Vivado Design Suite Tutorial

Partial Reconfiguration on Zed Board

*This document is based on the Xilinx document UG947: Vivado Design Suite Tutorial on Partial Reconfiguration
# Table of Contents

1. Objectives .................................................................................................................. 2
2. Vivado Partial Reconfiguration - Documentation ......................................................... 2
3. Tutorial .......................................................................................................................... 2

3.1 Led Shift Count ........................................................................................................ 2
   3.1.1 Extract the Tutorial Design files .......................................................................... 2
   3.1.2 Synthesize the Design ......................................................................................... 2
   3.1.3 Assemble the Design ............................................................................................ 3
   3.1.4 Build the Design Floorplan .................................................................................. 6
   3.1.5 Implement the First Configuration ....................................................................... 8
   3.1.6 Implement the Second Configuration ................................................................... 10
   3.1.7 Generate Bitstreams ............................................................................................ 10
   3.1.8 Partial Reconfiguration of the FPGA .................................................................. 11
1 Objectives

- Implement a project that can be dynamically reconfigured using the Zed Board.
- Learn the Partial Reconfiguration (PR) flow with the Vivado TCL console.

2 Vivado Partial Reconfiguration - Documentation

- UG947: Vivado Design Suite Tutorial – Partial Reconfiguration. You can follow this for the Xilinx-provided ug947-vivado-partial-reconfiguration-tutorial.zip file (this is a Verilog design for the KC705 demonstration board).

3 Tutorial

3.1 Led Shift Count

3.1.1 Extract the Tutorial Design files

- Extract the zip file contents from Dynamic_PR_Tutorial to any write-accessible location.

3.1.2 Synthesize the Design

- Open the Vivado TCL Shell. Navigate to the /led_shift_count directory.
- Run the design.tcl script by entering: source design.tcl -notrace. This will Synthesize the design and create output files in the /Synthesis folder. The 'top' design will be created with a blank circuit for the Reconfigurable Partition.

```
Vivado% cd E:/Grad_Project/led_shift_count
Vivado% source design.tcl -notrace
```

Fig 1. Vivado TCL Shell
3.1.3 Assemble the Design

- Open the Vivado IDE by entering `start_gui` in Vivado TCL Shell.

![Vivado TCL Shell after sourcing the design.tcl file]

---

Partial Reconfiguration on Zed Board
- Load the static design by issuing the following command in the Tcl Console:

  ```tcl
  open_checkpoint Synth/Static/top_synth.dcp
  ```

**Fig 3. Vivado TCL Console**

**Fig 4. Vivado after opening the Checkpoint**
- You can see the design structure in the Netlist pane, but black boxes exist for the inst_shift and inst_count modules. Note that the Flow Navigator pane is not present. You are working in non-project mode.
- Two critical warnings are issued regarding unmatched instances. These instances are the Reconfigurable Modules that have yet to be loaded, and you can therefore ignore these warnings safely.

- Load the synthesized checkpoints for first Reconfigurable Module variants for each of reconfigurable partitions:

  read_checkpoint -cell inst_shift Synth/shift_right/shift_synth.dcp

  read_checkpoint -cell inst_count Synth/count_up/count_synth.dcp

- Define each of these submodules as partially reconfigurable by setting the HD.RECONFIGURABLE property:

  set_property HD.RECONFIGURABLE 1 [get_cells inst_shift]

  set_property HD.RECONFIGURABLE 1 [get_cells inst_count]

- Save the assembled design state for this initial configuration:

  write_checkpoint ./Checkpoint/top_link_right_up.dcp
3.1.4 **Build the Design Floorplan**

Here, you create a floorplan to define the regions that will be partially reconfigured.

- Select the inst_count instance in the Netlist pane. Right click and select: **Floorplanning > Draw Pblock** and draw a tall narrow box. The exact size and shape do not matter at this point, but keep the box within the clock region.

![Fig 6. Draw Pblock for inst_count](image)

- In the Properties pane, select the checkbox for **RESET_AFTER_RECONFIG**. This will utilize the dedicated initialization of the logic in this module after reconfiguration has completed.

![Fig 7. Set Reset after Reconfiguration](image)
- Repeat the above steps for inst_shift instance.

Fig 8. Draw Pblock for inst_shift

- Run PR Design Rule Checks by selecting Tools > Report > Report DRC. You can uncheck All Rules and then check Partial Reconfiguration to focus this report strictly on PR DRCs.

Fig 9. Report DRC
To avoid the DRC warning automatically by setting the SNAPPING_MODE feature which automatically adjusts the size and shape of reconfigurable Pblocks to align with legal boundaries. It will make the Pblock taller, aligning with clock region boundaries, if the RESET_AFTER_RECONFIG feature is selected. It will make the Pblock narrower, adjusting left and/or right edges as needed. Note that the number and type of resources available will be altered if SNAPPING_MODE makes changes to the Pblock.

