



XAPP1261 (v1.0) June 19, 2015

Demonstration of Soft Error Mitigation IP and Partial Reconfiguration Capability on Monolithic Devices

Author: Michael Welter

Summary

This application note outlines how to use the UltraScale™ architecture Soft Error Mitigation (SEM) IP example design in conjunction with Partial Reconfiguration (PR) in monolithic devices. SEM IP and PR with SSI devices are currently not supported. While this reference design targets the Xilinx KCU105 evaluation board, it can be targeted for different devices, family architectures, and boards. The process for modifying the top-level SEM IP example design file is the same.

There are various reasons to use SEM IP with Partial Reconfiguration. SEM IP provides the ability to detect and correct soft errors in the configuration memory of UltraScale FPGAs. While SEM IP does not prevent soft errors, it does provide a method to better manage the system-level effects of soft errors. For more information on the UltraScale architecture SEM IP, see the *UltraScale Architecture Soft Error Mitigation Controller Product Guide* (PG187) [Ref 1]. Partial Reconfiguration provides the ability to time multiplex hardware dynamically on a single FPGA. This can provide the following benefits:

- Increase solution flexibility by time-multiplexing design functionality
- Reduce FPGA size or count (and therefore cost) by time-sharing functionality
- Reduce dynamic power consumption by loading functions on-demand

Using SEM IP with PR provides the benefits of detecting/correcting soft configuration errors before and after PR has been performed. The reference design provided with this document uses SEM IP with two Reconfigurable Partitions (RP).

The General Purpose Input/Output (GPIO) LEDs are used to provide visual indicators that PR occurred successfully, and the UART output is used to demonstrate that SEM IP continues to operate normally with PR.

Introduction

This reference design is the SEM IP example design that has been modified to include two RPs, which each have two Reconfigurable Modules (RMs). The first RP is a counter that either counts up or down, and the second RP is a block RAM shifter that shifts data from left-to-right or right-to-left. Both RPs are connected to the GPIO LEDs of the evaluation board.

After a full bitstream is loaded into a device, SEM IP calculates the golden frame ECC values and the device-level CRC value for the entire configuration memory. Anytime SEM IP is in the Observation state, it scans for errors in the configuration memory based upon these values. When PR is performed, the configuration memory is changed. To prevent false errors from being detected after PR is performed, SEM IP must recalculate the golden frame ECC values and the device-level CRC value. This must be performed before SEM IP transitions to the Observation state where it begins scanning for errors. Therefore after PR completes, SEM IP must be commanded to perform a soft reset. This results in the golden frame ECC values and the device-level CRC value for entire configuration memory being recalculated. Then SEM IP can be commanded to the Observation state where it scans the entire configuration memory for errors based on the PR changes that occurred.

SEM IP and PR must both access the configuration engine in the FPGA, but they cannot both access it at the same time. Therefore access to the configuration engine is managed with the following steps:

1. After configuring the device with a full bitstream, and prior to performing PR, SEM IP must stop scanning for errors. This can be accomplished by either commanding SEM IP to the Idle state or by using the ICAP Arbitration Interface to request control of the ICAP interface from SEM IP.

Note: Depending on the family architecture, the method of stopping SEM IP from scanning/relinquishing control of the ICAP interface is different.

2. Perform Partial Reconfiguration.
3. Reset SEM IP or grant SEM IP control of the ICAP interface by using the ICAP Arbitration Interface. SEM IP recalculates the golden frame ECC and device-level CRC for the entire configuration memory accounting for PR that was performed.

For more information on the ICAP Arbitration Interface, see the *UltraScale Architecture Soft Error Mitigation Controller Product Guide* (PG187) [[Ref 1](#)].

After PR is performed, the SEM controller is used to inject errors in PR and non-PR regions of the device. The errors are detected and corrected by the SEM controller when it is transitioned to the Observation state to resume scanning.

SEM IP and PR can be used together in most use cases. The time it takes to perform PR is dependent on the initiation of PR, the portion of the device being reconfigured, and the delivery of the partial BIT files. After PR, SEM IP must be given time to initialize and scan the device for errors. Therefore consideration must be given to use cases that require a high rate of PR events.

Using PR and SEM IP is most beneficial for cases that require infrequent reconfiguration. It is also reasonable to use PR with SEM IP in cases in which reconfiguration is performed less than once per minute or in many cases less than once per second. However if reconfiguration is performed more frequently, see the [Limitations and Considerations](#).

Reference Design

This design illustrates how SEM IP functions normally with Partial Reconfiguration. There are limitations and considerations that must be accounted for, see the [Limitations and Considerations](#).

Two main components of this design include:

- **SEM IP Example Design** – The design must run through synthesis to generate all of the DCP files required by the reference design Tcl script.
- **Reconfigurable Modules** – This contains the Count Module (up/down count) and block RAM Shift Module (left-to-right/right-to-left).

This reference design is essentially the SEM IP example design that has been modified to include two Reconfigurable Partitions, each with two Reconfigurable Modules. It contains the minimal port connections to test the reference design. LEDs are used to provide visual indicators that a change has occurred while the reference design is running.

The GPIO LEDs are connected to the RPs. The count RP can count either up or down, and it is built with slice logic. The counter is connected to the lowest four GPIO LEDs on an evaluation board. The shift RP is implemented with the block RAM and is connected to the highest four GPIO LEDs on an evaluation board. The block RAM in each shift RM is initialized with data that shifts the LEDs from left-to-right or right-to-left. The LEDs should be observed to verify the count and/or shift RMs were changed after PR.

For the required modifications to the SEM IP example design and constraints, see the following sections:

- [HDL Modifications – Copy, Modify, and Rename sem_ultra_0_example_design.v to pr_example.v](#)
- [XDC Modifications – Copy, Modify, and Rename sem_ultra_0_example_design.xdc to pr_example.xdc](#)

[Table 1](#) describes the ports used to implement the design. The `reset` and `led_out[7:0]` ports were added to the SEM IP example design and are connected to the Reconfigurable Partitions.

Table 1: pr_example Ports

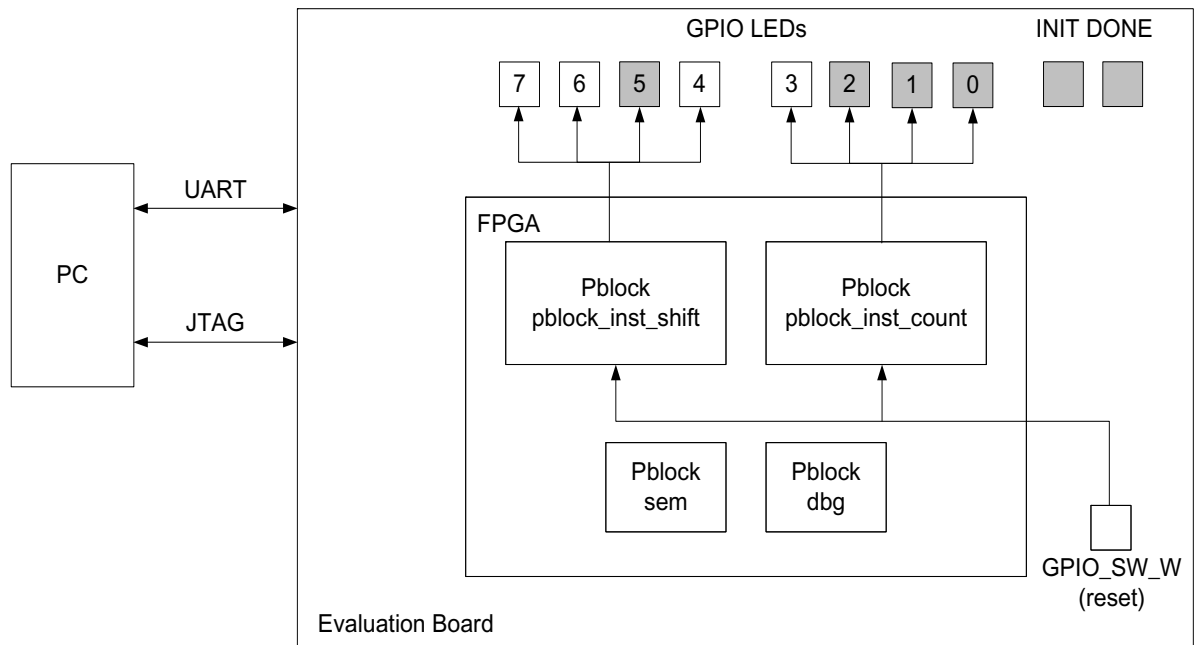
Signal Name	I/O	Description
reset	Input	Reset signal that only connects the two RPs to the GPIO pushbutton switch (GPIO_SW_W).
led_out[7:0]	Output	LED Output Bus LED[3:0] = Counter function LED[7:4] = Shift function
clock	Input	System clock, drives the ICAP, SEM controller, and RPs.

Table 1: pr_example Ports

Signal Name	I/O	Description
uart_tx	Output	Serial transmit data.
uart_rx	Input	Serial receive data.

Hardware Block Diagram

Figure 1 shows a high-level description of the reference design. A PC is used to connect to the evaluation board JTAG port and USB to serial port.



X14715-061215

Figure 1: Reference Design Block Diagram

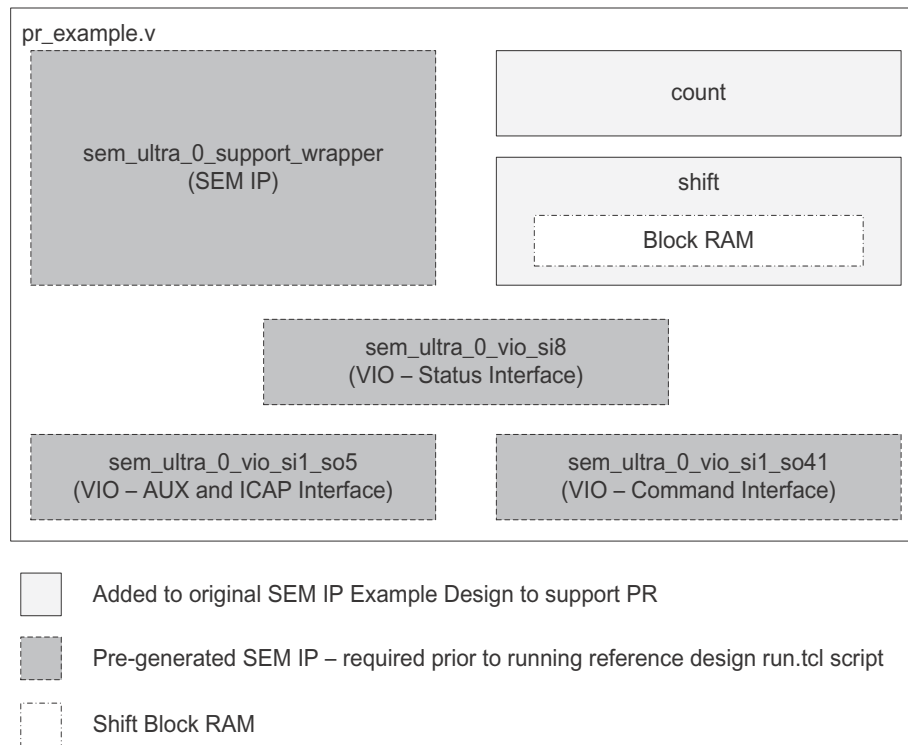
Creating the Reference Design

This reference design uses the SEM IP example design and adds reconfigurable regions to the FPGA design. SEM IP is contained within the Static region of the FPGA. This is the region of the FPGA that is not reconfigured. The count and shift RPs are contained within user-defined reconfigurable regions. After the FPGA has been configured with a full BIT file, a partial BIT file is used to modify the reconfigurable regions in the design without compromising the integrity of SEM IP and its function.

