



# intel AN 805 Hierarchical Partial Reconfiguration of a Design on Arria 10 SoC Development Board User Guide

[Home](#) » [Intel](#) » intel AN 805 Hierarchical Partial Reconfiguration of a Design on Arria 10 SoC Development Board User Guide 

## Contents

- [1 intel AN 805 Hierarchical Partial Reconfiguration of a Design on Arria 10 SoC Development Board](#)
- [2 Hierarchical Partial Reconfiguration Tutorial for Intel® Arria® 10 SoC Development Board](#)
- [3 Reference Design Overview](#)
  - [3.1 Reference Design Walkthrough](#)
  - [3.2 Related Information](#)
  - [3.3 Related Information](#)
  - [3.4 Updating the Top-Level Design](#)
  - [3.5 Related Information](#)
  - [3.6 Creating Synthesis-Only Revisions](#)
- [4 Specifying Revision Type](#)
  - [4.1 Related Information](#)
- [5 Related Information](#)
- [6 Modifying an Existing Persona](#)
- [7 Adding a New Persona to the Design](#)
- [8 Document Revision History](#)
- [9 Documents / Resources](#)
  - [9.1 References](#)
- [10 Related Posts](#)



**intel AN 805 Hierarchical Partial Reconfiguration of a Design on Arria 10 SoC Development Board**



## Hierarchical Partial Reconfiguration Tutorial for Intel® Arria® 10 SoC Development Board

This application note demonstrates transforming a simple design into a hierarchically partially reconfigurable design, and implementing the design on the Intel® Arria® 10 SoC development board. Hierarchical partial reconfiguration (HPR) is a special type of partial reconfiguration (PR), where you contain a PR region within another PR region. You can create multiple personas for both the child and parent partitions. You nest the child partitions within their parent partitions. Reconfiguring a child partition does not impact operation in the parent or static regions. 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 downtime
- 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.

### Related Information

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



The flat folder consists of the following files:

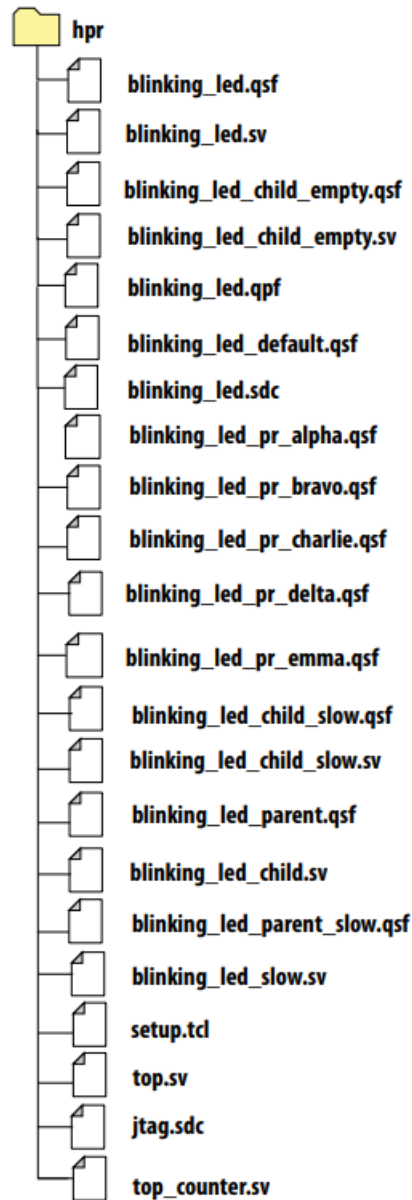
**Table 1. Reference Design Files**

File Name	Description
top. SV	The top-level file contains 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.
<b><i>continued...</i></b>	

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 the 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 Arria 10 SoC development board:

- **Step 1:** Getting Started on page 6
- **Step 2:** Create a Child Level Sub-module 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:** Adding the Intel Arria 10 Partial Reconfiguration Controller IP Core on page 10
- **Step 6:** Defining Personas on page 13
- **Step 7:** Creating Revisions on page 15
- **Step 8:** Generating the Hierarchical Partial Reconfiguration Flow Script on page 20
- **Step 9:** Running the Hierarchical Partial Reconfiguration Flow Script on page 21
- **Step 10:** Programming the Board on page 22

## Step 1: Getting Started

To copy the reference design files to your working environment and compile the blinking\_led flat design:

- Create a directory in your working environment, a10\_soc\_devkit\_blinking\_led\_hpr.
- Copy the downloaded tutorials/a10\_soc\_devkit\_blinking\_led\_hpr/flat sub-folder to the directory, a10\_soc\_devkit\_blinking\_led\_hpr.
- In the Intel Quartus Prime Pro Edition software, click File ► Open Project and select blinking\_led.qpf.
- To compile the flat design, click Processing ► Start Compilation.

