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

- 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

- UG909: Vivado Design Suite User Guide – Partial Reconfiguration.
- 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

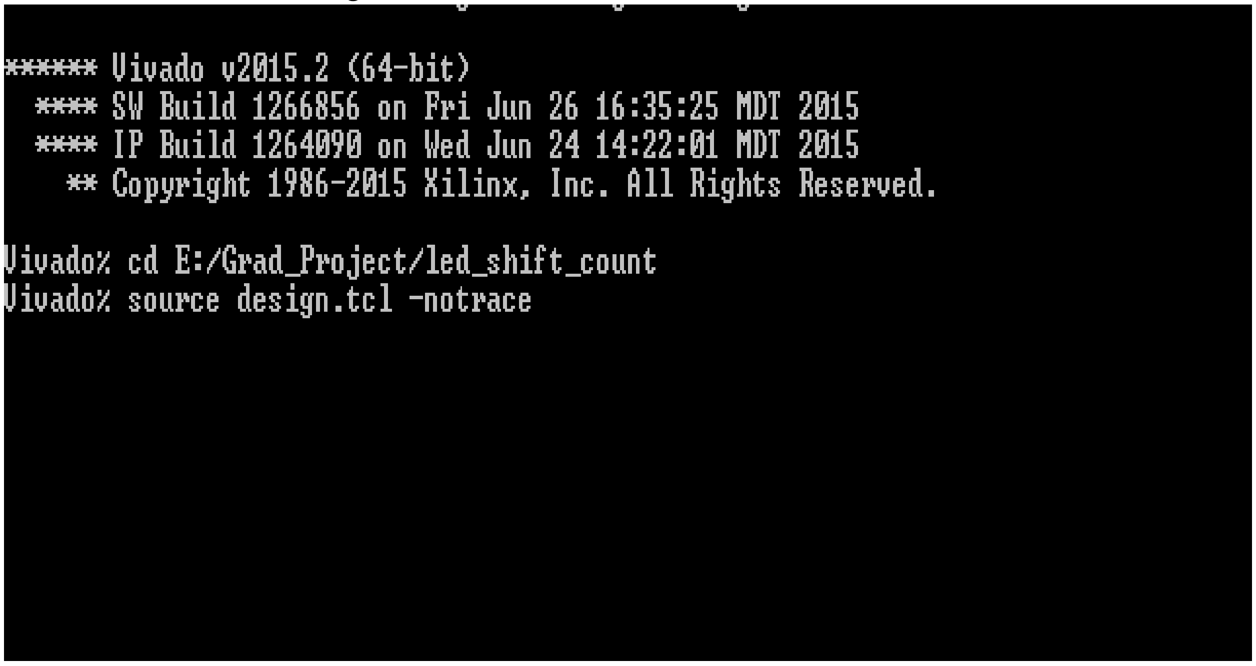
3.1 Led Shift Count

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.