Figure 2 shows the top-level design and all of the supporting IP and PR modules required. Before implementing PR on this reference design, the SEM IP core and example design had to be generated and synthesized. This provides all of the SEM IP files required by the reference design.

Two files, `sem_ultra_0_example_design.v` and `sem_ultra_0_example_design.xdc` were used to create `pr_example.v` and `pr_example.xdc`, respectively. The XDC file was also modified to add the GPIO LED connections and the PR Pblock definitions.

Finally, the `run.tcl` file is used to generate the full and partial BIT files. PR is supported through Tcl or command line only; there is no project support at this time.



X14716-061915

Figure 2: Top-Level HDL Diagram

The steps required to generate the reference design are discussed in the following section.

Generating the Reference Design

SEM IP – Generate SEM IP Core from the Vivado IP Catalog

1. Generate the SEM IP core from the Vivado IP catalog.



IMPORTANT: SEM IP must be regenerated any time a different version of Vivado® Design Suite is used or when a new device or architecture family is targeted. For more information on how to generate the IP core, see the Vivado Design Suite User Guide: Designing with IP (UG896) [Ref 2] in conjunction with the appropriate family architecture product guides for SEM IP. For this reference design, see the UltraScale Architecture Soft Error Mitigation Controller Product Guide (PG187) [Ref 1].

The SEM IP core was generated with the following settings:

```

MODE = mitigation_and_testing
INTERFACE = Stand-alone
CLOCK_PERIOD = 11111
ENABLE_CLASSIFICATION = false
  
```

```

ENABLE_CONFIG_SCAN      = false
MEMORY_TYPE             = none
MEMORY_IO_TYPE          = none
LOCATE_CONFIG_PRIM      = example_design
LOCATE_HELPER_BLOCKS    = example_design
Component_Name          = sem_ultra_0

```



TIP: For the complete list of options, see the `sem_ultra_0.xci` file if needed.

- Next, generate the output products. When the project is created for the first time, the output products are auto-generated. If using an existing project, open the Vivado project file (`pr_sem_jtag_KCU105_KU040/Sources/ip/SEM_KCU105_KU040/KCU105_KU040/KCU105_KU040.xpr`), right-click `sem_ultra_0` (`sem_ultra_0.xci`), select **Generate Output Products**, and click **Generate** (see [Figure 3](#)).

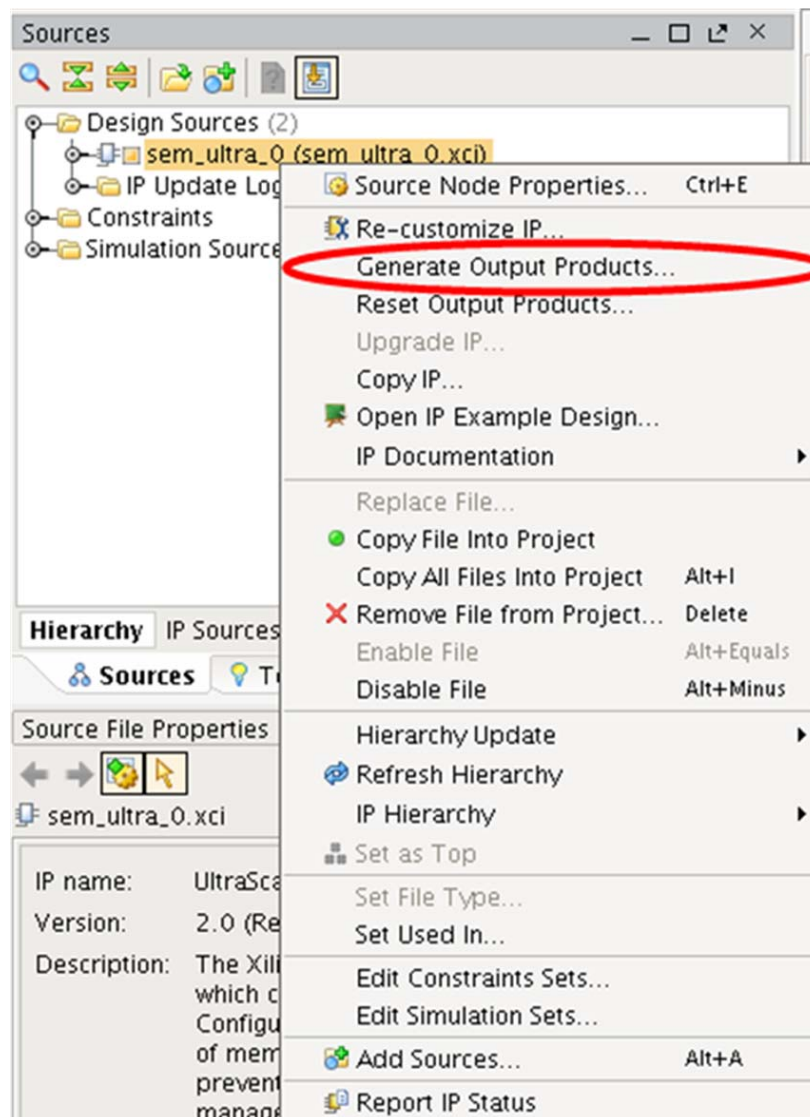


Figure 3: **Generate Output Products**

3. After the **Generate Output Products** step has completed, right-click the `sem_ultra_0` (`sem_ultra_0.xci`) and select **Open Example Design...**

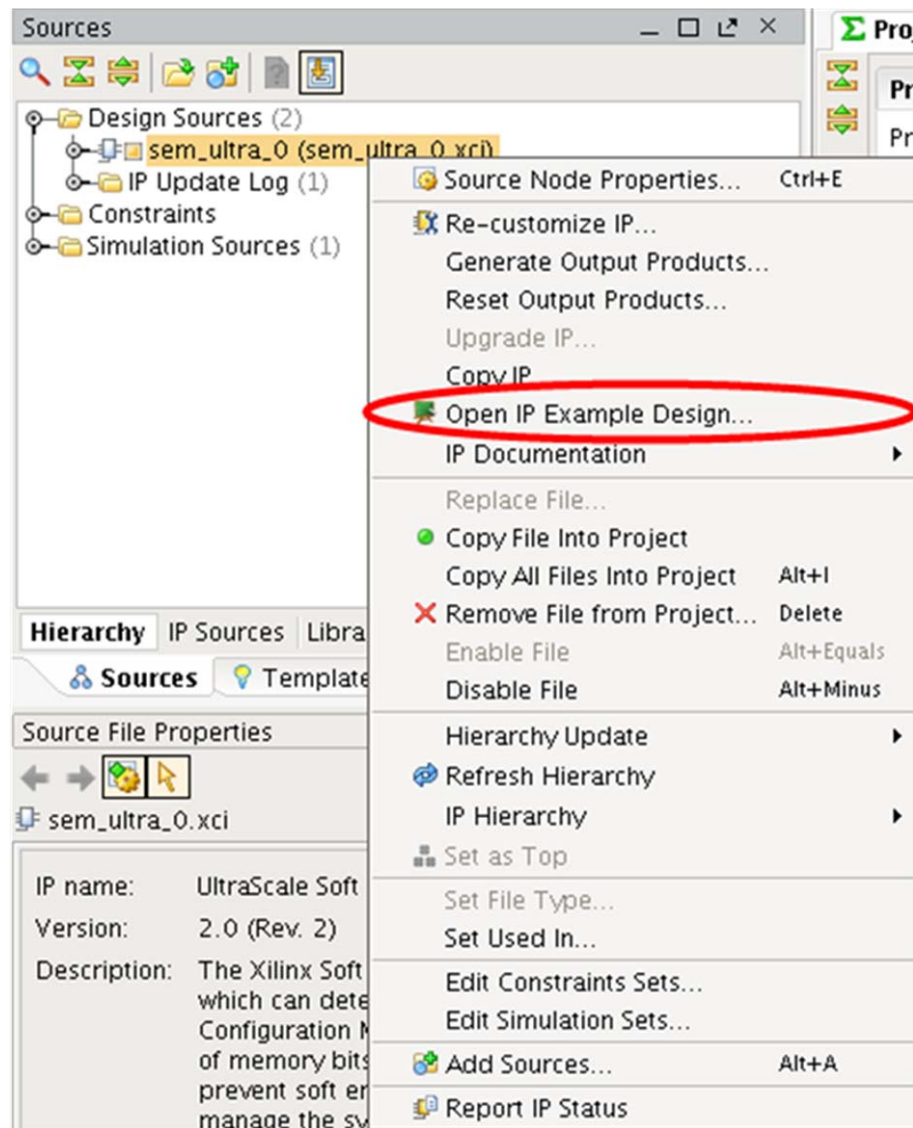


Figure 4: **Generate SEM IP Example Design**

4. Choose **Location For Example Project Directory** `sem_ultra_0_example` to be in the `SEM_KCU105_KU040/KCU105_KU040` directory and then click **Select** followed by **OK**.

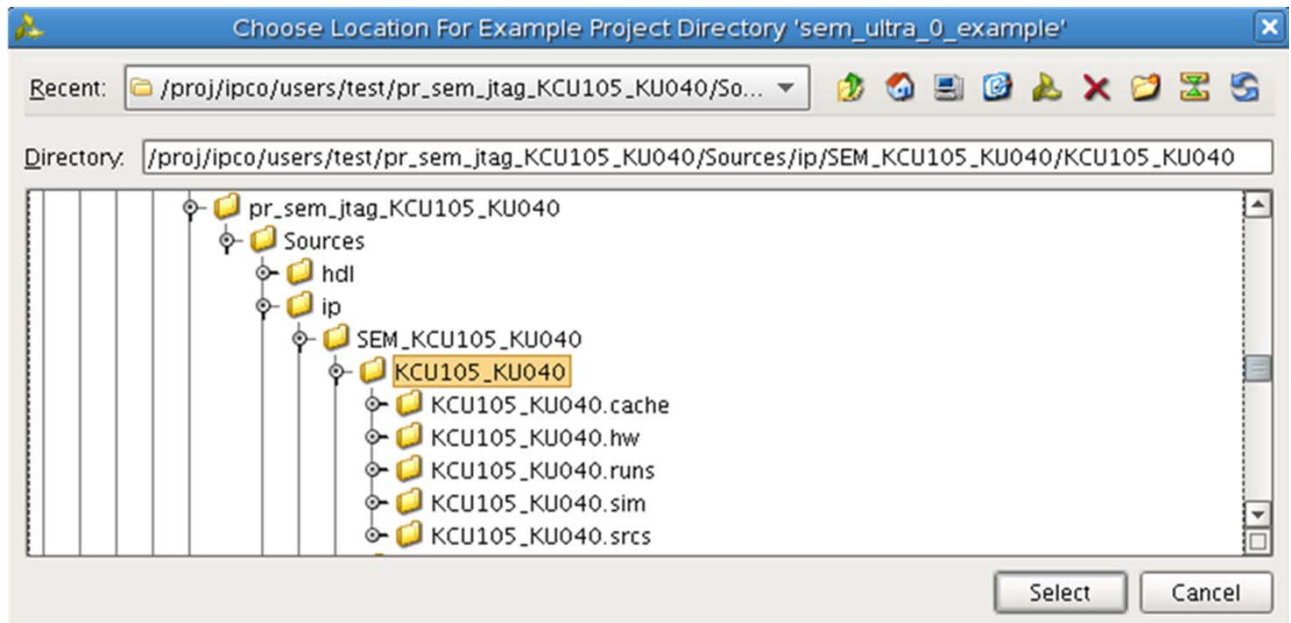


Figure 5: Location for Example Project Directory

5. After the Vivado Design Suite loads the example design, run synthesis (**Flow > Run Synthesis**). This generates all of the required files for the reference design.