## Step 2: Creating a Child Level Sub-module

To convert this flat design into a hierarchical PR design, you must create a child sub-module (blinking\_led\_child.SV) that is nested within the parent sub-module (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

```
`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 the blinking\_led.sv file to connect the led\_two\_on to bit 23 of the counter from the static region, and instantiate the blinking\_led\_child module. After modifications, your blinking\_led.sv file 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. On modifying all the design files, recompile the project by clicking Processing ► Start Compilation

### Step 3: Creating Design Partitions

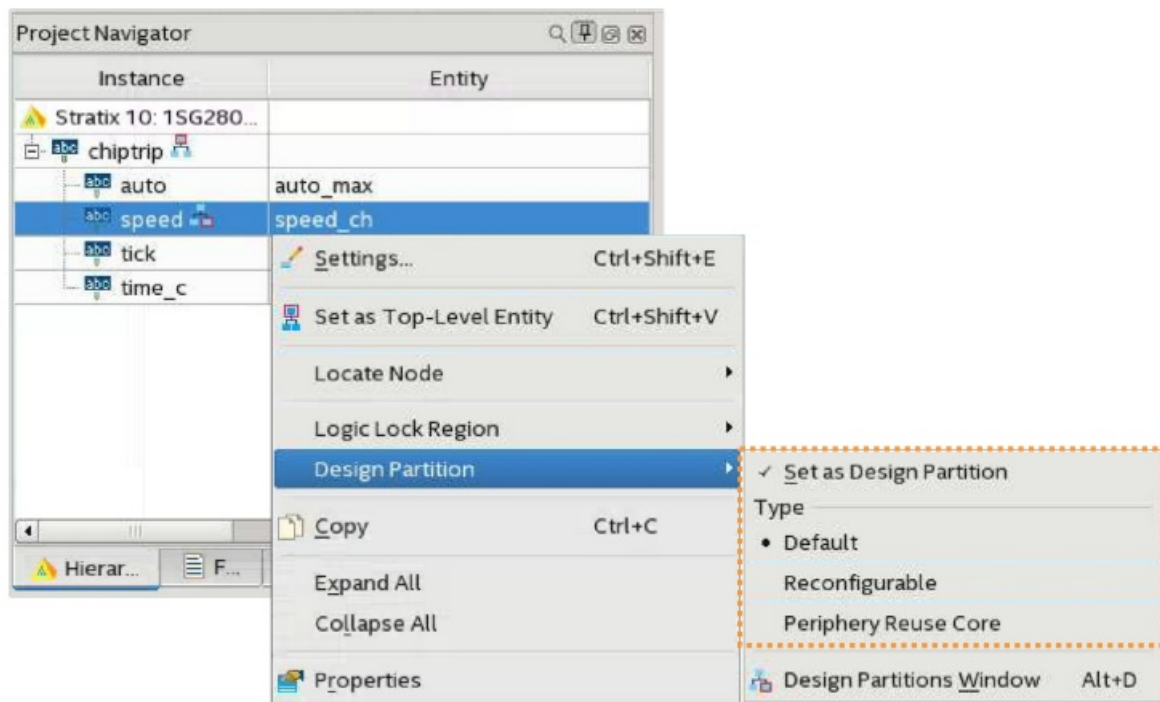
You must create design partitions for each PR region that you want to partially reconfigure. You can create any number of independent partitions or PR regions in your design. This tutorial creates two design partitions for the `u_blinking_led_child` and `u_blinking_led` instances.

**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 ► 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**





1. To define the partition Type, right-click the `u_blinking_led_child` 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**

Assignments View		Compilation View				
Partition Name	Hierarchy Path	Type	Preservation Level	Empty	Partition Database File	Color
<<new>>						
pr_partition	u_blinking_led u_blinking_led_child	Reconfigurable		No		
pr_parent_partition	u_blinking_led	Reconfigurable		No		

1. 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.

2. Repeat steps 1 and 2 to assign reconfigurable design partitions to the `u_blinking_led` instance. Rename this partition to `pr_parent_partition`.

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
```