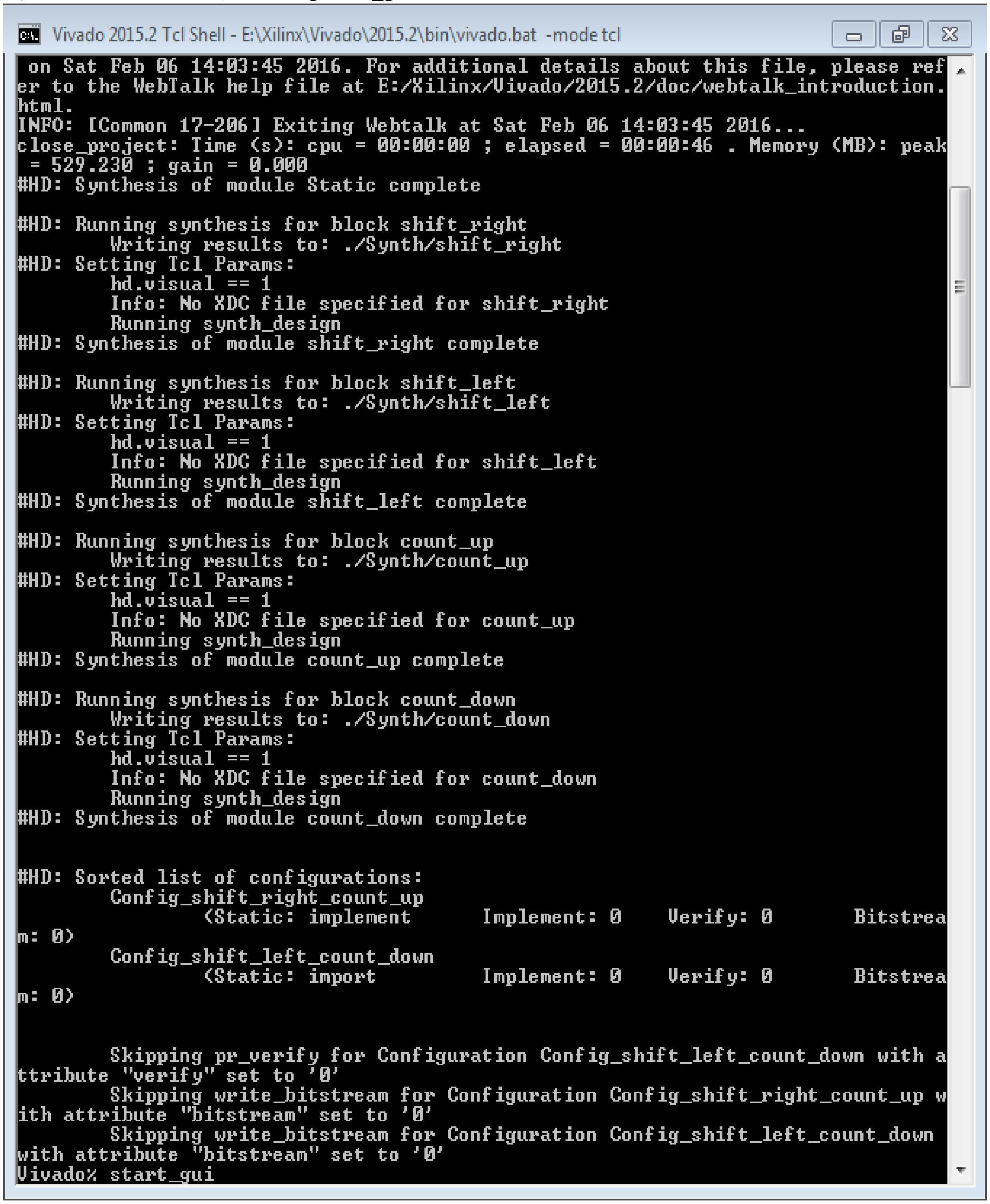


```
****** Vivado v2015.2 (64-bit)
  **** SW Build 1266856 on Fri Jun 26 16:35:25 MDT 2015
  **** IP Build 1264090 on Wed Jun 24 14:22:01 MDT 2015
    ** Copyright 1986-2015 Xilinx, Inc. All Rights Reserved.

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.

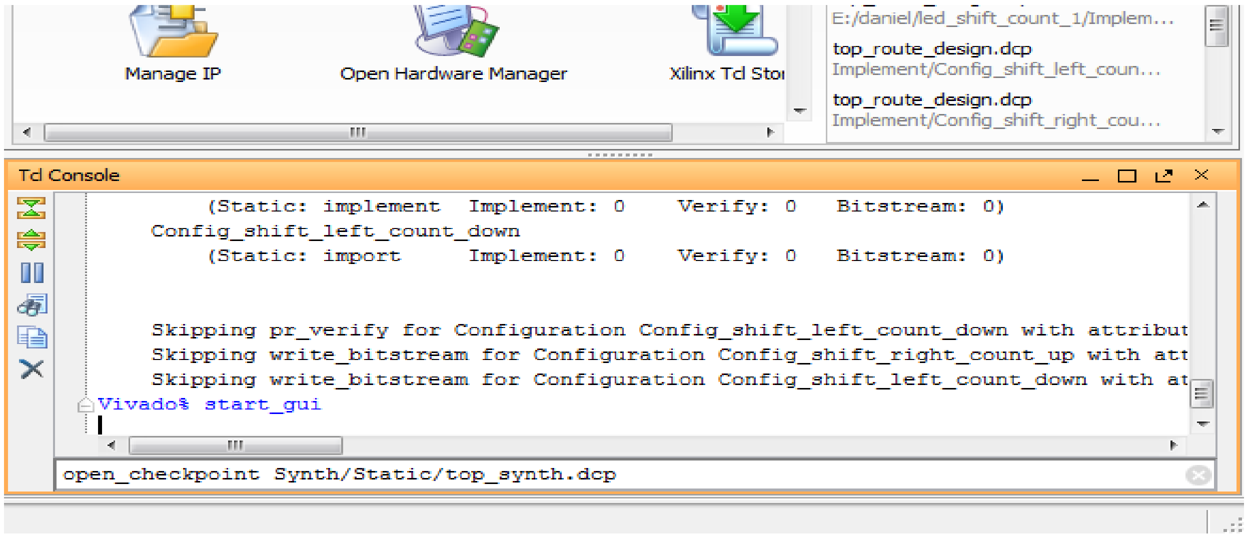

Vivado 2015.2 Tcl Shell - E:\Xilinx\Vivado\2015.2\bin\vivado.bat -mode tcl

```
 on Sat Feb 06 14:03:45 2016. For additional details about this file, please ref
er to the WebTalk help file at E:/Xilinx/Vivado/2015.2/doc/webtalk_introduction.
html.
INFO: [Common 17-206] Exiting Webtalk at Sat Feb 06 14:03:45 2016...
close_project: Time (s): cpu = 00:00:00 ; elapsed = 00:00:46 . Memory (MB): peak
 = 529.230 ; gain = 0.000
#HD: Synthesis of module Static complete

