints File		<None>

3. Use **Add/Remove Files in Project** to add the appropriate parent and child persona source files for each implementation revision. Remove any source files that don't apply to the implementation.

Implementation Revision Name	Parent Persona Source File	Child Persona Source File
`hpr_parent_slow_child_default`	`blinking_led_slow.sv`	`blinking_led_child.sv`

4. To specify the `.qdb` file associated with the root partition, click **Assignments ➤ Design Partitions Window**. Specify the `.qdb` file associated with the static region by double-clicking the **Partition Database File** cell and navigating to the `blinking_led_static.qdb` file.

Figure 9. Specify Partition Database File



Partition Name	Hierarchy Path	Type	Partition Database File	Entity Re-binding
<<new>>				
root_partition	|		blinking_led_static.qdb	
pr_partition	u_blinking_led|u_blinking_led_child	Reconfigurable		place_holder
pr_parent_partition	u_blinking_led	Reconfigurable		place_holder

Alternatively, the following command assigns this file:

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

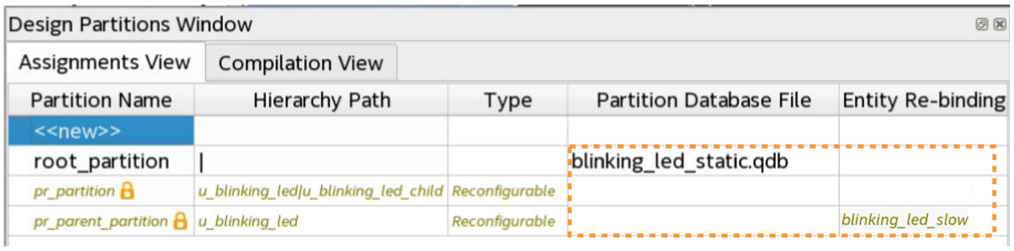
5. In the **Entity Re-binding** cell, specify the entity name for the PR parent partition. For this implementation revision, the entity name is `blinking_led_slow`. `blinking_led_slow` is the name of the entity that you are partially reconfiguring. `u_blinking_led` is the name of the instance that your entity overwrites during PR.

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

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

Note: Because the child PR logic is already defined by the parent PR partition, whose entity name is rebound, do not use an entity rebinding assignment for the child PR partition.

Figure 10. Entity Re-binding



Partition Name	Hierarchy Path	Type	Partition Database File	Entity Re-binding
<<new>>				
root_partition	|		blinking_led_static.qdb	
pr_partition	u_blinking_led|u_blinking_led_child	Reconfigurable		
pr_parent_partition	u_blinking_led	Reconfigurable		blinking_led_slow

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

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

8. To export this new parent PR partition as a finalized `.qdb` file, click **Project ➤ Export Design Partition**. Specify the following options for the partition:

Table 6. Export Design Partition Options

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

Alternatively, the following command exports the parent PR region:

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

9. Inspect the files generated to the `output_files` directory:

Table 7. Generated Files

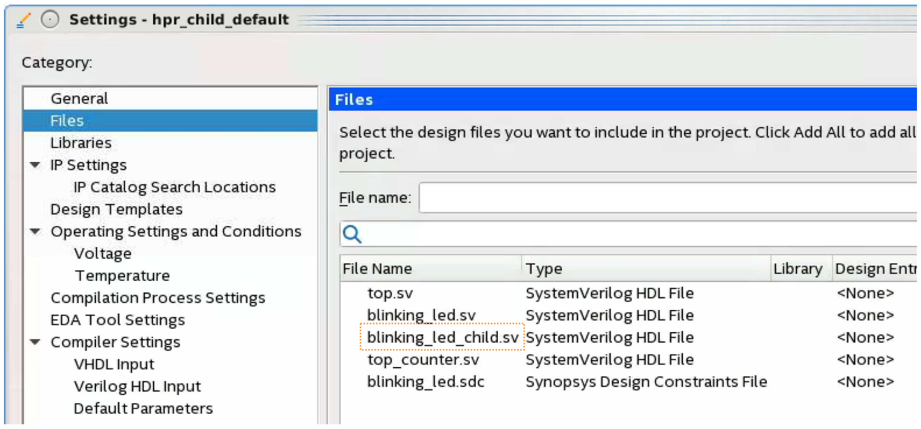
Name	Type	Description
`hpr_parent_slow_child_default.pr_parent_partition.rbf`	PR bitstream file for parent PR partition	For programing the default persona for the parent PR region. Causes the `led_two_on` to blink at a lower rate.
`hpr_parent_slow_child_default.pr_parent_partition.pr_partition.rbf`	PR bitstream file for child PR partition	For programming the default persona for the child PR region. Causes the `led_three_on` to blink at the default rate.
`pr_parent_partition_slow_final.qdb`	Finalized `.qdb` database file	For import of the slow parent PR partition.