## Related Information

### Create Design Partitions for Partial Reconfiguration

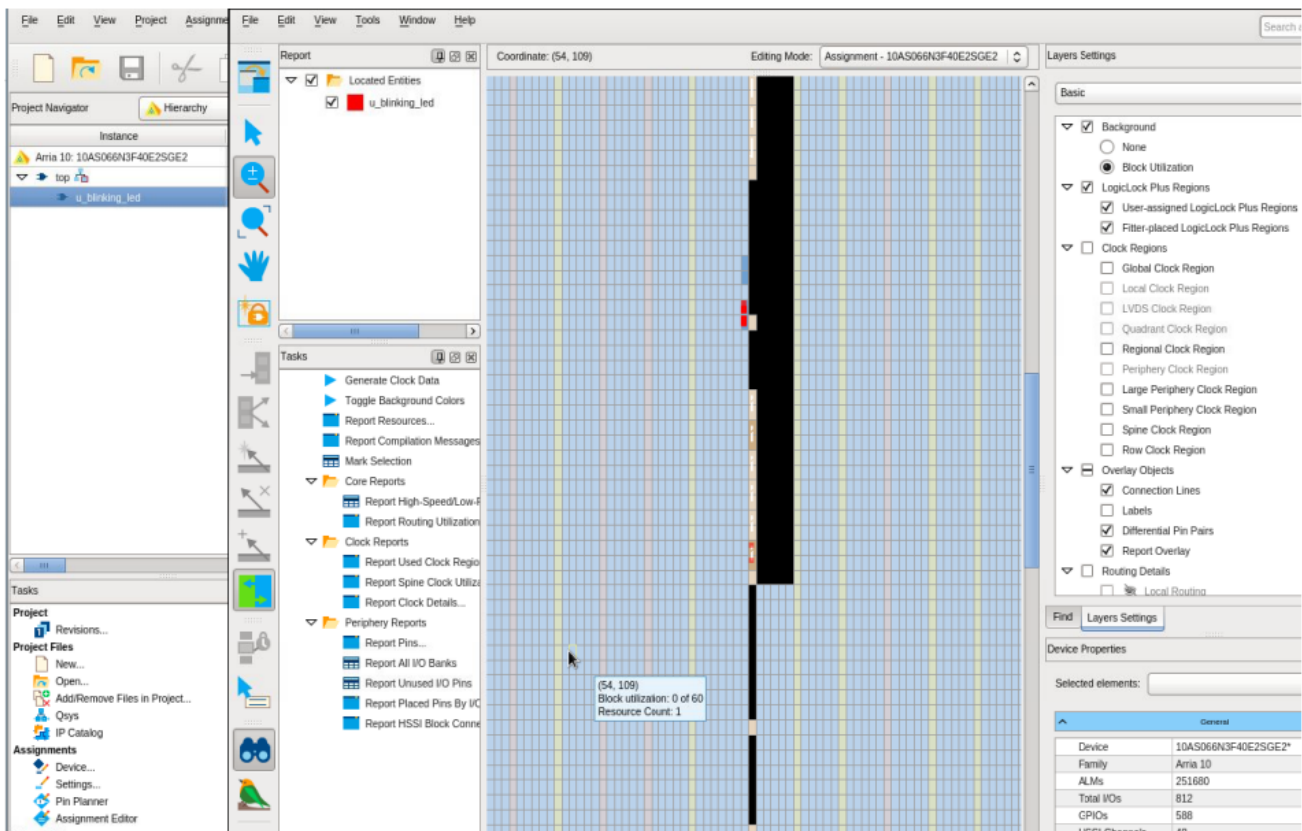


## Step 4: Allocating Placement and Routing Region for PR Partitions

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. Your placement region must enclose the `blinking_led_child` logic. Select the placement region by locating the node in Chip Planner. Right-click the `u_blinking_led_child` region name in the Project Navigator and click Locate Node ► Locate in Chip Planner.

**Figure 5. Chip Planner Node Location for `blinking_led`**



3. In the Logic Lock Regions window, specify the placement region coordinates 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) coordinates as (69 10), specify the Origin as X69\_Y10. The Intel Quartus Prime software automatically calculates the (X2 Y2) coordinates (top-right) for the placement region, based on the height and width you specify.

**Note:** This tutorial uses the (X1 Y1) coordinates – (69 10), and a height and width of 20 for the placement region. Define any value for the placement region, provided 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 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.

7. Repeat steps 1 -6 for the `u_blinking_led` instance. The parent-level placement 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. This tutorial uses the (X1 Y1) coordinates – (66 7), a height of 47, and a width of 26 for the placement region of the `u_blinking_led` instance.

**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	17	20	X172_Y410	On	On	Fixed/Locked	Fixed with expansion 1
u_blinking_led u_blinking_led_child	u_blinking_led u_blinking_led_child	13	6	X174_Y415	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 "69 10 88 29" -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 ROUTE_REGION "68 9 89 30" -to \
    u_blinking_led|u_blinking_led_child

set_instance_assignment -name PLACE_REGION "66 7 112 32" -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 "65 6 113 33" -to u_blinking_led
```

## Related Information

- Floorplan the Partial Reconfiguration Design
- Incrementally Implementing Partial Reconfiguration

## Step 5: Adding the Intel Arria 10 Partial Reconfiguration Controller IP Core

- Use the Intel Arria 10 Partial Reconfiguration Controller IP core to reconfigure 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.
  2. To launch the IP Parameter Editor Pro window, select the Intel Arria 10 Partial Reconfiguration Controller IP core from the IP library, and click Add.
  3. In the New IP Variant dialog box, type `pr_ip` as the file name and 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**

**Arria 10 Partial Reconfiguration Controller**  
alt\_pr Details

**General Info**

Constrain the input clk with a maximum frequency of 100MHz.  
The same clk frequency is applied to PR\_CLK signal during partial reconfiguration operation.  
You must supply the input clk signal to meet the device PR\_CLK Fmax specification.

**Settings**

☒ Use as partial reconfiguration internal host  
☒ Enable JTAG debug mode  
☐ Enable Avalon-MM slave interface  
☐ Enable interrupt interface  
☒ Enable freeze interface  
☐ Enable hierarchical PR support  
☐ Enable bitstream compatibility check

PR bitstream ID:   
Input data width:  bits  
Clock-to-data ratio:

Notes for CDRATIO: Select 1 for plain PR data, 4 for encrypted PR data, or 8 for compressed PR data.

Divide error detection frequency by:   
☐ Enable enhanced decompression

**Advanced Settings**

☒ Auto-instantiate partial reconfiguration control block  
☒ Auto-instantiate CRC block  
☒ Generate timing constraints file

1. Click Finish, and exit the parameter editor without generating the system. Intel Quartus Prime software creates the pr\_ip.ip IP variation file, and adds the file to the blinking\_led project.

#### Note:

1. If you are copying the pr\_ip.ip file from the hpr folder, manually edit the blinking\_led.qsf file to include the following line: set\_global\_assignment -name IP\_FILE pr\_ip.ip
2. Place the IP\_FILE assignment after the SDC\_FILE assignments (jtag.sdc and blinking\_led.sdc) in your blinking\_led.qsf file. This ordering ensures appropriate constraining of the Partial Reconfiguration IP core.

**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 used to create these clocks in the QSF file for your Intel Quartus Prime project revision. For more information, refer to Timing Constraints section in the Partial Reconfiguration IP Core User Guide.

#### Related Information

- Partial Reconfiguration IP Solutions User Guide
  - For information on the Partial Reconfiguration Region Controller IP core.
- Partial Reconfiguration IP Core User Guide
  - For information on the timing constraints.

#### 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 at the 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. However, to observe the LED continue blinking from the parent PR partition while PR programming the child partition, the freeze control signal does not turn off the led\_two\_on. Ensure that the pr\_led\_two\_on is directly