#HD: Running synthesis for block shift_right
        Writing results to: ./Synth/shift_right
#HD: Setting Tcl Params:
        hd.visual == 1
        Info: No XDC file specified for shift_right
        Running synth_design
#HD: Synthesis of module shift_right complete

#HD: Running synthesis for block shift_left
        Writing results to: ./Synth/shift_left
#HD: Setting Tcl Params:
        hd.visual == 1
        Info: No XDC file specified for shift_left
        Running synth_design
#HD: Synthesis of module shift_left complete

#HD: Running synthesis for block count_up
        Writing results to: ./Synth/count_up
#HD: Setting Tcl Params:
        hd.visual == 1
        Info: No XDC file specified for count_up
        Running synth_design
#HD: Synthesis of module count_up complete

#HD: Running synthesis for block count_down
        Writing results to: ./Synth/count_down
#HD: Setting Tcl Params:
        hd.visual == 1
        Info: No XDC file specified for count_down
        Running synth_design
#HD: Synthesis of module count_down complete


#HD: Sorted list of configurations:
        Config_shift_right_count_up
                (Static: implement      Implement: 0    Verify: 0       Bitstrea
m: 0)
        Config_shift_left_count_down
                (Static: import         Implement: 0    Verify: 0       Bitstrea
m: 0)


        Skipping pr_verify for Configuration Config_shift_left_count_down with a
ttribute "verify" set to '0'
        Skipping write_bitstream for Configuration Config_shift_right_count_up w
ith attribute "bitstream" set to '0'
        Skipping write_bitstream for Configuration Config_shift_left_count_down
with attribute "bitstream" set to '0'
Vivado% start_gui
```

Fig 2. Vivado TCL Shell after sourcing the design.tcl file

- Load the static design by issuing the following command in the Tcl Console:

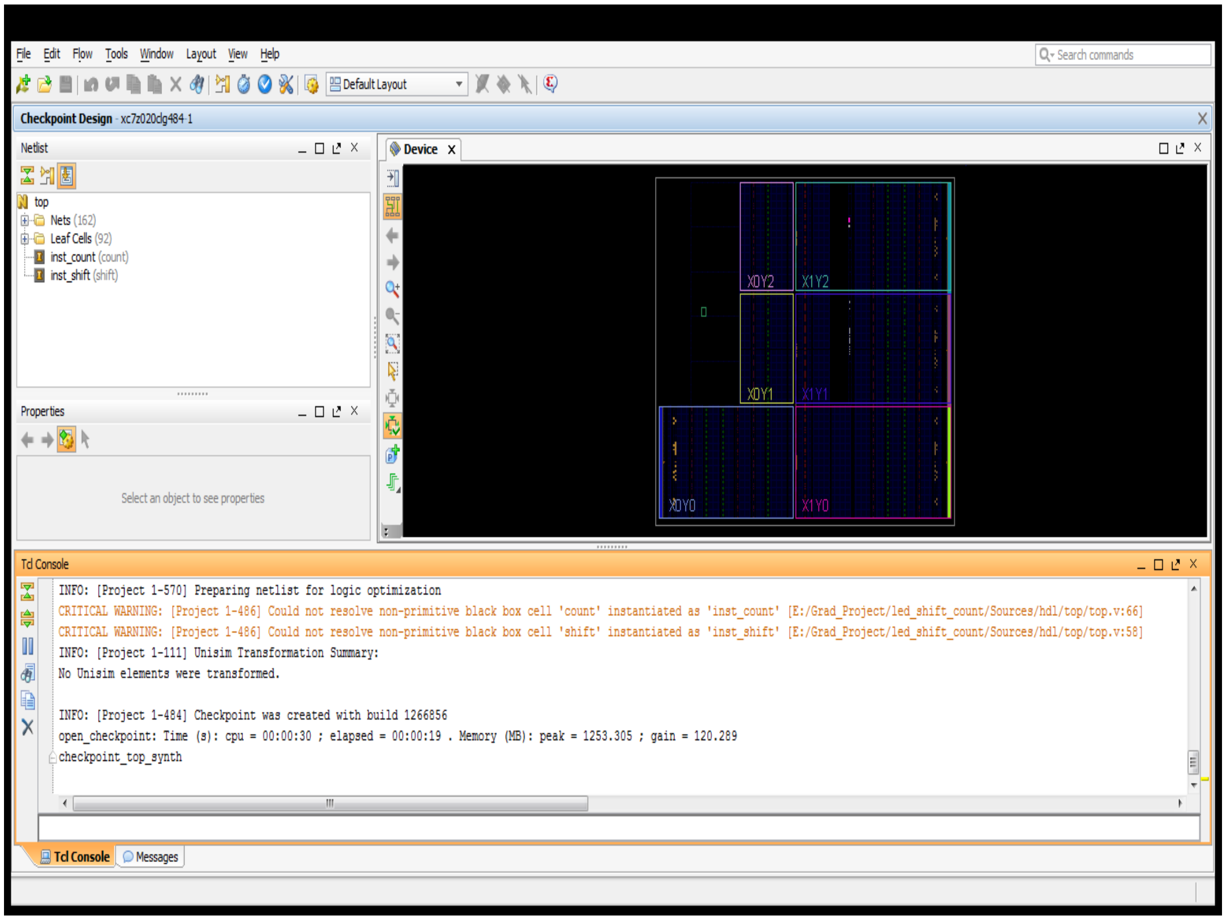
  open_checkpoint Synth/Static/top_synth.dcp


Manage IP
Open Hardware Manager
Xilinx Tcl Stor
E:/daniel/led_shift_count_1/Implem...
top_route_design.dcp
Implementation/Config_shift_left_coun...
top_route_design.dcp
Implementation/Config_shift_right_cou...
Tcl Console