Step 9: Preparing the PR Implementation Revisions for Child PR Partitions

This setup includes adding the static region `.qdb` file as the source file for each implementation revision. In addition, you must import the parent PR partition `.qdb` file and specify the corresponding entity of the PR region.

1. To set the current revision, click **Project ➤ Revisions**, select **hpr_child_default** as the **Revision name**, and then click **Set Current**.
2. To specify the correct child persona source file for each implementation revision, click **Project ➤ Add/Remove Files in Project**. Verify that `blinking_led_child.sv` appears in the file list.

Figure 11. Add/Remove Files in Project


Settings - hpr_child_default

Category:

- General
- Files
- Libraries
- IP Settings
  - IP Catalog Search Locations
- Design Templates
- Operating Settings and Conditions
  - Voltage
  - Temperature
- Compilation Process Settings
- EDA Tool Settings
- Compiler Settings
  - VHDL Input
  - Verilog HDL Input
  - Default Parameters

Files

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

File name:

File Name	Type	Library	Design Entr
top.sv	SystemVerilog HDL File		<None>
blinking_led.sv	SystemVerilog HDL File		<None>
blinking_led_child.sv	SystemVerilog HDL File		<None>
top_counter.sv	SystemVerilog HDL File		<None>
blinking_led.sdc	Synopsys Design Constraints File		<None>

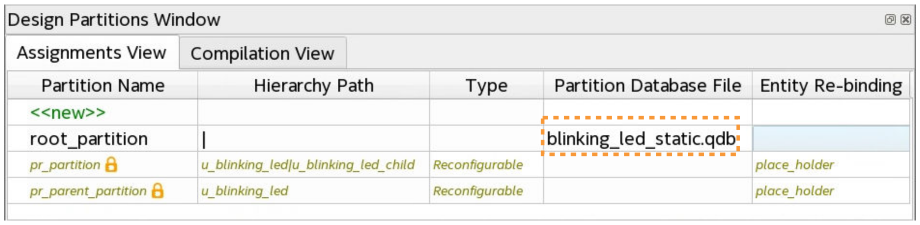
3. Repeat steps 1 through 2 to specify the following child persona source files for the other implementation revisions:

Table 8. Implementation Revision Source Files

Implementation Revision Name	Child Persona Source File
`hpr_child_default`	`blinking_led_child.sv`
`hpr_child_slow`	`blinking_led_child_slow.sv`
`hpr_child_empty`	`blinking_led_child_empty.sv`
`hpr_parent_slow_child_slow`	`blinking_led_child_slow.sv`

4. To verify the `.qdb` file associated with the root partition, click **Assignments ➤ Design Partitions Window**. Specify the `.qdb` file associated with the static region by double-clicking the **Partition Database File** cell and navigating to the `blinking_led_static.qdb` file.

Figure 12. Specify Partition Database File



Design Partitions Window

Assignments View | Compilation View

Partition Name	Hierarchy Path	Type	Partition Database File	Entity Re-binding
<<new>>				
root_partition	|		blinking_led_static.qdb	
pr_partition	*u_blinking_led|u_blinking_led_child*	*Reconfigurable*		*place_holder*
pr_parent_partition	*u_blinking_led*	*Reconfigurable*		*place_holder*

Alternatively, the following command assigns this file:

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

5. To specify the parent PR partition `.qdb` file, click **Assignments ➤ Design Partitions Window**. Double-click the **Partition Database File** for the `pr_parent_partition` and specify the respective `.qdb` file in the project directory.

Table 9. Specify the Parent PR Partition .qdb File

Implementation Revision Name	Parent Persona .qdb File
`hpr_child_default`	`pr_parent_partition_default_final.qdb`
`hpr_child_slow`	`pr_parent_partition_default_final.qdb`
`hpr_child_empty`	`pr_parent_partition_default_final.qdb`
`hpr_parent_slow_child_slow`	`pr_parent_partition_slow_final.qdb`

verify the following line exists in the `.qsf`:

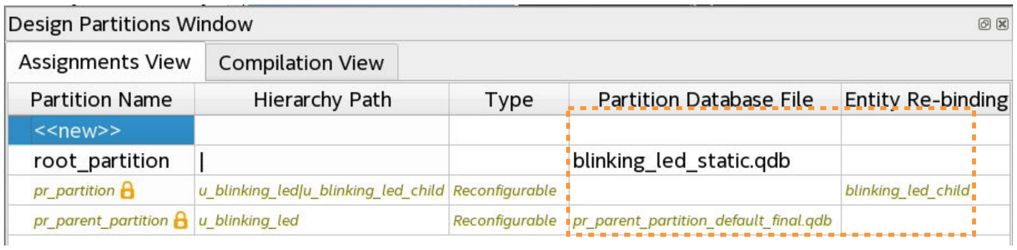
```
# To use the default parent PR persona:
set_instance_assignment -name QDB_FILE_PARTITION \
      pr_parent_partition_default_final.qdb -to u_blinking_led

# To use the slow parent PR persona:
set_instance_assignment -name QDB_FILE_PARTITION \
      pr_parent_partition_slow_final.qdb -to u_blinking_led
```