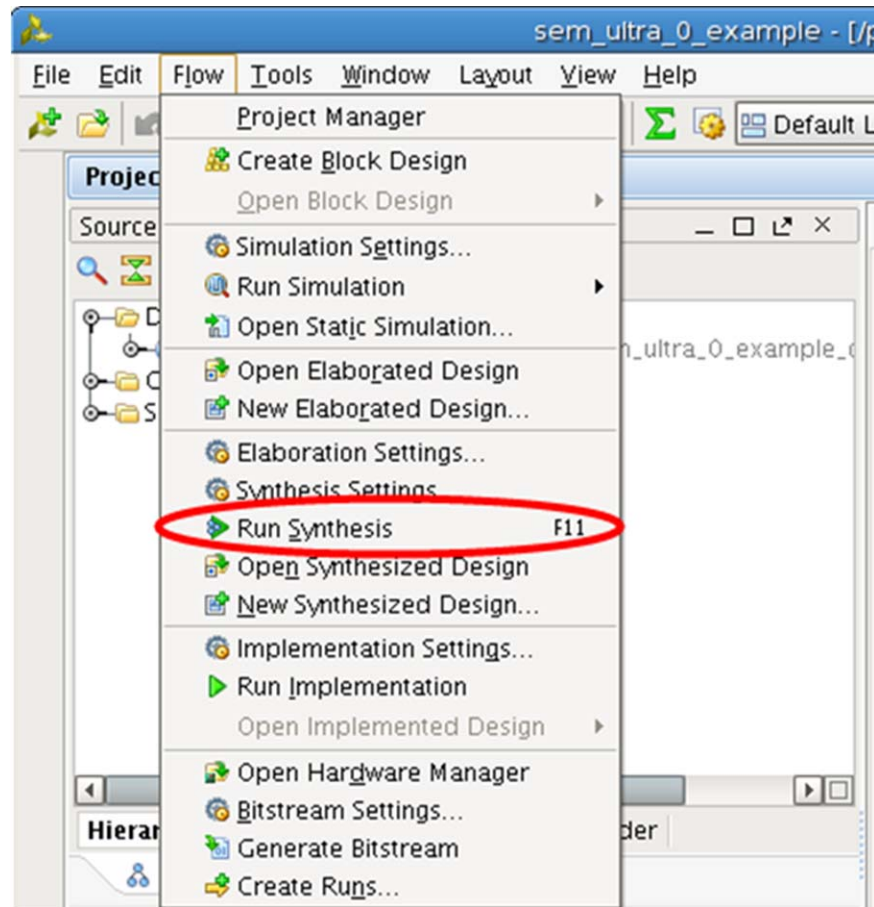


Figure 6: Run Synthesis

6. Next, define the RP Pblocks. Implementation does not need to be run, but the synthesized design must be opened to define the Pblock constraints for the RPs. In the **Synthesis Completed** pop-up window, select the **Open Synthesized Design** and click **OK**.

If a pop-up does not appear, use the Vivado Flow menu to open the synthesized design (**Flow > Open Synthesized Design**).

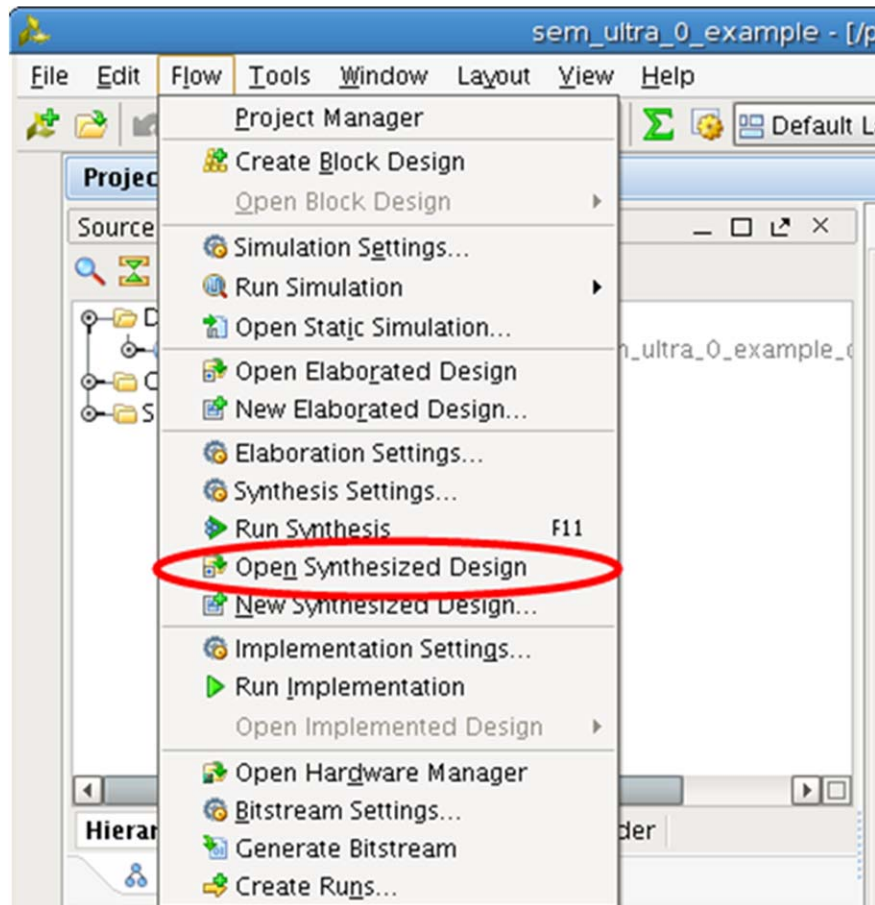


Figure 7: Open Synthesized Design

- Open the **Device View** (**Window > Device**).

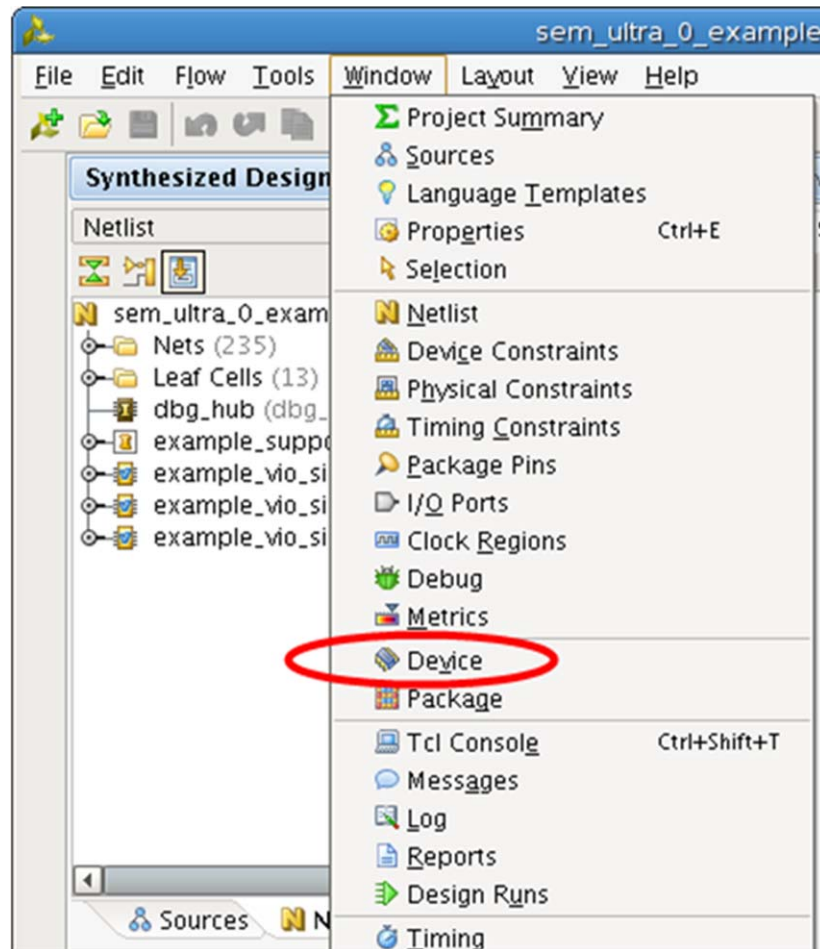


Figure 8: **Device View**

- Now, only SEM IP Pblocks are in the design and two RP Pblocks must be defined. When defining a Pblock, contain it to a single clock region. For more information, see the "Create a Floorplan for the Reconfigurable Region" section in the *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909) [Ref 3].

Draw a new Pblock to define the count RP. This Pblock will span from SLICE X62Y150 through SLICE X66Y179. Name the Pblock, `pblock_inst_count`, and verify only the Slice resources are enabled.

- Draw another Pblock to define the shift RP. This Pblock will span from SLICE X60Y120 through SLICE X60Y149, RAMB18 X6Y48 through RAMB18 X6Y59, and RAMB36 X6Y24 through RAMB36 X6Y29. Name it `pblock_inst_shift`, and verify only the Slice, RAMB18, and RAMB36 resources are enabled.

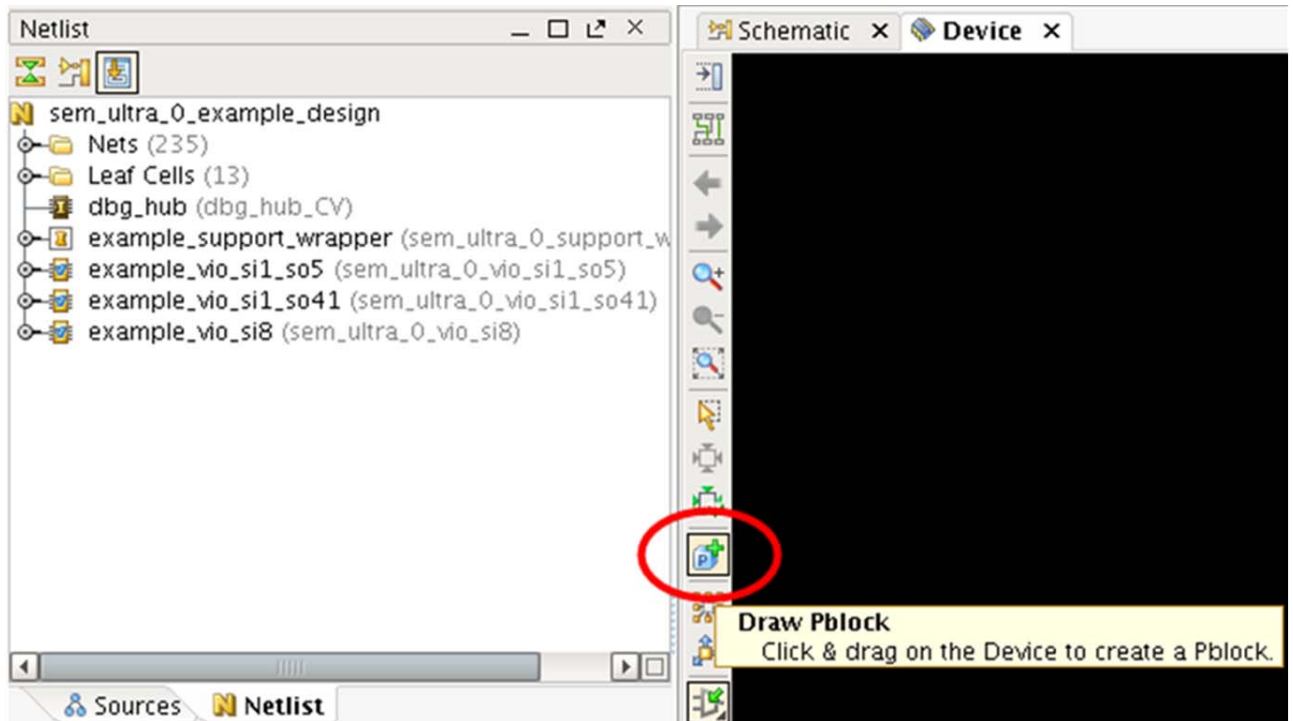


Figure 9: Drawing a Pblock

10. Figure 10 shows the final reference design Pblocks. The top Pblock is the count RP region. The center Pblock is the shift RP region. The lowest Pblock is the SEM and DBG Pblocks from the SEM IP example design. The SEM IP example design Pblocks were not modified.