```
        (Static: implement  Implement: 0    Verify: 0   Bitstream: 0)
    Config_shift_left_count_down
        (Static: import     Implement: 0    Verify: 0   Bitstream: 0)


    Skipping pr_verify for Configuration Config_shift_left_count_down with attribut
    Skipping write_bitstream for Configuration Config_shift_right_count_up with att
    Skipping write_bitstream for Configuration Config_shift_left_count_down with at
Vivado% start_gui
```

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

Fig 3. Vivado TCL Console



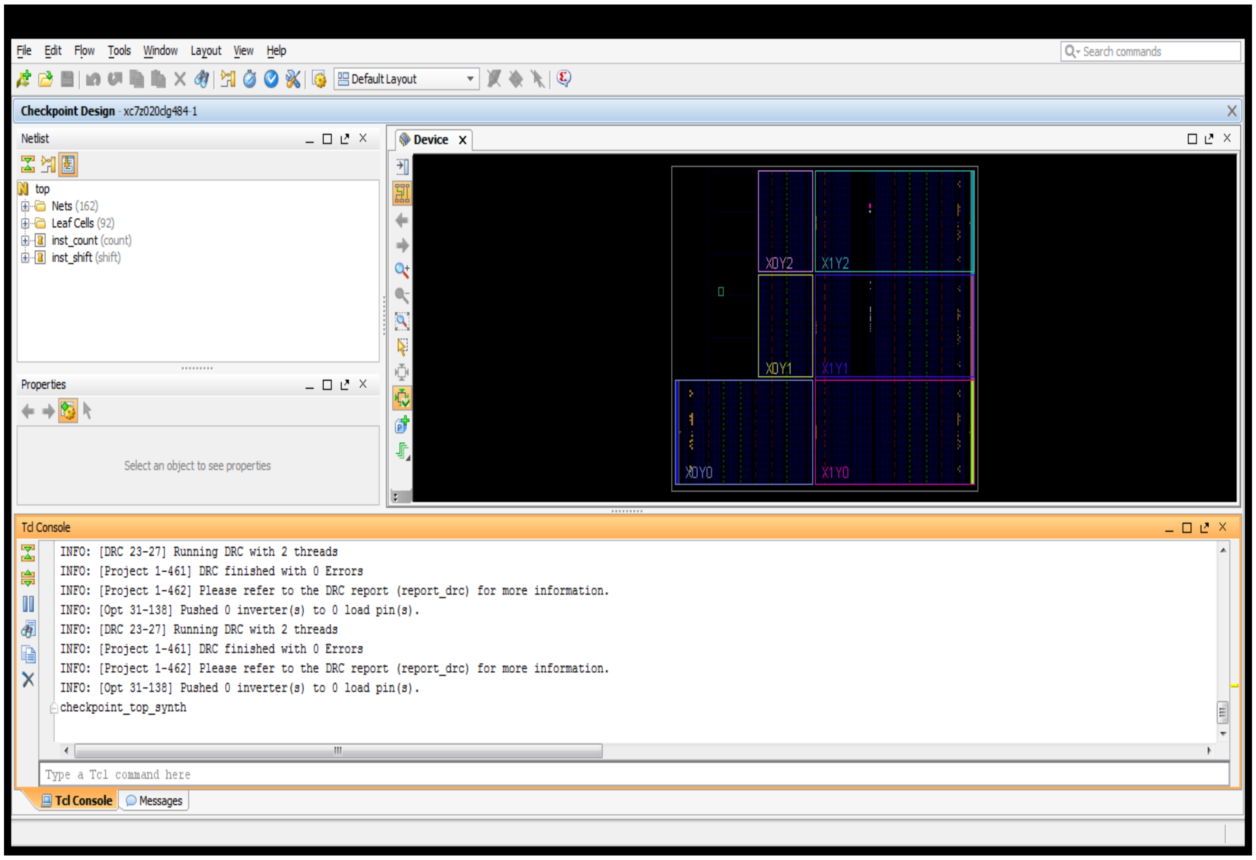
```
INFO: [Project 1-570] Preparing netlist for logic optimization
CRITICAL WARNING: [Project 1-486] Could not resolve non-primitive black box cell 'count' instantiated as 'inst_count' [E:/Grad_Project/led_shift_count/Sources/hdl/top/top.v:66]
CRITICAL WARNING: [Project 1-486] Could not resolve non-primitive black box cell 'shift' instantiated as 'inst_shift' [E:/Grad_Project/led_shift_count/Sources/hdl/top/top.v:58]
INFO: [Project 1-111] Unisim Transformation Summary:
No Unisim elements were transformed.