```

assign led_two_on_w = pr_led_two_on;
assign led_three_on_w = freeze ? 1'b1 : pr_led_three_on;

```

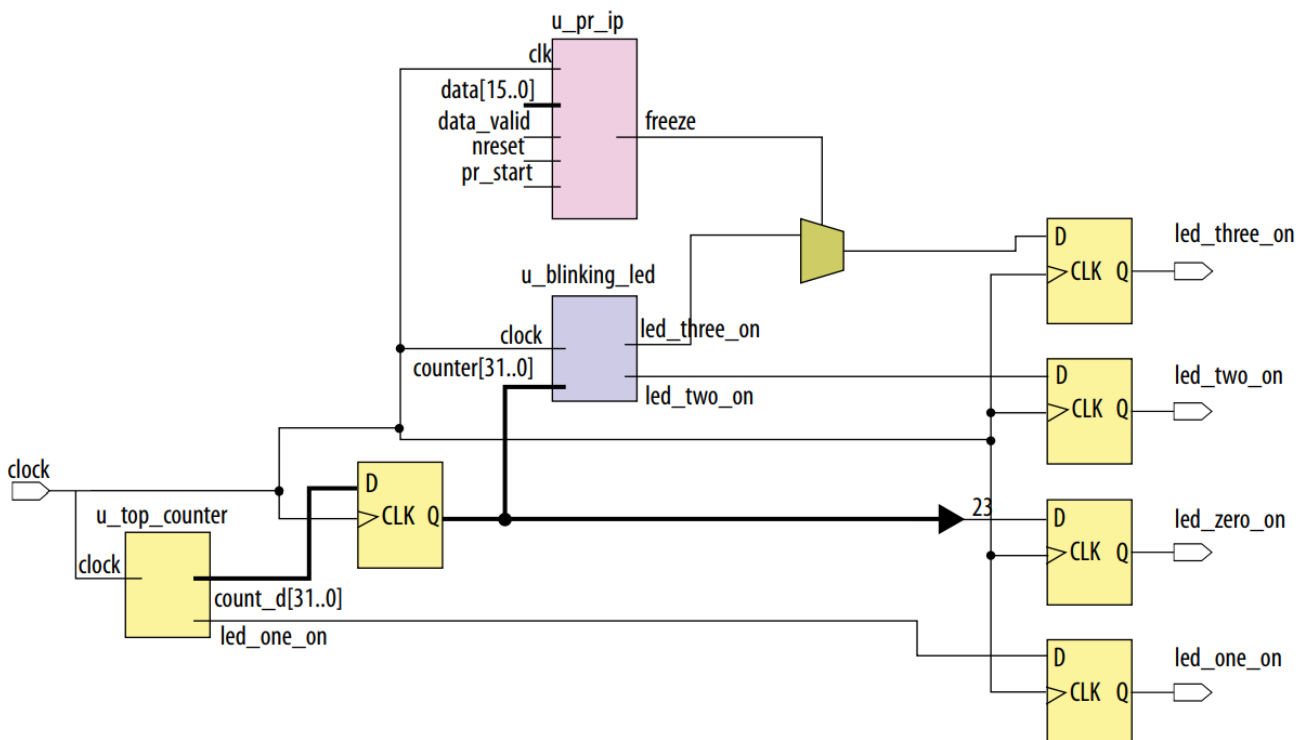
3. To assign an instance of the default parent persona (blinking\_led), update the top.SV file with the following block of code:

```

blinking_led u_blinking_led
(
    .clock            (clock),
    .counter          (count_d),
    .led_two_on       (pr_led_two_on),
    .led_three_on     (pr_led_three_on)
);

```

**Figure 8. Partial Reconfiguration IP Core Integration**



## Step 6: 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	<pre> `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 @(p osedge clock) begin led_three_on_r  &lt;= counter[COUNTER_TAP]; end  endmodule </pre>
<b>continued...</b>		

File Name	Description	Code
blinking_led_child_slow.sv	The LED_THREE blinks slower	<pre>`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 );</pre>