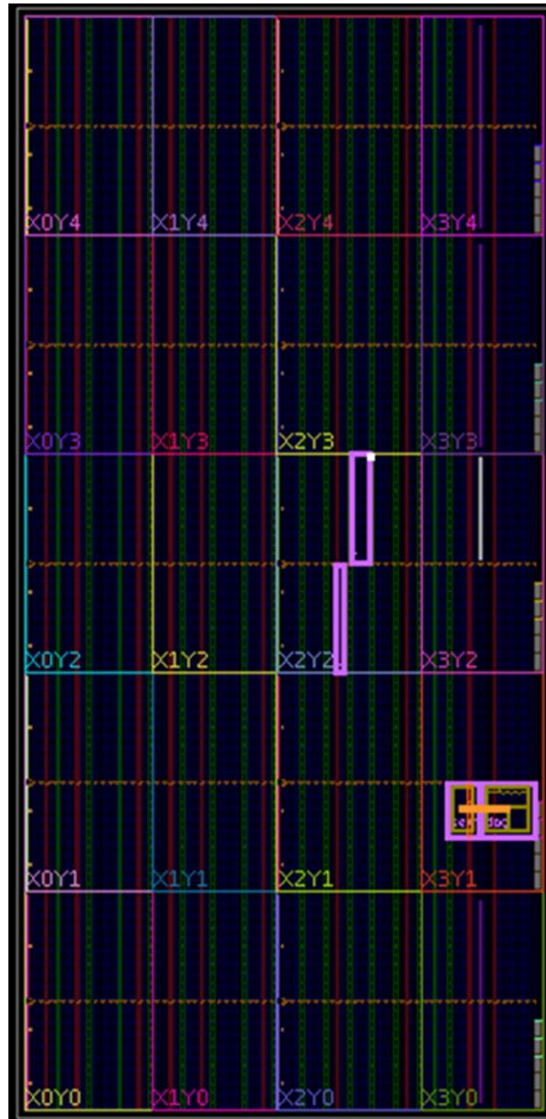


Figure 10: Final Reference Design Pblocks

11. Save the PR Pblock constraints using the **Save Constraints As...**

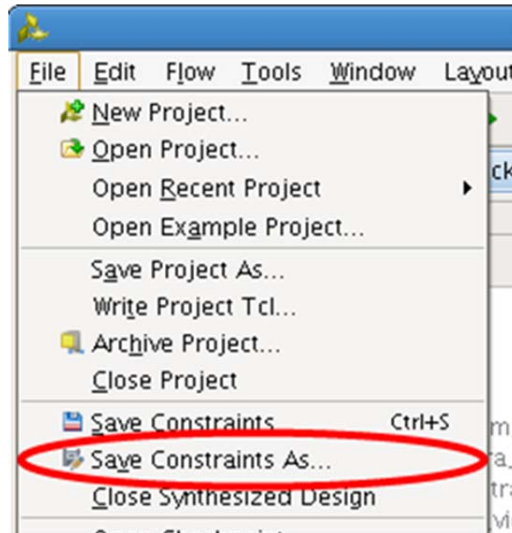


Figure 11: Save Constraints

12. Uncheck the **Make active** and set the **New Constraint set name** to `inactive_constraints`. Click **OK**.

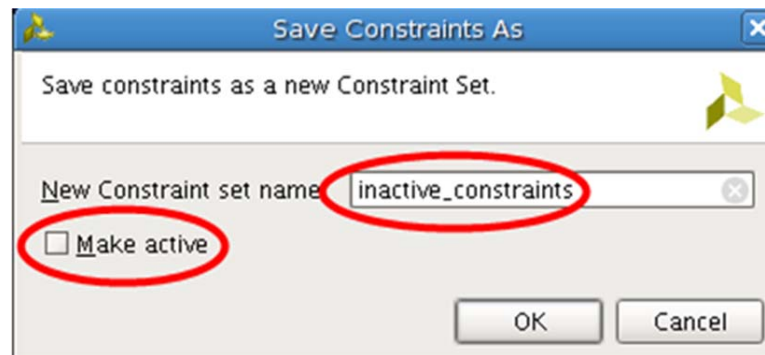


Figure 12: New Constraint Set Name

These constraints must be added to the `pr_example.xdc` constraints in the [XDC Modifications – Copy, Modify, and Rename sem_ultra_0_example_design.xdc to pr_example.xdc, page 16](#) step below. The path where these constraints are saved is listed here:

```
pr_sem_jtag_KCU105_KU040/Sources/ip/SEM_KCU105_KU040/KCU105_KU040/sem_ultra_0_example/sem_ultra_0_example.srcs/inactive_constraints/imports/example_design/sem_ultra_0_example_design.xdc
```

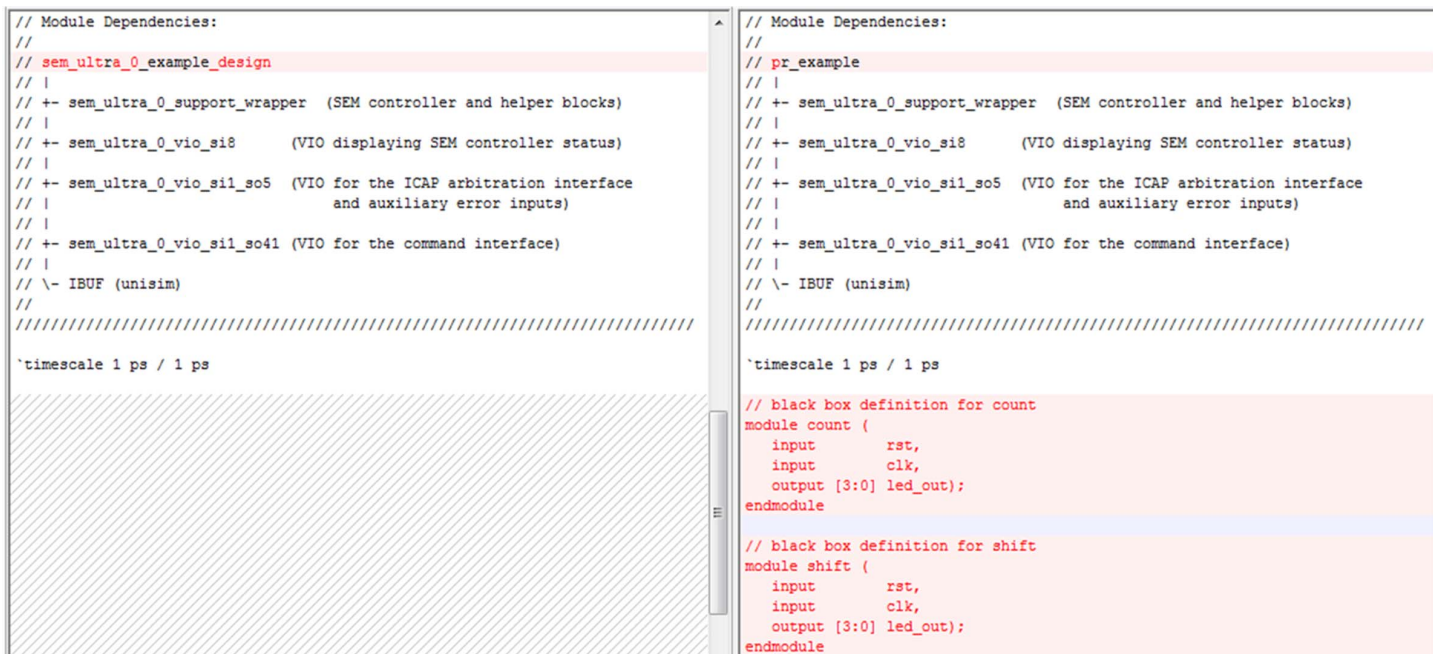
13. Close the synthesized design and all the Vivado windows.

HDL Modifications – Copy, Modify, and Rename `sem_ultra_0_example_design.v` to `pr_example.v`

The top-level HDL file, `pr_example.v`, is a modified version of the SEM IP example design top-level HDL file, `sem_ultra_0_example_design.v`. It contains the minimal port connections to test the reference design. The file has been modified to add instantiations of the RPs and the supporting I/O. Therefore, the SEM IP example design must be generated before this file can be modified.

The file on the left is the original SEM IP example design top-level HDL file. The file on the right is the top-level HDL file for this reference design which shows all of the changes in RED.

1. Add the Black Box Definitions for the RPs.



```
// Module Dependencies:
//
// sem_ultra_0_example_design
// |
// +- sem_ultra_0_support_wrapper (SEM controller and helper blocks)
// |
// +- sem_ultra_0_vio_si8 (VIO displaying SEM controller status)
// |
// +- sem_ultra_0_vio_si1_so5 (VIO for the ICAP arbitration interface
// | and auxiliary error inputs)
// |
// +- sem_ultra_0_vio_si1_so41 (VIO for the command interface)
// |
// \- IBUF (unisim)
//
////////////////////////////////////
`timescale 1 ps / 1 ps

////////////////////////////////////

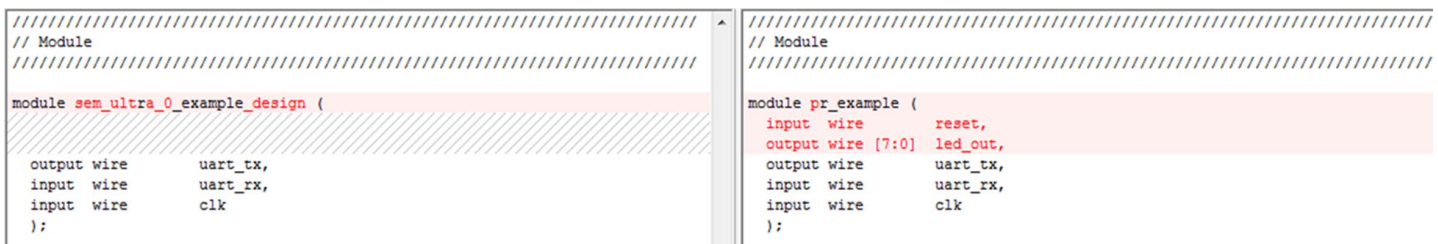
// Module Dependencies:
//
// pr_example
// |
// +- sem_ultra_0_support_wrapper (SEM controller and helper blocks)
// |
// +- sem_ultra_0_vio_si8 (VIO displaying SEM controller status)
// |
// +- sem_ultra_0_vio_si1_so5 (VIO for the ICAP arbitration interface
// | and auxiliary error inputs)
// |
// +- sem_ultra_0_vio_si1_so41 (VIO for the command interface)
// |
// \- IBUF (unisim)
//
////////////////////////////////////
`timescale 1 ps / 1 ps

////////////////////////////////////
// black box definition for count
module count (
    input rst,
    input clk,
    output [3:0] led_out);
endmodule

////////////////////////////////////
// black box definition for shift
module shift (
    input rst,
    input clk,
    output [3:0] led_out);
endmodule
```