INFO: [Project 1-484] Checkpoint was created with build 1266856
open_checkpoint: Time (s): cpu = 00:00:30 ; elapsed = 00:00:19 . Memory (MB): peak = 1253.305 ; gain = 120.289
checkpoint_top_synth
```

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



```
INFO: [DRC 23-27] Running DRC with 2 threads
INFO: [Project 1-461] DRC finished with 0 Errors
INFO: [Project 1-462] Please refer to the DRC report (report_drc) for more information.
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
INFO: [DRC 23-27] Running DRC with 2 threads
INFO: [Project 1-461] DRC finished with 0 Errors
INFO: [Project 1-462] Please refer to the DRC report (report_drc) for more information.
INFO: [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
checkpoint_top_synth
```

Fig 5. Read Checkpoints

- 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.

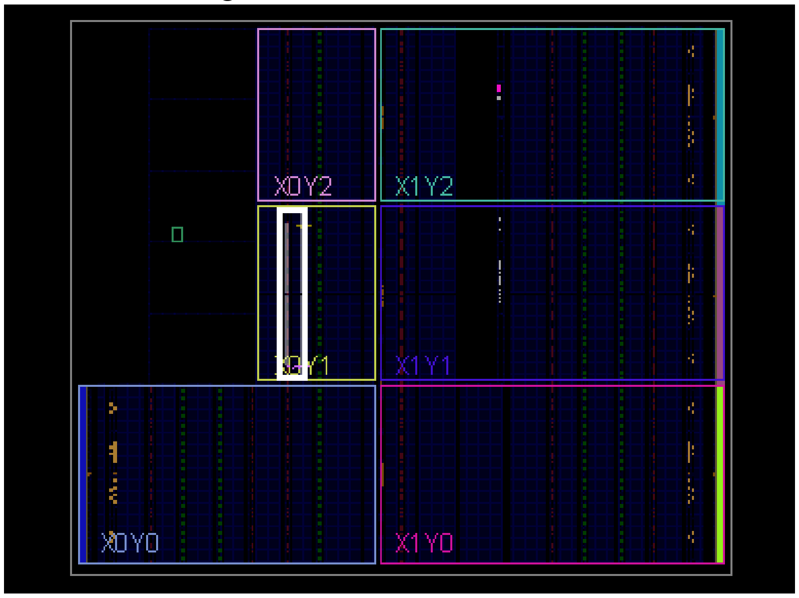
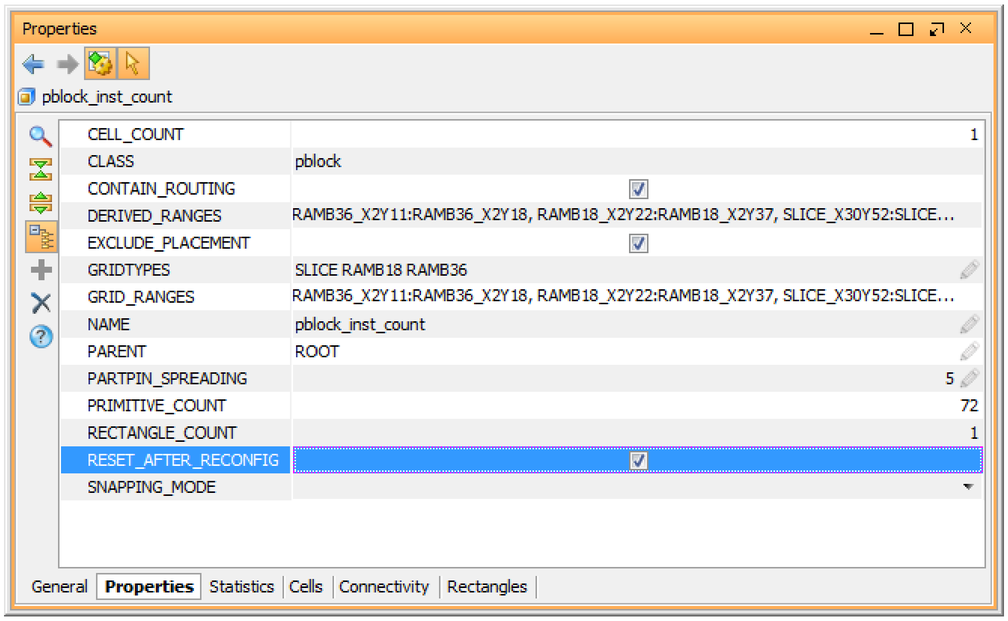


Fig 6. Draw Pblock for inst_count

- 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 .



CELL_COUNT	1
CLASS	pblock