```
localparam COUNTER_TAP = 27; 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_empty.sv	<p>The LED_THREE stays ON</p>	<pre> `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 </pre>

blinking\_led\_slow.sv

The LED\_TW  
O  
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)
```

--	--	--

File Name	Description	Code
		);ndmodule

#### Related Information

#### Step 3: Creating Design Partitions on page 7

#### Step 7: 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, in addition to the base revision (blinking\_led), the three child-level personas and the two parent-level personas contain five separate synthesis revisions and five separate implementation revisions:

**Table 3. Revisions for the Two-Parent Personas and Three-Child Personas**

Synthesis Revision	Implementation Revision
blinking_led_parent, blinking_led_default	blinking_led_pr_alpha
blinking_led_parent, blinking_led_child_slow	blinking_led_pr_bravo
blinking_led_parent, blinking_led_child_empty	blinking_led_pr_charlie
blinking_led_parent_slow, blinking_led_child_slow	blinking_led_pr_delta
blinking_led_parent_slow, blinking_led_child_empty	blinking_led_pr_emma

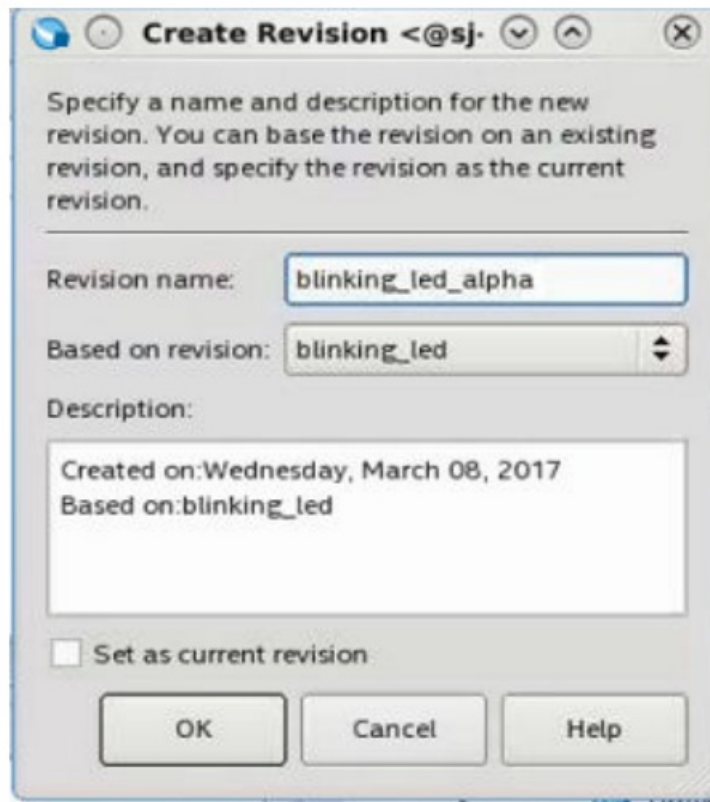
#### Creating Implementation Revisions

##### 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, blinking\_led\_pr\_charlie, blinking\_led\_pr\_delta, and blinking\_led\_pr\_emma 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 the `blinking_led_default.qsf` file to include the following assignments:  
`set_global_assignment -name TOP_LEVEL_ENTITY blinking_led_child`  
`set_global_assignment -name SYSTEMVERILOG_FILE`
4. Similarly, create `blinking_led_child_slow`, `blinking_led_child_empty`, `blinking_led_parent`, and `blinking_led_parent_slow` revisions based on `blinking_led` revision. Do not set these revisions as current revisions.
5. Update the `blinking_led_child_slow.qsf`, `blinking_led_child_empty.qsf`, `blinking_led_parent.qsf`, and `blinking_led_parent_slow.qsf` files with their corresponding **TOP\_LEVEL\_ENTITY** and **SYSTEMVERILOG\_FILE** assignments:

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

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

```
##blinking_led_parent.qsf

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

```
##blinking_led_parent_slow.qsf

set_global_assignment -name TOP_LEVEL_ENTITY blinking_led_slow
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led_slow.sv
set_global_assignment -name SYSTEMVERILOG_FILE blinking_led_child.sv
```

6. To avoid synthesis errors, ensure that the synthesis revision files for the child partitions do not contain any design partition, pin assignments, or Logic Lock region assignments. Also, the synthesis revision files for the parent partitions must only contain design partition assignments for the corresponding child partitions. Remove these assignments, if any, in the blinking\_led\_default.qsf, blinking\_led\_child\_slow.qsf, blinking\_led\_child\_empty.qsf, blinking\_led\_parent.qsf, and blinking\_led\_parent\_slow.pdf files

```
#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 PLACE_REGION "69 10 88 29" -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 ROUTE_REGION "68 9 89 30" -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 PLACE_REGION "66 7 112 32" -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 "65 6 113 33" -to \
    u_blinking_led
```

7. Include the following assignments in blinking\_led\_parent.qsf and blinking\_led\_parent\_slow.qsf files:

```
set_instance_assignment -name PARTITION pr_partition -to \
    u_blinking_led_child
set_instance_assignment -name PARTIAL_RECONFIGURATION_PARTITION ON -to \
    u_blinking_led_child
```