Figure 13: Black Box Definitions for RPs

2. Add the reset input and led_out [7:0] outputs to the port list and change module name.



```
////////////////////////////////////
// Module
////////////////////////////////////

module sem_ultra_0_example_design (

    output wire    uart_tx,
    input wire     uart_rx,
    input wire     clk
);

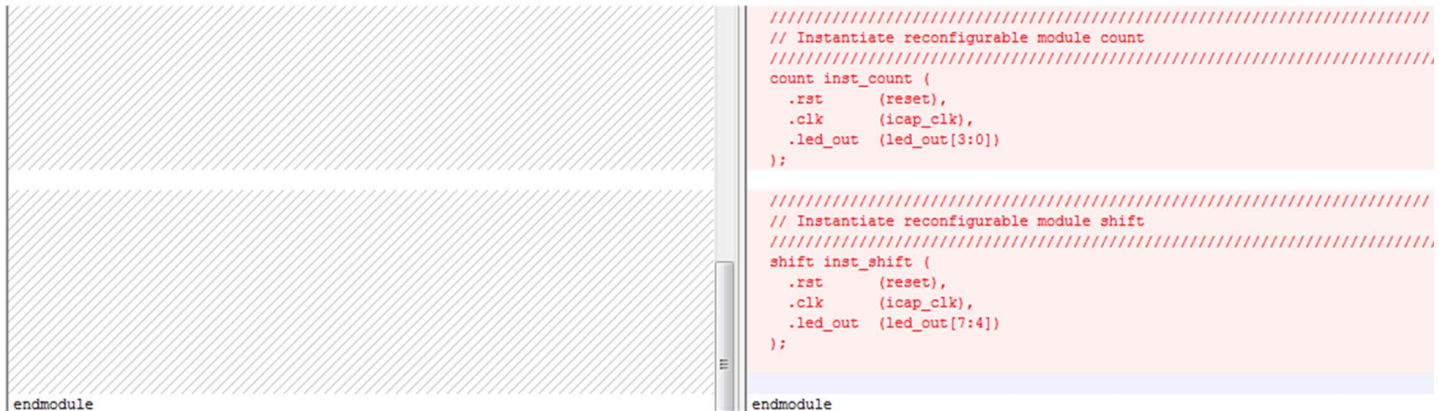
////////////////////////////////////

////////////////////////////////////
// Module
////////////////////////////////////

module pr_example (
    input wire     reset,
    output wire [7:0] led_out,
    output wire    uart_tx,
    input wire     uart_rx,
    input wire     clk
);
```

Figure 14: Reset Input and led_out[7:0] Outputs

- Then, instantiate the RMs in the file.



```

////////////////////////////////////
// Instantiate reconfigurable module count
////////////////////////////////////
count inst_count (
    .rst      (reset),
    .clk      (icap_clk),
    .led_out  (led_out[3:0])
);

////////////////////////////////////
// Instantiate reconfigurable module shift
////////////////////////////////////
shift inst_shift (
    .rst      (reset),
    .clk      (icap_clk),
    .led_out  (led_out[7:4])
);
endmodule

```

Figure 15: Instantiate RMs

XDC Modifications – Copy, Modify, and Rename sem_ultra_0_example_design.xdc to pr_example.xdc

The file `pr_example.xdc`, is a modified version of the SEM IP example design XDC file, `sem_ultra_0_example_design.xdc`. The file has been modified to add Pblocks, package pins, I/O standards, drive strength, and slew rate. Therefore, the SEM IP example design must be generated before this file can be modified.

- Add the Pblock constraints for all RPs in the `pr_example.v`. Use the constraints that were previously defined in the [SEM IP – Generate SEM IP Core from the Vivado IP Catalog, step 8](#) to cut and paste the `create_pblock` and `resize_pblock` constraints for both RPs. The constraints are listed here:

```

create_pblock pblock_inst_count
resize_pblock [get_pblocks pblock_inst_count] -add {SLICE_X62Y150:SLICE_X66Y179}
create_pblock pblock_inst_shift
resize_pblock [get_pblocks pblock_inst_shift] -add {SLICE_X60Y120:SLICE_X60Y149}
resize_pblock [get_pblocks pblock_inst_shift] -add {RAMB18_X6Y48:RAMB18_X6Y59}
resize_pblock [get_pblocks pblock_inst_shift] -add {RAMB36_X6Y24:RAMB36_X6Y29}

```

The constraints are in this file:

```

pr_sem_jtag_KCU105_KU040/Sources/ip/SEM_KCU105_KU040/KCU105_KU040/sem_ultra_0_example/
sem_ultra_0_example.srscs/inactive_constraints/imports/example_design/sem_ultra_0_examp
le_design.xdc

```

- Also, some additional constraints must be added. Add the `add_cells_to_pblock` constraint, which is used to specify the instances that belong to each of the RP Pblocks.

The final PR constraints should look like the following:

```

####Pblock pblock_inst_count####
create_pblock pblock_inst_count
add_cells_to_pblock [get_pblocks pblock_inst_count] [get_cells [list inst_count]]
resize_pblock [get_pblocks pblock_inst_count] -add {SLICE_X62Y150:SLICE_X66Y179}

```



```
#####Pblock pblock_inst_shift#####
create_pblock pblock_inst_shift
add_cells_to_pblock [get_pblocks pblock_inst_shift] [get_cells [list inst_shift]]
resize_pblock [get_pblocks pblock_inst_shift] -add {SLICE_X60Y120:SLICE_X60Y149}
resize_pblock [get_pblocks pblock_inst_shift] -add {RAMB18_X6Y48:RAMB18_X6Y59}
resize_pblock [get_pblocks pblock_inst_shift] -add {RAMB36_X6Y24:RAMB36_X6Y29}
```

For more information on Partial Reconfiguration constraints, see the *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909) [Ref 3].

3. Add package pins, I/O standards, drive strength, and slew rate (if applicable) for all I/O in `pr_example.v`. Drive strength and slew rate depends on the I/O capability of the FPGA bank and the board characteristics.

```
# Clock
set_property IOSTANDARD LVCMOS18 [get_ports clk]
set_property PACKAGE_PIN K20 [get_ports clk]
set_property CLOCK_DEDICATED_ROUTE FALSE [get_nets clk_ibufg]

# UART RX
set_property IOSTANDARD LVCMOS18 [get_ports uart_rx]
set_property PACKAGE_PIN G25 [get_ports uart_rx]

# UART TX
set_property IOSTANDARD LVCMOS18 [get_ports uart_tx]
set_property PACKAGE_PIN K26 [get_ports uart_tx]
set_property SLEW SLOW [get_ports uart_tx]
set_property DRIVE 4 [get_ports uart_tx]

# GPIO_SW_W on Bank64 (VCC1V8 1.8 volt)
set_property PACKAGE_PIN AF9 [get_ports reset]
set_property IOSTANDARD LVCMOS18 [get_ports reset]
set_property DRIVE 8 [get_ports {reset}]

# GPIO_LED0_LS on Bank64 (VCC1V2 1.8 volt)
set_property PACKAGE_PIN AP8 [get_ports {led_out[0]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led_out[0]}]
set_property DRIVE 8 [get_ports {led_out[0]}]

# GPIO_LED1_LS on Bank65 (VCC1V8 1.8 volt)
set_property PACKAGE_PIN H23 [get_ports {led_out[1]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led_out[1]}]
set_property DRIVE 8 [get_ports {led_out[1]}]

# GPIO_LED2_LS on Bank65 (VCC1V8 1.8 volt)
set_property PACKAGE_PIN P20 [get_ports {led_out[2]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led_out[2]}]
set_property DRIVE 8 [get_ports {led_out[2]}]

# GPIO_LED3_LS on Bank65 (VCC1V8 1.8 volt)
set_property PACKAGE_PIN P21 [get_ports {led_out[3]}]
set_property IOSTANDARD LVCMOS18 [get_ports {led_out[3]}]
set_property DRIVE 8 [get_ports {led_out[3]}]

# GPIO_LED7_LS on Bank65 (VCC1V8 1.8 volt)
set_property PACKAGE_PIN P23 [get_ports led_out[4]]
set_property IOSTANDARD LVCMOS18 [get_ports led_out[4]]
set_property DRIVE 8 [get_ports {led_out[4]}]

# GPIO_LED6_LS on Bank65 (VCC1V8 1.8 volt)
```

```

set_property PACKAGE_PIN R23 [get_ports led_out[5]]
set_property IOSTANDARD LVCMOS18 [get_ports led_out[5]]
set_property DRIVE 8 [get_ports {led_out[5]}]

# GPIO_LED5_LS on Bank65 (VCC1V8 1.8 volt)
set_property PACKAGE_PIN M22 [get_ports led_out[6]]
set_property IOSTANDARD LVCMOS18 [get_ports led_out[6]]
set_property DRIVE 8 [get_ports {led_out[6]}]

# GPIO_LED4_LS on Bank65 (VCC1V8 1.8 volt)
set_property PACKAGE_PIN N22 [get_ports led_out[7]]
set_property IOSTANDARD LVCMOS18 [get_ports led_out[7]]
set_property DRIVE 8 [get_ports {led_out[7]}]

```

Tcl – Use run.tcl to Implement the Design and Generate Bitstreams

PR is supported through Tcl or by command line only; there is not project support at this time. For more information on the Partial Reconfiguration flow, see the *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909) [Ref 3]. The `run.tcl` file contains the flow used to generate this reference design. To regenerate the reference design, execute the following Vivado command from the `pr_sem_jtag_KCU105_KU040` directory:

```
vivado -mode batch -source run.tcl -notrace
```

This command can be ran from a Linux Shell or a Windows Command Prompt.