CONTAIN_ROUTING	✓
DERIVED_RANGES	RAMB36_X2Y11:RAMB36_X2Y18, RAMB18_X2Y22:RAMB18_X2Y37, SLICE_X30Y52:SLICE...
EXCLUDE_PLACEMENT	✓
GRIDTYPES	SLICE RAMB18 RAMB36
GRID_RANGES	RAMB36_X2Y11:RAMB36_X2Y18, RAMB18_X2Y22:RAMB18_X2Y37, SLICE_X30Y52:SLICE...
NAME	pblock_inst_count
PARENT	ROOT
PARTPIN_SPREADING	5
PRIMITIVE_COUNT	72
RECTANGLE_COUNT	1
RESET_AFTER_RECONFIG	✓
SNAPPING_MODE	

Fig 7. Set Reset after Reconfiguration

- Repeat the above to steps for inst_shift instance.

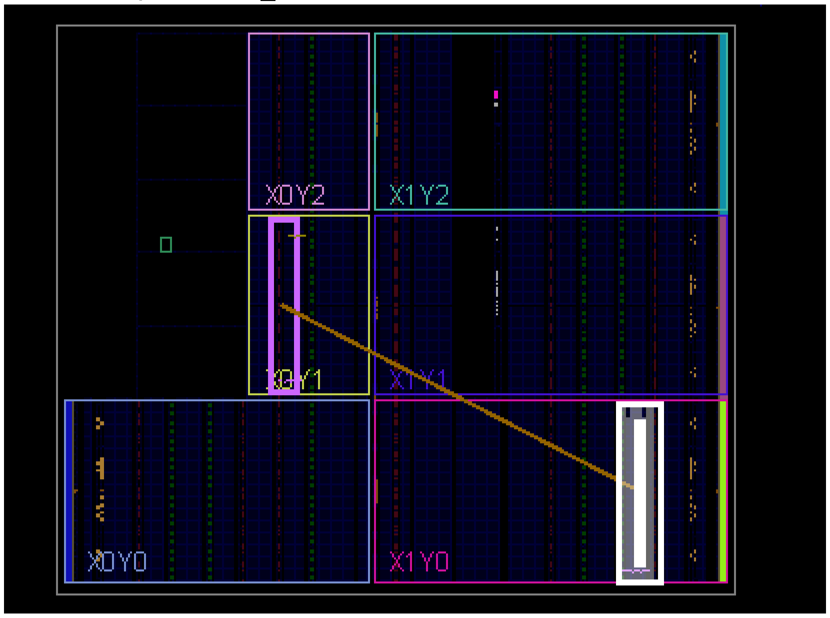
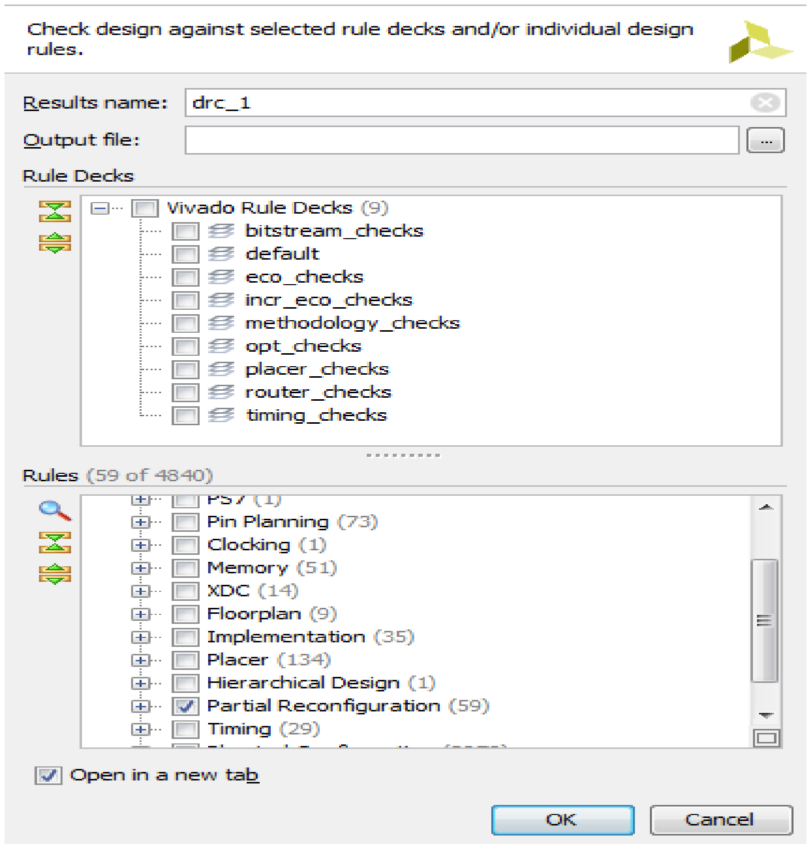


Fig 8. Draw Pblock for inst_shift

- 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.



Check design against selected rule decks and/or individual design rules.

Results name: drc_1

Output file:

Rule Decks

- [ ] Vivado Rule Decks (9)
  - [ ] bitstream_checks
  - [ ] default
  - [ ] eco_checks
  - [ ] incr_eco_checks
  - [ ] methodology_checks
  - [ ] opt_checks
  - [ ] placer_checks
  - [ ] router_checks
  - [ ] timing_checks

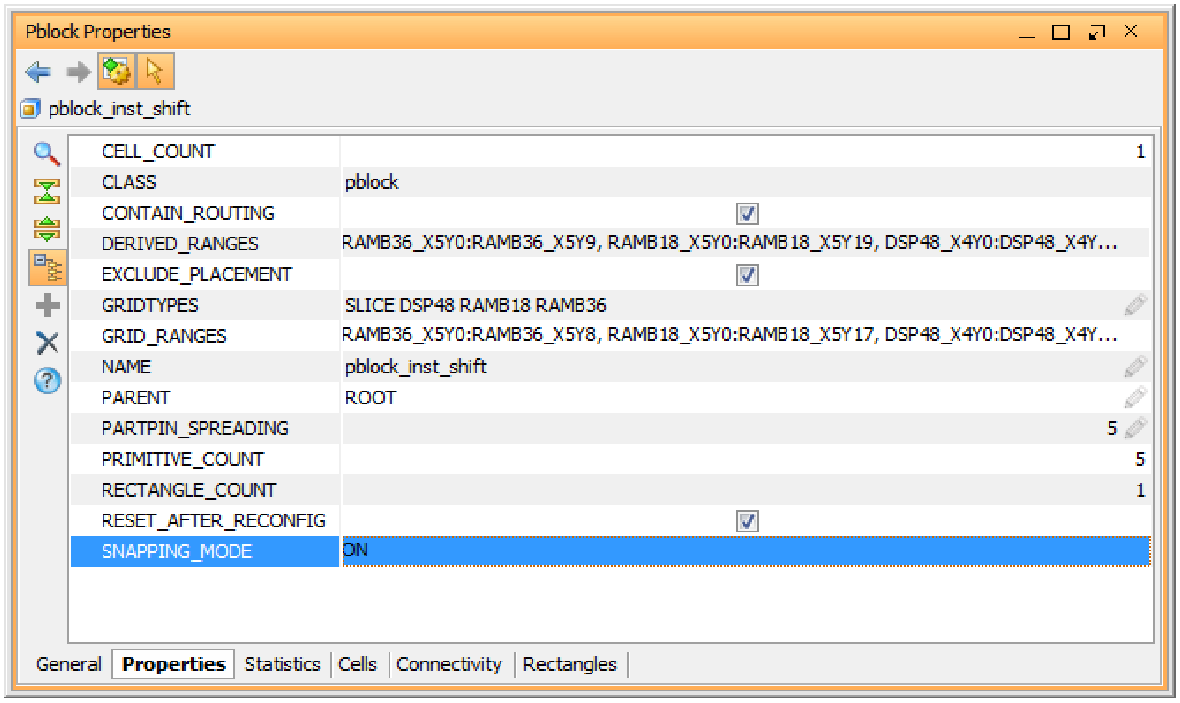
Rules (59 of 4840)