Fig 10. Set Snapping mode

- Select the Pblock for inst_count, and in the Properties tab of the Pblock Properties pane, change the value of SNAPPING_MODE from OFF to ROUTING (or ON). Repeat same for inst_shift instance. Then Run PR Design Check again.
- Save these Pblock definitions and its associated properties on a .xdc file:

  ```
  write_xdc ./Sources/xdc/fplan.xdc
  ```

### 3.1.5 Implement the First Configuration

- Load the top-level constraint file by issuing the command:

  ```
  read_xdc Sources/xdc/top_io.xdc
  ```

- Optimize, place, and route the design. Notice the Partition Pins (interface points between static and dynamic regions)

  ```
  opt_design
  place_design
  route_design
  ```
• Save the full design checkpoint and create report files:

```plaintext
write_checkpoint -force Implement/Config_shift_right_count_up/top_route_design.dcp
report_utilization -file Implement/Config_shift_right_count_up/top_utilization.rpt
report_timing_summary -file Implement/Config_shift_right_count_up/top_timing_summary.rpt
```

At this point, you can use the static portion of this configuration for all subsequent configurations (variants of the circuit with different RMs for each RP). We need to isolate the static design by removing the Reconfigurable Modules:

• Clear out Reconfigurable Module logic:

```plaintext
update_design -cell inst_shift -black_box
update_design -cell inst_count -black_box
```

Fig 11. Updated design

• Lock down all placement and routing. This is an important step to guarantee consistency for different RMs for each RP.

```plaintext
lock_design -level routing
```

• Write out the remaining static-only checkpoint (this checkpoint will be used for any future configurations).

```plaintext
write_checkpoint -force Checkpoint/static_route_design.dcp
```
3.1.6 Implement the Second Configuration

- With the locked static design open in memory, read in post-synthesis checkpoints for the other two Reconfigurable Modules.

  read_checkpoint -cell inst_shift Synth/shift_left/shift_synth.dcp
  read_checkpoint -cell inst_count Synth/count_down/count_synth.dcp

- Optimize, place, and route the design. Notice the Partition Pins (interface points between static and dynamic regions)

  opt_design
  place_design
  route_design

- Save the full design checkpoint and create report files:

  write_checkpoint –force Implement/Config_shift_left_count_down/top_route_design.dcp
  report_utilization -file Implement/Config_shift_left_count_down/top_utilization.rpt
  report_timing_summary -file Implement/Config_shift_left_count_down/top_timing_summary.rpt

- At this point, you have implemented the static design and all Reconfigurable Module variants. This process would be repeated for designs that have more than two Reconfigurable Modules per RP, or more RPs. Close the current design:

  close_project

3.1.7 Generate Bitstreams

- Run the pr_verify command from the Tcl Console:

  pr_verify Implement/Config_shift_right_count_up/top_route_design.dcp
  Implement/Config_shift_left_count_down/top_route_design.dcp

- Read the first configuration into memory:

  open_checkpoint Implement/Config_shift_right_count_up/top_route_design.dcp

- Generate full and partial bitstreams for this design.

  write_bitstream –force -file Bitstreams/Config_RightUp.bit
  close_project
• Notice the three bitstreams have been created:
  o Config_RightUp.bit - This is the power-up, full design bitstream.
  o Config_RightUp_pblock_inst_shift_partial.bit - This is the partial bit file for the shift_right module.
  o Config_RightUp_pblock_inst_count_partial.bit - This is the partial bit file for the count_up module.
• Read the Second configuration into memory:
  open_checkpoint Implement/Config_shift_left_count_down/top_route_design.dcp
• Generate full and partial bitstreams for this design.
  write_bitstream –force -file Bitstreams/Config_LeftDown.bit
  close_project
• Generate a full bitstream with a blackbox for the RP, plus blanking bitstreams for the RMs, these can be used to erase an existing configuration to reduce power consumption:
  open_checkpoint Checkpoint/static_route_design.dcp
  update_design -cell inst_count -buffer_ports
  update_design -cell inst_shift -buffer_ports
  place_design
  route_design
  write_checkpoint –force Checkpoint/Config_black_box.dcp
  write_bitstream –force -file Bitstreams/config_black_box.bit
  close_project

3.1.8 Partial Reconfiguration of the FPGA

• From the main Vivado IDE, select Flow>Open Hardware Manager.

---

Fig 12. Open Hardware Manager
• Select **Open Target > open new target** on the green banner. Follow the steps in the wizard to establish communication with the board.

![Vivado 2015.2](image)

**Fig 13. Open New Target**

• Select **Program device** on the green banner and pick the xc7z020_1. Navigate to the Bitstreams folder to select Config_RightUp.bit, then click **OK** to program the device.

• You should now see the bank of GPIO LEDs performing two tasks. Four LEDs are performing a counting-up function (MSB is on the left), and the other four are shifting to the right. Note the amount of time it took to configure the full device.

At this point, you can partially reconfigure the active device with any of the partial bitstreams that you have created.

• Select **Program device** on the green banner again. Navigate to the Bitstreams folder to select Config_LeftDown_pblock_inst_shift_partial.bit, then click **OK** to program the device.
  o The shift portion of the LEDs has changed direction, but the counter kept counting up, unaffected by the reconfiguration. Note the much shorter configuration time.

• Select **Program device** on the green banner again. Navigate to the Bitstreams folder to select Config_LeftDown_pblock_inst_count_partial.bit, then click **OK** to program the device.
  o The counter is now counting down, and the shifting LEDs were unaffected by the reconfiguration. This process can be repeated with the Config_RightUp partial bit files to return to the original configuration, or with the blanking partial bit files to stop activity on the LEDs (they will stay on).

*This document is based on the Xilinx document UG947: Vivado Design Suite Tutorial on Partial Reconfiguration*