The file `run.tcl`, is divided into numerous steps. The snippets from the Tcl file below are not full snippets, but contain enough information to cross-reference the information below with the `run.tcl` file. The `run.tcl` file steps are listed here:

1. Directory Structure Definition
2. Synthesize the static and reconfigurable modules separately.

- Static Region (SEM IP) uses the following commands:

```

synth_design -mode default -flatten_hierarchy rebuilt -top <top_module_name> -part
<part>
write_checkpoint -force $synthDir/Static/pr_example_synth.dcp

```

- Synthesis of RMs (`count_up`, `count_down`, `shift_left`, `shift_right`) use the following commands:

```

synth_design -mode default -flatten_hierarchy rebuilt -top pr_example -part $part
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top count -part $part
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top count -part $part
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top shift -part $part
synth_design -mode out_of_context -flatten_hierarchy rebuilt -top shift -part $part

```

3. Implementation of the initial configuration – `count_up_shift_right`. Implement a complete design in context, then save a design checkpoint for the full routed design and generate reports.

```

set_property HD.RECONFIGURABLE TRUE [get_cells <inst_name>]
HD.RECONFIGURABLE- This property is used to implement a PR design. It must be used
to specify each reconfigurable partition and it is set on the top-level
hierarchical cell that is going to be reconfigured (inst_count and inst_shift in

```

this reference design). It must be set prior to the implementation of the first configuration.

```

set_property HD.RECONFIGURABLE 1 [get_cells inst_count]
set_property HD.RECONFIGURABLE 1 [get_cells inst_shift]
read_checkpoint -cell inst_count $synthDir/count_up/count_synth.dcp -strict
read_checkpoint -cell inst_shift $synthDir/shift_right/shift_synth.dcp
-strict
opt_design
place_design
route_design
write_checkpoint -force
$implDir/Config_count_up_shift_right/pr_example_route_design.dcp
report_utilization -file
$implDir/Config_count_up_shift_right/reports/pr_example_utilization_route_de
sign.rpt

```

4. Remove reconfigurable modules from the design, and save a static only design checkpoint.

```

update_design -cell inst_count -black_box
update_design -cell inst_shift -black_box
lock_design -level routing
write_checkpoint -force $implDir/Config_count_up_shift_right/pr_example_static.dcp
file copy -force $implDir/Config_count_up_shift_right/pr_example_static.dcp $dcpDir

```

5. The current design in memory has black boxes for the RMs. Load the synthesis results for the second configuration (count down/shift left) of the design and implement the design. Then, write out the implementation checkpoint for the design and generate reports.

```

read_checkpoint -cell inst_count $synthDir/count_down/count_synth.dcp -strict
read_checkpoint -cell inst_shift $synthDir/shift_left/shift_synth.dcp -strict
opt_design
place_design
route_design
write_checkpoint -force
$implDir/Config_count_down_shift_left/pr_example_route_design.dcp
report_utilization -file
$implDir/Config_count_down_shift_left/reports/pr_example_utilization_route_design.r
pt

```

6. Verify the PR results. This confirms that the static portions of these two configurations are identical, and that the resulting bitstreams operate safely in hardware.

```

pr_verify -full_check -initial
$implDir/Config_count_up_shift_right/pr_example_route_design.dcp -additional
"$implDir/Config_count_down_shift_left/pr_example_route_design.dcp" -file
pr_verify.log

```

7. Create BIT files.

```
open_checkpoint $implDir/Config_count_up_shift_right/pr_example_route_design.dcp
write_debug_probes -force $bitDir/Config_count_up_shift_right
write_bitstream -force $bitDir/Config_count_up_shift_right
open_checkpoint $implDir/Config_count_down_shift_left/pr_example_route_design.dcp
write_debug_probes -force $bitDir/Config_count_down_shift_left
write_bitstream -force $bitDir/Config_count_down_shift_left
```

Requirements

Hardware

The following hardware is required for running the reference design:

- HW-U1-KCU105 Rev 1.0 or newer
- KU040 Production Silicon
- JTAG Programming Cable
- USB cable for the JTAG connection
- USB cable for the UART connection

Software

The following software tools are required for running the reference design:

- Vivado Design Suite 2015.2
- Tera Term 4.83 for serial UART connection

Reference Design Files

[Click here](#) to download the design files associated with this application note.

Directory Structure

After unzipping `xapp1261-demo-sem-pr.zip` the following directory structure exists.

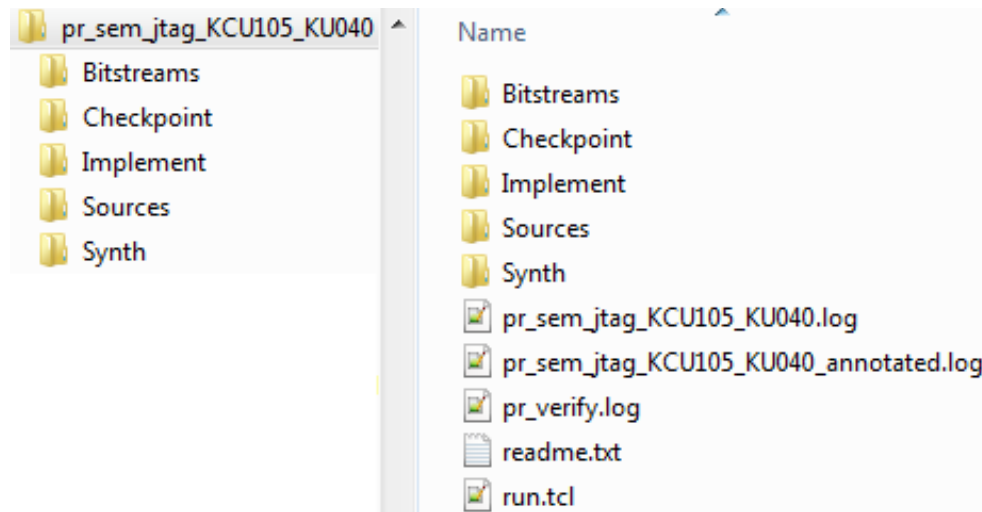


Figure 16: Top-Level Directory Structure

The directory structure is broken down further in the `readme.txt` file. A high-level description of each directory is given in the following sections.

Bitstreams

This folder contains all of the BIT and LTX files generated after `run.tcl` completes execution.

- **Full BIT and LTX Files** – One for each configuration:
 - `Config_count_down_shift_left.bit`
 - `Config_count_down_shift_left.ltx`
 - `Config_count_up_shift_right.bit`
 - `Config_count_up_shift_right.ltx`
- **Partial BIT Files** – The partial bit files must be loaded immediately after clearing the bit file. The partial BIT files are listed here:
 - `Config_count_down_shift_left_pblock_inst_count_partial.bit`
 - `Config_count_down_shift_left_pblock_inst_shift_partial.bit`
 - `Config_count_up_shift_right_pblock_inst_count_partial.bit`
 - `Config_count_up_shift_right_pblock_inst_shift_partial.bit`
- **Clearing BIT Files** – The clearing bitstream prepares the device for any subsequent partial bitstream for the Reconfigurable Partition by establishing a global mask signal for the region to be reconfigured. Each clearing bitstream is built for a specific RM, to be applied to clear the existing reconfigurable module in the device, and it must be followed immediately with the programming of the partial bitstream.

The clearing BIT files are listed here:

- `Config_count_down_shift_left_pblock_inst_count_partial_clear.bit`
- `Config_count_down_shift_left_pblock_inst_shift_partial_clear.bit`
- `Config_count_up_shift_right_pblock_inst_count_partial_clear.bit`
- `Config_count_up_shift_right_pblock_inst_shift_partial_clear.bit`
- **Important Programming Note** – This reference design has two RPs. Each RP has two RM functions. The counter RP either counts up or down. The shift RP either shifts left or right. Starting with the count up and shift right full bitstream, the following steps must be followed when programming this reference design:
 - a. Program the full bitstream and debug probe files:
 - `Config_count_up_shift_right.bit`
 - `Config_count_up_shift_right.ltx`
 - b. Clear the count RP by programming the `Config_count_up_shift_right_pblock_inst_count_partial_clear.bit`.
 - c. Program the count RM partial bit file `Config_count_down_shift_left_pblock_inst_count_partial.bit`.
 - d. Clear the shift RP by programming the `Config_count_up_shift_right_pblock_inst_shift_partial_clear.bit`.
 - e. Program the shift RM partial bit file `Config_count_down_shift_left_pblock_inst_shift_partial.bit`.

Note: [step b](#) and [step c](#) can be swapped with [step d](#) and [step e](#) respectively, but these pairs must remain linked (`partial_clear` followed by `partial`).

Alternatively, if choosing to start with the count down and shift left bitstream, the following steps must be followed when programming this reference design:

- a. Program the full bitstream and debug probe files:
 - `Config_count_down_shift_left.bit`
 - `Config_count_down_shift_left.ltx`
 - b. Clear the count RP by programming the `Config_count_down_shift_left_pblock_inst_count_partial_clear.bit`.
 - c. Program the count RM partial bit file `Config_count_up_shift_right_pblock_inst_count_partial.bit`.
 - d. Clear the shift RP by programming the `Config_count_down_shift_left_pblock_inst_shift_partial_clear.bit`.
 - e. Program the shift RM partial bit file `Config_count_up_shift_right_pblock_inst_shift_partial.bit`.
- Note:** [step b](#) and [step c](#) can be swapped with [step d](#) and [step e](#) respectively, but these pairs must remain linked (`partial_clear` followed by `partial`).

Checkpoint

This folder contains the saved routed design checkpoint (DCP) file for the static design (non-reconfigurable portion of the design). This DCP file is a copy of the original file in the Implement directory.

```
pr_example_static.dcp
```

Implement

The directory contains all of the design checkpoint files and reports that were created during implementation when the `run.tcl` file was executed. There is a directory for each of the full configurations: `Config_count_down_shift_left` and `Config_count_up_shift_right`. A checkpoint file was saved for the routed and static design.

Sources

This directory contains the `ip`, `hdl`, and `xdc` directories. The `ip` directory contains all of the IP catalog generated IP and example design.

The `hdl` directory contains the source files for the count up, count down, shift left, and shift right modules. It also contains the reference design top-level HDL file – `pr_example.v`. As mentioned previously, this is the modified SEM IP example design file. It was modified to add PR capabilities to this reference design.

The `xdc` directory contains the final constraints file for this reference design. Again, the SEM IP example design constraint file was modified to define the PR Pblocks in addition to the I/O constraints.

Synth

This directory contains the synthesis report files and design checkpoint files for the count up, count down, shift left, shift right, and `pr_example` modules generated when the `run.tcl` file was executed.

`pr_sem_jtag_KCU105_KU040.log`

This file is the Tera Term output log for the SEM IP controller. It contains the full UART log of running this reference design.

`pr_sem_jtag_KCU105_KU040_annotated.log`

This file is the Tera Term output log for the SEM IP controller. It is the same file as previous, but it has been annotated to illustrate where the different steps of running this reference design were performed.

pr_verify.log

Resulting file after the `pr_verify` command is executed in `run.tcl`. This command validates the consistency between the complete routed configurations. It compares the following files to determine if they are compatible:

- `Implementation/Config_count_up_shift_right/pr_example_route_design.dcp`
- `Implementation/Config_count_down_shift_left/pr_example_route_design.dcp`

readme.txt

This file provides a more complete breakdown of the directory structure and files used in the reference design.

run.tcl

This Tcl file is used by the Vivado Design Suite to regenerate the outputs of the reference design if desired.

Licensing

Partial Reconfiguration Licensing

Partial Reconfiguration is available as a licensed product within the Vivado Design Suite. Contact your [local sales offices](#) for pricing and ordering details.

A Partial Reconfiguration license is checked when the individual implementation steps (`place_design`, `route_design`, etc.) are run. A license is required to implement (place and route) a Partial Reconfiguration design and to generate partial bitstreams.

SEM IP Licensing

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado Design Suite under the terms of the [Xilinx End User License](#).

Information about this and other Xilinx LogiCORE IP modules is available at the [Xilinx Intellectual Property](#) page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your [local Xilinx sales representative](#).

Reference Design Steps

Tera Term Setup

- Baud: 115200
- Settings: Data 8; Parity None; Stop 1
- Flow Control: None
- Terminal ID: VT100
- New-line Transmit: CR (Terminal transmits CR as end of line)
- New-line Receive: CR + LF (Terminal receives CR as end of line and expands to CR + LF)
- Local Echo: NO

Running the Reference Design

This section provides a guide for downloading full and partial BIT files through JTAG to the KCU105 board. It also shows how to interact with the SEM IP controller through a serial UART using Tera Term software.

Unzip the Reference Design

1. Unzip the `ready_to_download.zip` and `xapp1261-demo-sem-pr.zip`. Then, see the `readme.txt` file in the `pr_sem_jtag_KCU105_KU040` folder.