8. 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_pr_delta"
PROJECT_REVISION = "blinking_led_pr_emma"
PROJECT_REVISION = "blinking_led_default"
PROJECT_REVISION = "blinking_led_child_slow"
PROJECT_REVISION = "blinking_led_child_empty"
PROJECT_REVISION = "blinking_led"
PROJECT_REVISION = "blinking_led_slow"
```

- **Note:** If you are copying the revision files from hpr 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_child_empty.qsf	Partial Reconfiguration – Persona Synthesis
blinking_led_child_slow.qsf	Partial Reconfiguration – Persona Synthesis
blinking_led_parent.qsf	Partial Reconfiguration – Persona Synthesis
blinking_led_parent_slow.qsf	Partial Reconfiguration – Persona Synthesis
blinking_led_pr_alpha.qsf	Partial Reconfiguration – Persona Implementation
blinking_led_pr_bravo.qsf	Partial Reconfiguration – Persona Implementation
blinking_led_pr_charlie.qsf	Partial Reconfiguration – Persona Implementation
blinking_led_pr_delta.qsf	Partial Reconfiguration – Persona Implementation
blinking_led_pr_emma.qsf	Partial Reconfiguration – Persona Implementation

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 ► General.
  5. Select the Revision Type as Partial Reconfiguration – Base.
  6. Similarly, set the revision types for the other ten 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_child_slow.qsf  
set_global_assignment -name REVISION_TYPE PR_SYN
```

```
##blinking_led_child_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_parent.qsf  
set_global_assignment -name REVISION_TYPE PR_SYN
```

```
##blinking_led_parent_slow.qsf  
set_global_assignment -name REVISION_TYPE PR_SYN
```

```
##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
```

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

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

- **Note:** Add any Fitter-specific settings that you want 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 Information

### Create Revisions for Personas

#### Step 8: Generating the Hierarchical Partial Reconfiguration Flow Script

To generate the hierarchical partial reconfiguration flow script:

1. From the Intel Quartus Prime command shell, create a flow template by running the following command:
2. Intel Quartus Prime generates the a10\_hier\_partial\_reconfig/flow.tcl file.

```
quartus_sh --write_flow_template -flow a10_hier_partial_reconfig
```

3. Rename the generated a10\_hier\_partial\_reconfig/setup.tcl.example to a10\_hier\_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
```

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

```
#####
#####
# SETUP CONFIGURTION SCRIPT
#####
#####
# Define the name of the project.
define_project blinking_led

# Define the base revision name. This revision represents the static
# region of the design
define_base_revision blinking_led

# Define each of the partial reconfiguration implementation revisions
define_pr_impl_partition -impl_rev_name blinking_led_pr_alpha \
    -partition_name pr_partition \
    -source_rev_name blinking_led_default \
    -source_partition root_partition \
    -source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_alpha \
    -partition_name pr_parent_partition \
    -source_rev_name blinking_led_parent \
    -source_partition root_partition \
    -source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_bravo \
    -partition_name pr_partition \
    -source_rev_name blinking_led_child_slow \
    -source_partition root_partition \
    -source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_bravo \
    -partition_name pr_parent_partition \
    -source_rev_name blinking_led_pr_alpha \
    -source_partition pr_parent_partition \
    -source_snapshot final

define_pr_impl_partition -impl_rev_name blinking_led_pr_charlie \
    -partition_name pr_partition \
    -source_rev_name blinking_led_child_empty \
```

**Note:** All the revision projects must be in the same directory as blinking\_led.qpf. Otherwise, update the flow script accordingly.

```

-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_charlie \
-partition_name pr_parent_partition \
-source_rev_name blinking_led_pr_alpha \
-source_partition pr_parent_partition \
-source_snapshot final