- [ ] PS7 (1)
- [ ] Pin Planning (73)
- [ ] Clocking (1)
- [ ] Memory (51)
- [ ] XDC (14)
- [ ] Floorplan (9)
- [ ] Implementation (35)
- [ ] Placer (134)
- [ ] Hierarchical Design (1)
- [x] Partial Reconfiguration (59)
- [ ] Timing (29)

- [x] Open in a new tab

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.



CELL_COUNT	1
CLASS	pblock
CONTAIN_ROUTING	✓
DERIVED_RANGES	RAMB36_X5Y0:RAMB36_X5Y9, RAMB18_X5Y0:RAMB18_X5Y19, DSP48_X4Y0:DSP48_X4Y...
EXCLUDE_PLACEMENT	✓
GRIDTYPES	SLICE DSP48 RAMB18 RAMB36
GRID_RANGES	RAMB36_X5Y0:RAMB36_X5Y8, RAMB18_X5Y0:RAMB18_X5Y17, DSP48_X4Y0:DSP48_X4Y...
NAME	pblock_inst_shift
PARENT	ROOT
PARTPIN_SPREADING	5
PRIMITIVE_COUNT	5
RECTANGLE_COUNT	1
RESET_AFTER_RECONFIG	✓
SNAPPING_MODE	ON

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:

  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:
  update_design -cell inst_shift -black_box
  update_design -cell inst_count -black_box

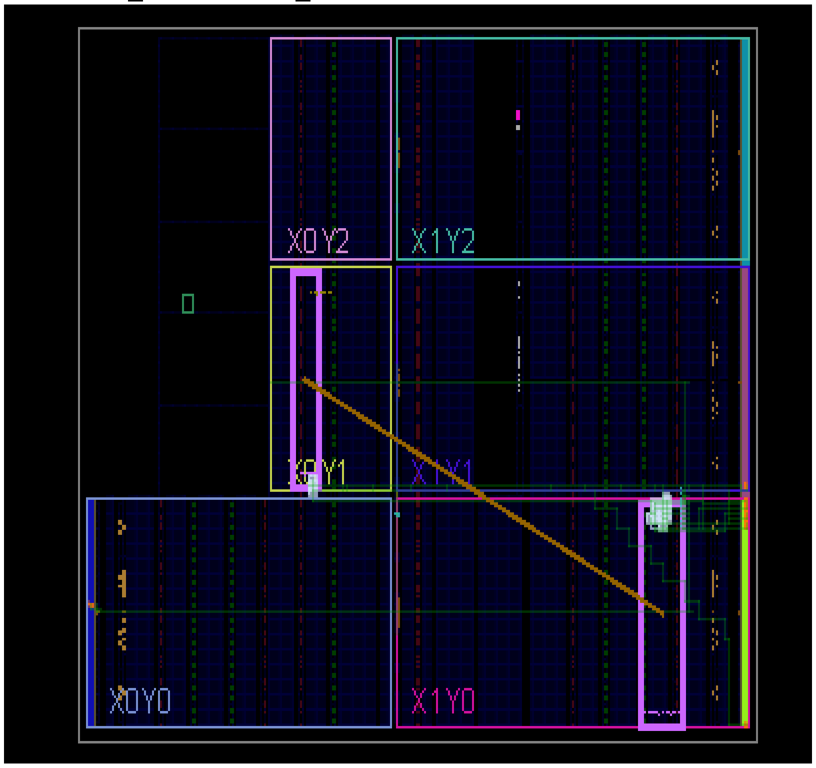


Fig 11. Updated design

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