Board Setup

2. Connect the Power, USB UART cable, and JTAG Programmer to the board, then apply power.



IMPORTANT: *Ensure the KCU105 board is powered on, the USB UART cable is connected, the JTAG cable is connected, and the JTAG cable drivers have been installed.*

UART Driver Setup and COM Port Selection

3. To setup the UART driver and COM port selection, open Tera Term and set up a serial connection with settings shown in the [Tera Term Setup](#).



RECOMMENDED: *The Silicon Labs USB UART drivers might need to be installed to get this connection to work for the first time. For more information, see the Silicon Labs CP210x USB-to-UART Installation Guide (UG1033) [Ref 5].*

4. In Tera Term, select the **COM XX** port that is associated with the **Silicon Labs Dual CP210x USB to UART Bridge: Standard COM Port (COM XX)** in the **Device Manager**; where **COM XX** is dependent on your computer.

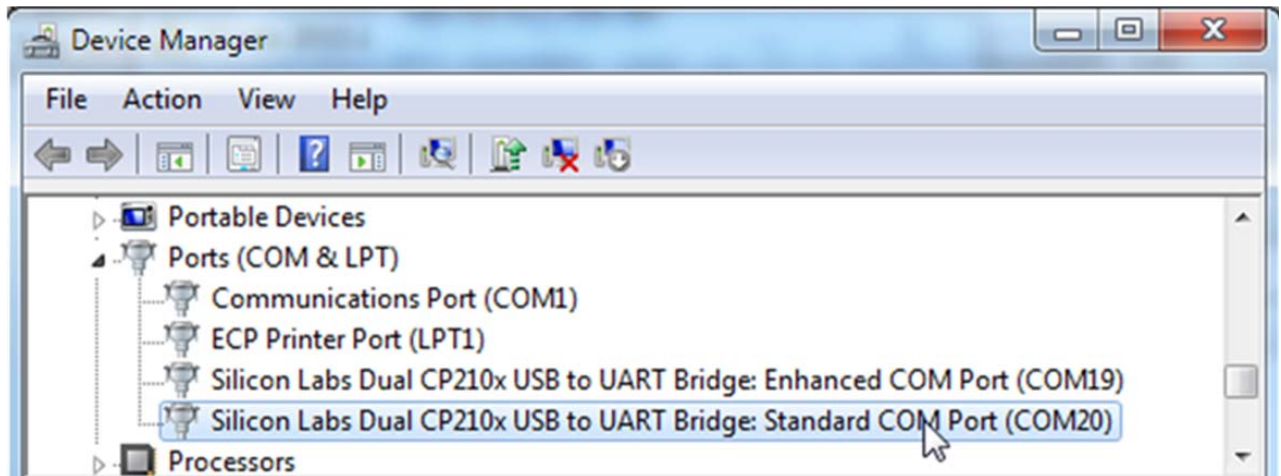


Figure 17: Device Manager

Program the Full BIT File – Config_count_up_shift_right

5. To program the full BIT file, `Config_count_up_shift_right`, open Vivado Design Suite in the GUI mode (no project or design needs to be opened, just the GUI).
6. Open the **Hardware Manager** by clicking the button, or from the **Menu Flow > Open Hardware Manager**.
7. Click the **Open target** link in the status bar or click **Tools > Auto Connect**.

8. In the **Hardware** window, right-click the **xcku040_0** and choose **Program Device...**

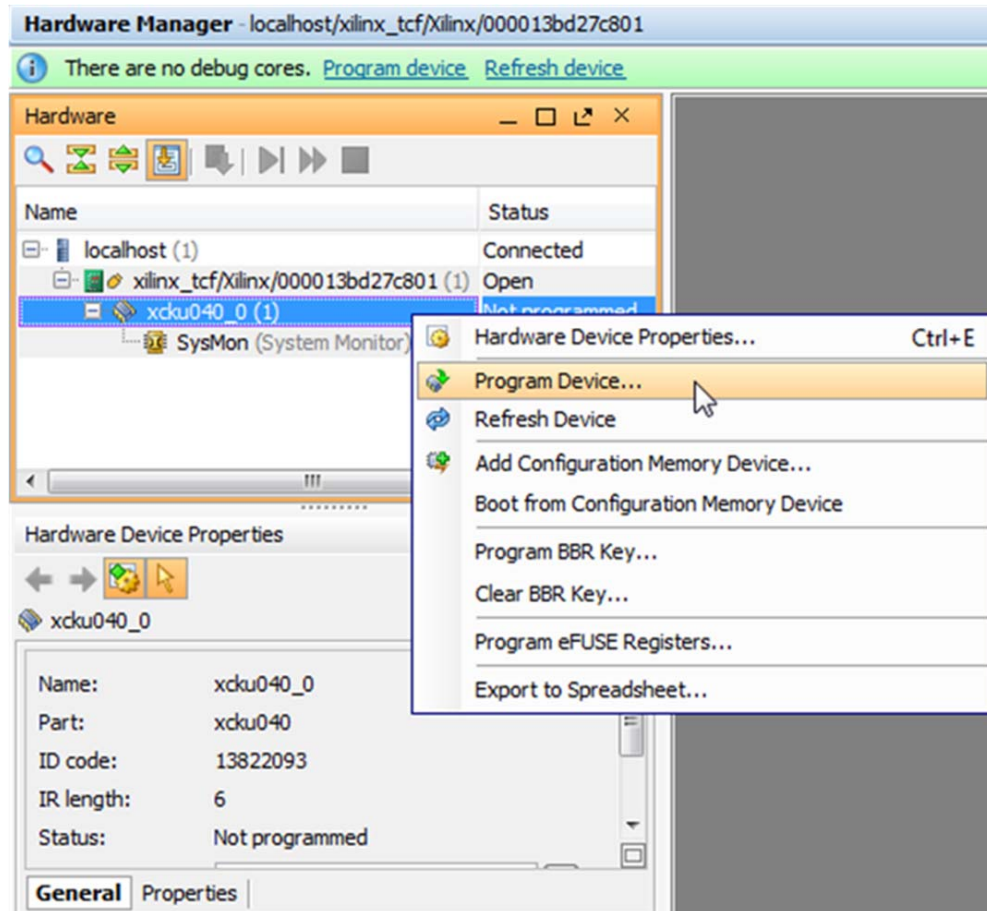


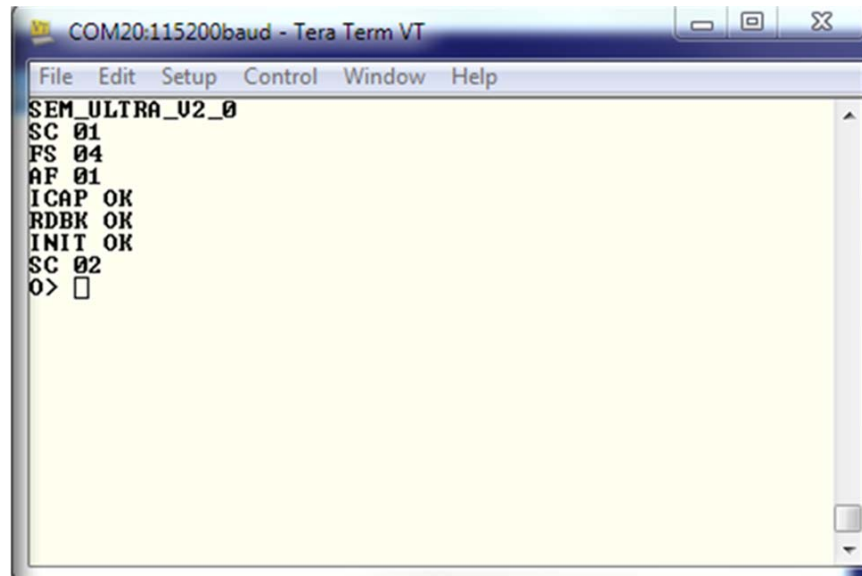
Figure 18: Hardware Manager Connected to KCU105

9. Using the **Program Device** dialog box:
- Browse to the demo design location, find the file `./Bitstreams/Config_count_up_shift_right.bit` and assign it as the Bitstream file.
 - Browse to the demo design location, find the file `./Bitstreams/Config_count_up_shift_right.ltx` and assign it as the Debug probes file.
10. Click **Program** to start the program process through JTAG.

Observe LEDs and SEM IP Serial Connection

11. Observe the eight GPIO LEDs at the top-right of the board. They will all light up during programming, and then should start toggling after programming is complete. The DONE LED will remain OFF until programming completes (the DONE LED is the second LED located to the right of the GPIO LEDs).
- DONE LED is ON.
 - The four left-most LEDs should be shifting the active LED from left-to-right.

- The four right-most LEDs should count up from 0x0 to 0xF.
12. [Figure 19](#) shows the SEM IP serial connection which indicates it has successfully completed initialization and is in the Observation state where it is scanning for errors.



```

COM20:115200baud - Tera Term VT
File Edit Setup Control Window Help
SEM_ULTRA_U2_0
SC 01
FS 04
AF 01
ICAP OK
RDBK OK
INIT OK
SC 02
0> █

```

Figure 19: Established UART Connection

13. Command the SEM IP to stop scanning for errors by putting it into the Idle state. Type an uppercase "I" at the Tera Term prompt. The prompt should change from "0>" to "I," indicating the IP is in Idle state.



IMPORTANT: All input to the terminal must be in uppercase letters.

Program the Count Up Clear BIT File

14. To program the count up clear BIT file, use the same procedure as in [step 8](#) through [step 10](#). Clear the partial reconfiguration and the count RM of the device with the following files. For UltraScale devices, it is required to clear an existing RM before loading a new RM.
- Browse to the demo design location, find the file `Config_count_up_shift_right_pblock_inst_count_partial_clear.bit` and assign it as the Bitstream file.
 - Browse to the demo design location, find the file `./Bitstreams/Config_count_up_shift_right.ltx` and assign it as the Debug probes file.
15. Observe the LED behavior:
- DONE LED is OFF.
 - Count LEDs are stalled.
 - Shift LEDs are still shifting from left-to-right.

Program the Count Down Partial BIT File

16. To program the count down partial BIT file, use the same procedure as in [step 8](#) through [step 10](#). Partially reconfigure the device with the count down files as follows:
- Browse to the demo design location, find the file `Config_count_down_shift_left_pblock_inst_count_partial.bit` and assign it as the Bitstream file.
 - Browse to the demo design location, find the file `./Bitstreams/Config_count_up_shift_right.ltx` and assign it as the Debug probes file.
17. This reconfiguration is very fast as this is only reconfiguring a small part of the device. When the partial reconfiguration completes, proceed to the next step.

Observe LEDs and Perform Soft SEM IP Reset after Count PR

18. Observe LED behavior and perform the SEM IP soft reset:
- DONE LED is ON.
 - Shift LEDs are still shifting from left-to-right.
 - Count LEDs are now counting down from 0xF to 0x0.
 - Reset SEM IP by typing "R 00" into the terminal.



IMPORTANT: Note that there is a space between the "R" and two zeros. It might take a few moments to complete this operation.

- SEM IP re-initializes to include the modified RM.



```

I> R 00
SEM_ULTRA_U2_0
SC 01
FS 04
AF 01
ICAP OK
RDBK OK
INIT OK
SC 02
O> 

```

Figure 20: SEM IP Re-Initialization with Count Down RM

19. Observe that the behavior of the LEDs have not changed. The left (shift) LEDs are shifting from left-to-right, and the right (count) LEDs are counting down. Also, observe SEM IP has gone back into Observation mode where it is scanning for errors.