define_pr_impl_partition -impl_rev_name blinking_led_pr_delta \
-partition_name pr_partition \
-source_rev_name blinking_led_child_slow \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_delta \
-partition_name pr_parent_partition \
-source_rev_name blinking_led_parent_slow \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_emma \
-partition_name pr_partition \
-source_rev_name blinking_led_child_empty \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_emma \
-partition_name pr_parent_partition \
-source_rev_name blinking_led_pr_delta \
-source_partition pr_parent_partition

```

## Step 9: Running the Hierarchical Partial Reconfiguration Flow Script

### To run the hierarchical 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\_hier\_partial\_reconfig/flow.tcl.
3. Select the a10\_hier\_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:

### Related Information

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

## Step 10: Programming the Board

### Before you begin:

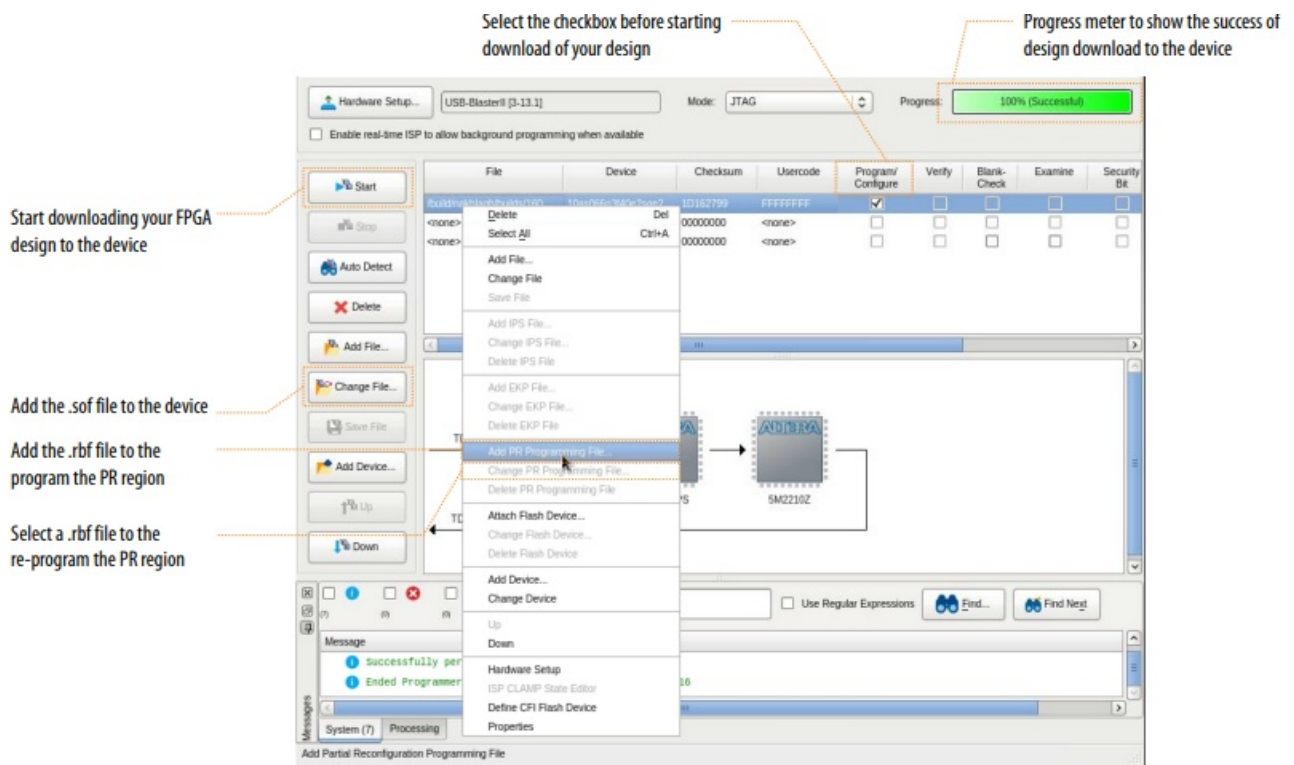
1. Connect the power supply to the Intel Arria 10 SoC development board.

2. Connect the USB Blaster cable between your PC USB port and the USB Blaster port on the development board.

**To run the design on the Intel Arria 10 SoC 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, 10AS066N3.
4. Click OK. The Intel Quartus Prime software detects and updates the Programmer with the three FPGA chips on the board.
5. Select the 10AS066N3 device, click Change File and load the blinking\_led\_pr\_alpha.sof file.
6. Enable Program/Configure for blinking\_led\_pr\_alpha.sof file.
7. Click Start and wait for the progress bar to reach 100%.
8. Observe the LEDs on the board blinking at the same frequency as the original flat design.
9. To program only the child 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.pr\_parent\_partition.pr\_partition.rbf file.
11. Disable Program/Configure for blinking\_led\_pr\_alpha.sof file.
12. Enable Program/Configure for blinking\_led\_pr\_bravo.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 Programming File.
14. Select the blinking\_led\_pr\_delta.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 10. Programming the Intel Arria 10 SoC 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_child_slow` persona to blink even slower:
- In the `blinking_led_child_slow.sv` file, modify the `COUNTER_TAP` parameter from 27 to 28.
  - To re-synthesize and re-implement this persona, you must recompile all the synthesis-only revisions and implementation revisions affected by the change. Modify the `setup.tcl` script to include the following lines:

```
define_project blinking_led

define_base_revision blinking_led

define_pr_impl_partition -impl_rev_name blinking_led_pr_bravo \
    -partition_name pr_partition \
    -source_rev_name blinking_led_child_slow \
    -source_partition root_partition \
    -source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_bravo \
    -partition_name pr_parent_partition \
    -source_rev_name blinking_led_pr_alpha \
    -source_partition pr_parent_partition \
    -source_snapshot final

define_pr_impl_partition -impl_rev_name blinking_led_pr_delta \
    -partition_name pr_partition \
    -source_rev_name blinking_led_child_slow \
    -source_partition root_partition \
    -source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_delta \
    -partition_name pr_parent_partition \
```

**Note:** When defining the `pr_parent_partition` for `blinking_led_pr_delta` revision, you import the final snapshot of that persona for implementation. As a result, the implementation of the parent partition logic remains the same, while modifying and implementing the corresponding child partition.

```
-source_rev_name blinking_led_pr_delta \
-source_partition pr_parent_partition \
-source_snapshot final
```

This command re-synthesizes the blinking\_led\_child\_slow synthesis revision, and then runs the PR implementation compile using blinking\_led\_pr\_bravo.

3. To perform compilation of the synthesis-only revisions, run the following command: This command does not recompile the base revision.