6. In the **Entity Re-binding** cell, specify the entity name of the child PR partition. For the default persona, the entity name is `blinking_led`. For this implementation revision, `blinking_led_child` is the name of the entity that

Send Feedback

you are partially reconfiguring. `u_blinking_led|u_blinking_led_child` is the name of the instance that your entity overwrites during PR. Verify that the following line now exists in the `.qsf`:

Figure 13. Entity Rebinding Assignment



Design Partitions Window

Assignments View | Compilation View

Partition Name	Hierarchy Path	Type	Partition Database File	Entity Re-binding
<<new>>				
root_partition	|		blinking_led_static.qdb	
pr_partition	u_blinking_led|u_blinking_led_child	Reconfigurable		blinking_led_child
pr_parent_partition	u_blinking_led	Reconfigurable	pr_parent_partition_default_final.qdb	

```
#hpr_child_default.qsf
set_instance_assignment -name ENTITY_REBINDING \
      blinking_led_child -to u_blinking_led|u_blinking_led_child

#hpr_child_slow.qsf and hpr_parent_slow_child_slow.qsf
set_instance_assignment -name ENTITY_REBINDING \
      blinking_led_child_slow -to u_blinking_led|u_blinking_led_child

#hpr_child_empty.qsf
set_instance_assignment -name ENTITY_REBINDING \
      blinking_led_child_empty -to u_blinking_led|u_blinking_led_child
```

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

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

8. Repeat the above steps to prepare `hpr_child_slow`, `hpr_child_empty`, and `hpr_parent_slow_child_slow` revisions.

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

9. Inspect the bitstream files generated to the `output_files` directory.

   Note: Since you imported the parent PR partition as a finalized `.qdb` file, and used entity-rebinding only on the child PR region, the software generates the PR bitstream only for the child PR partition.

   Verify that the `output_files` directory contains the following generated `.rbf` files after compiling all the implementation revisions:

   - `hpr_child_default.pr_parent_partition.pr_partition.rbf`
   - `hpr_child_slow.pr_parent_partition.pr_partition.rbf`
   - `hpr_child_empty.pr_parent_partition.pr_partition.rbf`
   - `hpr_parent_slow_child_slow.pr_parent_partition.pr_partition.rbf`

Step 10: Programming the Board

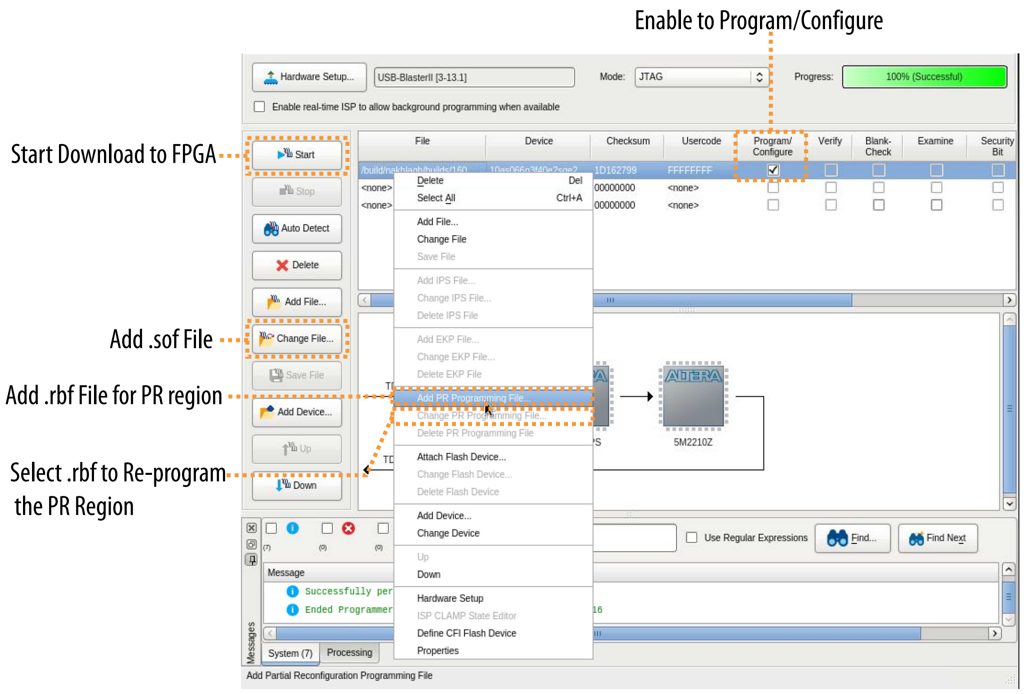
This tutorial uses an Intel Agilex F-Series FPGA development board on the bench, outside of the PCIe slot in your host machine. Before you program the board, ensure that you have completed the following steps:

1. Connect the power supply to the Intel Agilex F-Series 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.

To run the design on the Intel Agilex F-Series 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 appropriate device for your development kit.
4. Click **OK**. The Intel Quartus Prime software detects and updates the Programmer with the three FPGA chips on the board.
5. Select the Intel Agilex device, click **Change File** and load the `blinking_led.sof` file.
6. Enable **Program/Configure** for `blinking_led.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 child PR region, right-click the `blinking_led.sof` file in the Programmer and click **Add PR Programming File**.
10. Select the `hpr_child_slow.pr_parent_partition.pr_partition.rbf` file.
11. Disable **Program/Configure** for `blinking_led.sof` file.
12. Enable **Program/Configure** for `hpr_child_slow.pr_parent_partition.pr_partition.rbf` file and click **Start**. On the board, observe `LED[0]` and `LED[1]` continuing to blink. When the progress bar reaches 100%, `LED[2]` blinks at the same rate, and `LED[3]` blinks slower.
13. To program both the parent and child PR region, right-click the `.rbf` file in the Programmer and click **Change PR Programing File**.
14. Select the `hpr_parent_slow_child_slow.pr_parent_partition.rbf` file.
15. Click **Start**. On the board, observe that `LED[0]` and `LED[1]` continuing to blink. When the progress bar reaches 100%, both `LED[2]` and `LED[3]` blink slower.
16. Repeat the above steps to dynamically re-program just the child PR region, or both the parent and child PR regions simultaneously.

Figure 14. Programming the Intel Agilex F-Series FPGA Development Board



If you face any PR programming errors, refer to the *Avoiding PR Programming Errors* section in the Partial Reconfiguration User Guide.

Related Information

Avoiding PR Programming Errors

Programming the Child PR Region

The current version of the Intel Quartus Prime Pro Edition software does not provide a mechanism to check for incompatible child PR bitstreams for Intel Agilex devices. So, it is very important that you program the correct child persona to match the parent persona. Programming an incompatible bitstream on an Intel Agilex device can result in one of the following:

- Successful PR programming, but corrupted FPGA functionality
- Unsuccessful PR programming, and corrupted FPGA functionality

If you wish to reprogram a child PR region on the FPGA, you must ensure that the child PR `.rbf` is generated from an implementation revision compile whose parent PR persona matches the persona currently on the FPGA. For example, when you program the base `blinking_led.sof` onto the FPGA, the parent PR persona is `default`. The child PR persona is `default` as well. To change the child PR persona to `slow` persona, you have the choice of using the following bitstreams:

1. `hpr_child_slow.pr_parent_partition.pr_partition.rbf`
2. `hpr_parent_slow_child_slow.pr_parent_partition.pr_partition.rbf`

In this case, you must choose the `hpr_child_slow.pr_parent_partition.pr_partition.rbf` bitstream, because the `hpr_child_slow.pr_parent_partition.pr_partition.rbf` is generated by an implementation revision that has the `default` parent persona. Choosing `hpr_parent_slow_child_slow.pr_parent_partition.pr_partition.rbf` results in unsuccessful PR programming, corrupted FPGA functionality, or both.

Modifying an Existing Persona

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

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

1. In the `blinking_led_child_slow.sv` file, modify the `COUNTER_TAP` parameter from 27 to 28.
2. Recompile any implementation revision that uses this source file, such as `hpr_child_slow` or `hpr_parent_slow_child_slow`.
3. Regenerate the PR bitstreams from the `.pmsf` files.
4. Follow the steps in Step 10: Programming the Board on page 20 to program the resulting RBF file into the FPGA.

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 causes `led_two` (parent) to blink at a slower rate, while keeping `led_three` (child) on:

1. Create an implementation revision, `hpr_parent_slow_child_empty`, by following the steps in Creating Implementation Revisions on page 12.
2. Compile the revision by clicking **Processing ➤ Start Compilation**.

Related Information

- Creating a Partial Reconfiguration Design
- Partial Reconfiguration Online Training

Document Revision History for AN 954: Hierarchical Partial Reconfiguration Tutorial for Intel Agilex F-Series FPGA Development Board

Document Version	Intel Quartus Prime Version	Changes
2021.08.04	21.1	Initial release.