  lock_design -level routing

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

  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:
    - Config_RightUp.bit - This is the power-up, full design bitstream.
    - Config_RightUp_pblock_inst_shift_partial.bit - This is the partial bit file for the shift_right module.
    - 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.

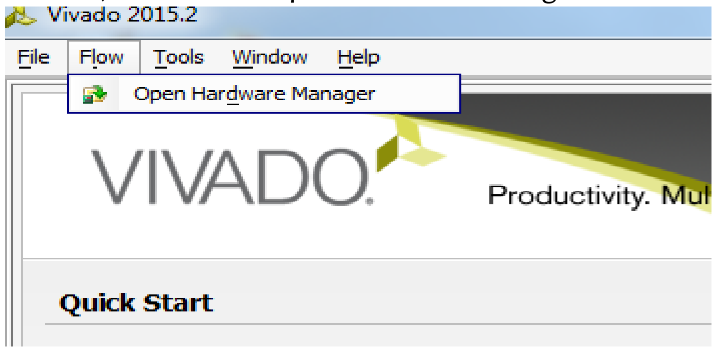


Fig 12. Open Hardware Manager

- Select Open Target >open new target on the green banner. Follow the steps in the wizard to establish communication with the board.

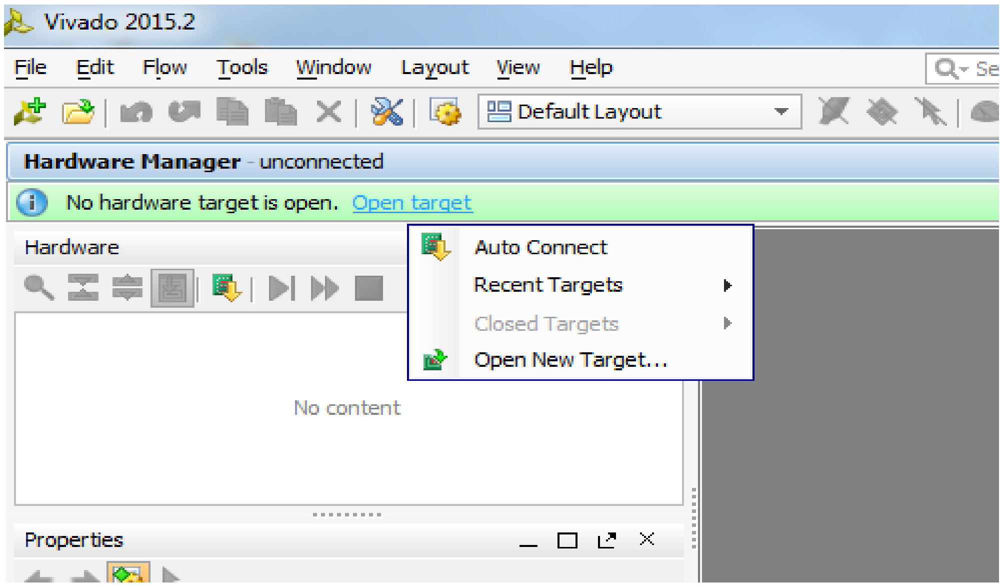

Vivado 2015.2
File Edit Flow Tools Window Layout View Help
Default Layout
Hardware Manager - unconnected
No hardware target is open. Open target
Hardware
Auto Connect
Recent Targets
Closed Targets
Open New Target...
No content
Properties

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.
  - 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.
  - 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