```
quartus_sh -t a10_hier_partial_reconfig/flow.tcl -setup_script \
a10_hier_partial_reconfig/setup.tcl -all_impl
```

4. To perform compilation of the implementation revisions, run the following command:

```
quartus_sh -t a10_hier_partial_reconfig/flow.tcl -setup_script \
a10_hier_partial_reconfig/setup.tcl -all_impl
```

5. This command does not recompile the base revision.
6. Follow the steps in Step 10: Programming the Board on page 22 to program the resulting RBF file into the FPGA.

**Note:** To avoid running the entire flow for every revision, define the synthesis-only revisions and implementation revisions in the setup.tcl script, and run the script.

## 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 child persona for blinking\_led\_parent\_slow, that turns led\_three off:

1. Copy blinking\_led\_child\_empty.sv to blinking\_led\_child\_off.sv.
2. In the blinking\_led\_child\_off.sv file, modify the assignment, assign led\_three\_on = 1'b0; to assign led\_three\_on = 1'b1;. Ensure you change the module name from blinking\_led\_child\_empty to blinking\_led\_child\_off.
3. Create a new synthesis revision, blinking\_led\_child\_off, by following the steps in Creating Synthesis-Only Revisions on page 16.

**Note:** The blinking\_led\_child\_off revision must use the blinking\_led\_child\_off.sv file.

4. Create a new implementation revision, blinking\_led\_pr\_foxtrot, by following the steps in Creating Implementation Revisions on page 15.
5. Update the a10\_hier\_partial\_reconfig/setup.tcl file to define the new PR implementation:

```
define_pr_impl_partition -impl_rev_name blinking_led_pr_foxtrot \
-partition_name pr_partition \
-source_rev_name blinking_led_child_off \
-source_partition root_partition \
-source_snapshot synthesized

define_pr_impl_partition -impl_rev_name blinking_led_pr_foxtrot \
-partition_name pr_parent_partition \
-source_rev_name blinking_led_pr_delta \
-source_partition pr_parent_partition \
-source_snapshot final
```

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

```
quartus_sh -t a10_hier_partial_reconfig/flow.tcl -setup_script \
a10_hier_partial_reconfig/setup.tcl -all_syn

quartus_sh -t a10_hier_partial_reconfig/flow.tcl -setup_script \
a10_hier_partial_reconfig/setup.tcl -all_impl
```



For complete information on hierarchical partial reconfiguration for Intel Arria 10 devices, refer to Creating a Partial Reconfiguration Design in Volume 1 of the Intel Quartus Prime Pro Edition Handbook.

## Related Information


- Creating a Partial Reconfiguration Design
- Partial Reconfiguration Online Training

## Document Revision History

Table 5. Document Revision History

Document Version	Software Version	Changes
2017.11.06	17.1.0	<ul style="list-style-type: none"><li>• Updated the <i>Reference Design Requirements</i> section with the software version</li><li>• Updated the <i>Flat Reference Design without PR Partitioning</i> figure with design block changes</li><li>• Updated the <i>Reference Design Files</i> table with information on the Top_counter.sv module</li><li>• Updated the <i>Partial Reconfiguration IP Core Integration</i> figure with design block changes</li><li>• Updated the figures – <i>Design Partitions Window</i> and <i>Logic Lock Regions Window</i> to reflect the new GUI</li><li>• File name changes</li><li>• Text edits</li></ul>
2017.05.08	17.0.0	The initial release of the document

## Documents / Resources

	<p><a href="#">intel AN 805 Hierarchical Partial Reconfiguration of a Design on Arria 10 SoC Development Board [pdf]</a> User Guide</p> <p>AN 805, Hierarchical Partial Reconfiguration of a Design on Arria 10 SoC Development Board, AN 805, Hierarchical Partial Reconfiguration of a Design on Arria 10 SoC Development Board, Reconfiguration of a Design on Arria 10 SoC Development Board, Arria 10 SoC Development Board, 10 SoC Development Board</p>
---	--

## References



- [intel PSG Documentation](#)
- [intel PSG Documentation](#)
- [intel PSG Documentation](#)
- [intel PSG Documentation](#)
- [intel PSG Documentation](#)
- [intel PSG Documentation](#)
- [intel PSG Documentation](#)
- [intel PSG Documentation](#)
- [intel PSG Documentation](#)
- [GitHub - intel/fpga-partial-reconfig: Tutorials, scripts and reference designs for the Intel FPGA partial reconfiguration \(PR\) design flow](#)
- [intel Intel FPGA Technical Training Catalog | Intel](#)
- [intel Hierarchical Partial Reconfiguration Tutorial for Intel® Arria® 10...](#)
- [intel 1. Answers to Top FAQs](#)
- [intel Intel® FPGAs and Programmable Devices-Intel® FPGA](#)
- [intel Intel® FPGAs and Programmable Devices-Intel® FPGA](#)
- [intel Intel® FPGAs and Programmable Devices-Intel® FPGA](#)
- [intel Intel® FPGAs and Programmable Devices-Intel® FPGA](#)
- [intel Intel® FPGAs and Programmable Devices-Intel® FPGA](#)
- [intel Intel ISO 9001:2015 Registrations](#)

Manuals±.