Program the Shift Right Clear BIT File

20. To program the shift right clear BIT file, put SEM IP into Idle by typing an uppercase "I" at the Tera Term prompt. The prompt should change from "O>" to "I," indicating the IP is in Idle state.

21. Using the same procedure as in [step 8](#) through [step 10](#), clear the partial reconfiguration in the shift RM of the device with the following files. For UltraScale devices, it is required to clear an existing RM before loading a new RM.
 - Browse to the demo design location, find the file `Config_count_up_shift_right_pblock_inst_shift_partial_clear.bit` and assign it as the Bitstream file.
 - Browse to the demo design location, find the file `./Bitstreams/Config_count_up_shift_right.ltx` and assign it as the Debug probes file.
22. Observe LED behavior:
 - DONE LED is OFF.
 - Count LEDs are still counting down from 0xF to 0x0.
 - Shift LEDs are stalled.

Program the Shift Left Partial BIT File

23. To program the shift-left partial BIT file, use the same procedure as in [step 8](#) through [step 10](#). Partial reconfigure the device with the following files. This causes the left (shift) LEDs to change direction and now shift from right-to-left. The count LEDs continue to count down.
 - Browse the demo design location, find the file `./Bitstreams/Config_count_down_shift_left_pblock_inst_shift_partial.bit` and assign it as the Bitstream file.
 - Browse the demo design location, find the file `./Bitstreams/Config_count_up_shift_right.ltx` and assign it as the Debug probes file.

This reconfiguration is very fast as this is only reconfiguring a small part of the device. When the partial reconfiguration completes, proceed to the next step.

Observe LEDs and Perform Soft SEM IP Reset after Shift PR

24. Observe the LED behavior and perform SEM IP soft reset:
 - DONE LED is ON.
 - Shift LEDs are now shifting from right-to-left.
 - Count LEDs are still counting down from 0xF to 0x0.
 - Reset SEM IP by typing "R 00" into the terminal.



IMPORTANT: Note that there is a space between the "R" and two zeros. It might take a few moments to complete this operation.

- SEM IP re-initializes to include the modified RM.

```

I> R 00
SEM_ULTRA_U2_0
SC 01
FS 04
AF 01
ICAP OK
RDBK OK
INIT OK
SC 02
O> 

```

Figure 21: SEM IP Re-Initialization with Shift Left RM

25. Observe that the behavior of the LEDs have not changed. The left (shift) LEDs are shifting from right-to-left, and the right (count) LEDs are counting down. Also, observe SEM IP has gone back into Observation mode where it is scanning for errors.

Injecting Errors in the Reference Design

SEM IP can inject errors at specific locations through the UART connection. The error injections performed below are by Linear Frame Address (LFA). This process is a simple read-modify-write to invert one configuration memory bit at an address specified as part of the error injection command.

SEM IP must stop scanning for errors before injecting an error, so it is commanded to transition to the Idle state. While in the Idle state, errors can be injected into the configuration memory one bit at a time. After injecting errors, SEM IP is commanded to transition to the Observation state. In the Observation state, SEM IP begins scanning for errors in the configuration memory. When an error is detected, SEM IP corrects the error and generates a report through the UART interface. The report contains the address in which the error occurred. For more information on SEM IP commands, reports, and behavior, see the *UltraScale Architecture Soft Error Mitigation Controller Product Guide* (PG187) [Ref 1].



CAUTION! *If any changes are made to the reference design that cause the implementation results to change, the injections below could have different behavior. This is expected because error injection addresses change anytime the implementation results change. A non-inclusive list of changes that could cause this to occur are listed here:*

- *Using a different version of Vivado*
- *Targeting a different family architecture*
- *Targeting a different device*
- *Changing the RTL*
- *Modifying any of the Pblocks*

26. Put SEM IP into Idle by typing "I" at the Tera Term prompt.

27. Inject errors into the static region by typing the following commands into the Tera Term prompt:

```

N C0027562D5
N C004A36C61
N C0025B3123

```

28. Note that the LEDs continue to operate as expected.
29. Place SEM IP back into the Observation mode by typing "O." Observe the Tera Term log file for the SEM IP corrections that occurred.
30. Put SEM IP back into Idle and inject errors in PR region using the specified commands:

```
N C0035DD000  
N C00360C109  
N C0036238F3
```
31. Place SEM IP back into the Observation mode by typing "O." Observe the Tera Term log file for the SEM IP corrections that occurred.
32. Put SEM IP back into Idle and inject an error into the critical clock connection in the count PR region using the specified command.

```
N C00359A7D5
```
33. Observe the count LEDs stall.
34. Place SEM IP back into the Observation mode by typing "O." Observe the Tera Term log file for the SEM IP corrections that occurred.
35. Observe the count LEDs resume counting again.
36. Put SEM IP back into Idle and inject an error into the critical clock connection of the shift PR region using the specified command.

```
N C0034FE7C5
```
37. Observe the shift LEDs stall.
38. Place SEM IP back into the Observation mode by typing "O." Observe the Tera Term log file for the SEM IP corrections that occurred.
39. Observe the shift LEDs resume shifting again.

Conclusion

This reference design has shown that PR and SEM IP can operate together. The SEM IP example design was modified to include Reconfigurable Partitions then the reference design was synthesized bottom-up to allow for Partial Reconfiguration. The GPIO LEDs were used to confirm PR occurred and Tera Term was used to ensure SEM IP continued to detect errors.

There are limitations and considerations to account for when implementing these two features together in a real design. See the [Limitations and Considerations](#) section that follows.

Also, see the *UltraScale Architecture Soft Error Mitigation Controller Product Guide* (PG187) [\[Ref 1\]](#) to gain a better understanding of SEM IP and its requirements. Such as putting SEM IP into Idle prior to performing partial reconfiguration, and resetting SEM IP after PR completion.

Limitations and Considerations

7 Series Devices

INIT LED

With 7 series devices, the SEM controller does not stop scanning for errors while in the Idle state. On a board such as the KC705, after SEM IP is commanded to the Idle state and after PR is performed, SEM IP detects an error which causes the INIT LED to illuminate RED. This is the expected behavior. SEM IP must be commanded to perform a reset and recalculate the expected frame values. The INIT LED illuminates GREEN after reset completes.

XDC

- **SNAPPING_MODE Property** – This property must be explicitly set in `pr_example.xdc`. It automatically creates legal, reconfigurable Pblocks. Set each respective property after its Pblock constraints.

```
set_property SNAPPING_MODE ON [get_pblocks pblock_inst_count]
set_property SNAPPING_MODE ON [get_pblocks pblock_inst_shift]
```

Partial Reconfiguration and SEM IP

1. Partial Reconfiguration is not limited to the JTAG interface. It can also be performed through ICAP, MCAP, or through other external configuration ports. Consult the *Dual Use of ICAP with SEM Controller Application Note* (XAPP517) [Ref 6] for instructions on how to share the ICAP between SEM and another function (such as a PR controller).
2. A PR event should complete before another one is issued.
 - a. Assume design operation is a repeated cycle of:
 - **SEMOBSV** – Soft Error Mitigation observation time (variable, determined by you)
 - **PRLOAD** – Partial bitstream load (determined by bitstream size, configuration data width, and configuration clock frequency)
 - **SEMINIT** – Soft Error Mitigation initialization time (see the *UltraScale Architecture Soft Error Mitigation Controller Product Guide* (PG187) [Ref 1], SEM start-up latency table, "Initialization Time")
 - b. Wait at least (PRLOAD + SEMINIT) before starting another PR event. However in this case, SEMOBSV is zero and SEM IP provides no protection against soft errors.
 - c. Approximate SEU coverage in time is $SEMOBSV / (SEMOBSV + PRLOAD + SEMINIT)$. Xilinx recommends maintaining as high coverage as possible (for example, > 99%).
3. Any time PR is performed, it reduces soft error coverage; therefore, the frequency in which PR is performed must be considered as part of the SEU mitigation calculation.

4. Both PR and SEM IP require access to the configuration engine to control monitoring and writing of the configuration memory. Both cannot access to the configuration memory at the same time.
5. SEM IP and PR is only supported with monolithic devices. SEM IP and PR is not supported with SSI devices.

SEM IP

For the latest information, see the *UltraScale Architecture Soft Error Mitigation Controller Product Guide* (PG187) [\[Ref 1\]](#).

1. The SEM controller does not operate on soft errors in block memory, distributed memory, or flip-flops. Soft error mitigation in these memory resources must be addressed by the user logic through preventive measures such as redundancy or error detection and correction codes.
2. The SEM controller initializes and manages the FPGA integrated silicon features for soft error mitigation and when included in a design, does not include any design constraints or options that would enable the built-in detection functions. For example, do not set POST_CRC, POST_CONFIG_CRC, or any other related constraints. Similarly, do not include options to modify GLUTMASK.
3. Software computed ECC and CRC values are not supported.
4. Design simulations that instantiate the SEM controller are supported. However, it is not possible to observe the controller behaviors in simulation. Design simulation including the controller compiles, but the controller does not exit the Initialization state. Hardware-based evaluation of the controller behaviors is required.
5. Use of bitstream security (encryption and authentication) is currently not supported by the controller.
6. Use of SelectMAP persistence is not supported by the controller.
7. Only a single ICAP instance is supported for each SEM controller and it must reside at the primary/top physical location.

Partial Reconfiguration

See Chapter 4 in the *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909) [\[Ref 3\]](#).

Debugging

SEM IP

For SEM IP debugging, see the “Debugging” appendix in the *UltraScale Architecture Soft Error Mitigation Controller Product Guide* (PG187) [\[Ref 1\]](#).

Partial Reconfiguration

When implementation and bitstream generation steps (`opt_design`, `route_design`, `write_bitstream`, etc.) encounter the `HD.RECONFIGURABLE` property, the Vivado Design Suite checks for a Partial Reconfiguration license. If a valid license is available, the Vivado Design Suite generates the following status:

```
Feature available: PartialReconfiguration
```

If this status is not received, a Partial Reconfiguration license is not available. Contact your local Xilinx sales office for more information. Evaluation licenses are also available.

Running the Reference Design

The INIT LED should never illuminate RED when running this reference design on UltraScale devices. With 7 series devices, there is one case this could occur and it is covered in [Limitations and Considerations](#).

When running this UltraScale reference design, the INIT LED could illuminate RED due to the following:

1. SEM IP was not put in Idle before PR was performed. In this case, SEM IP detects an error occurred.
2. SEM IP was put in Idle before PR was performed. However, after PR was performed and before issuing the reset, SEM IP was commanded to go to the Observation state.

For other reasons, contact [Xilinx Support](#).

References

1. *UltraScale Architecture Soft Error Mitigation Controller Product Guide* ([PG187](#))
2. *Vivado Design Suite User Guide: Designing with IP* ([UG896](#))
3. *Vivado Design Suite User Guide: Partial Reconfiguration* ([UG909](#))
4. *FIFO Generator Product Guide* ([PG057](#))
5. *Silicon Labs CP210x USB-to-UART Installation Guide* ([UG1033](#))
6. *Dual Use of ICAP with SEM Controller Application Note* ([XAPP517](#))
7. *Vivado Design Suite Tutorial: Partial Reconfiguration* ([UG947](#))

Revision History

The following table shows the revision history for this document.

Date	Version	Changes
06/19/2015	1.0	Initial Xilinx release.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at <http://www.xilinx.com/legal.htm#tos>.

© Copyright 2015 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.