Tutorial on Partial Reconfiguration of Image Processing Blocks using Vivado and SDK

Nishmitha Naveenchandra Kajekar

nishmi@unm.edu

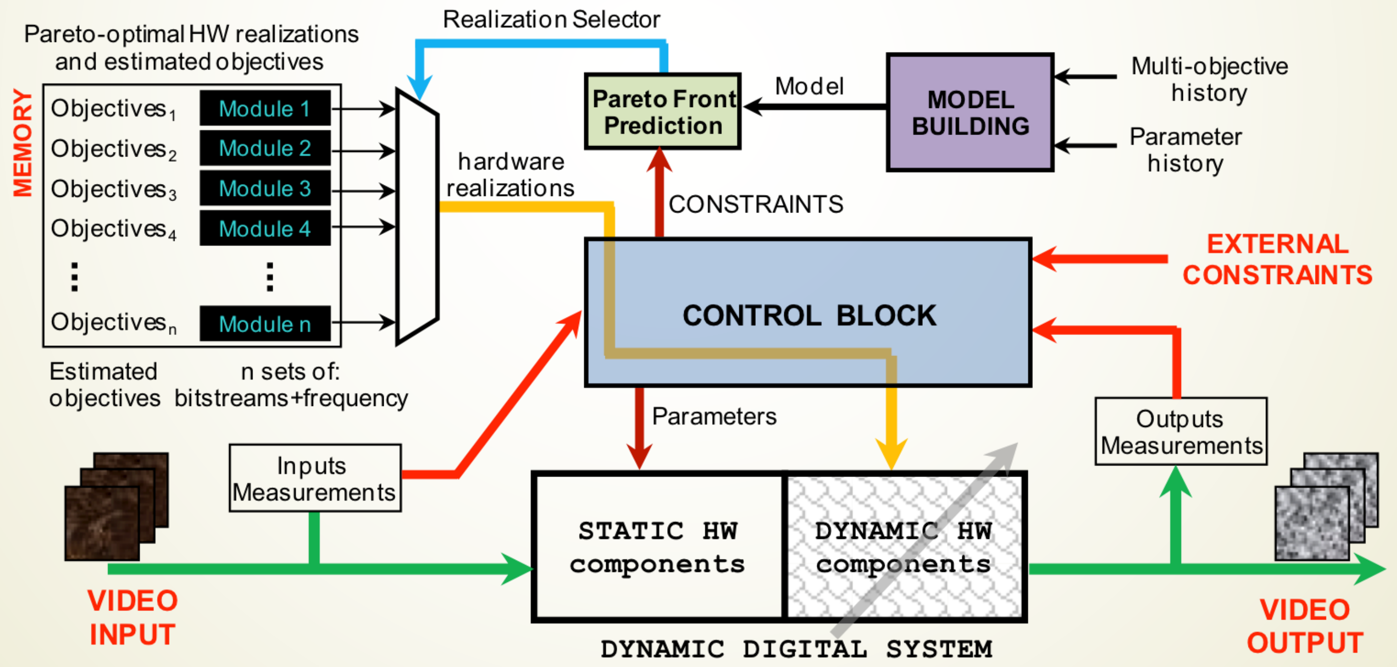
Department of Electrical and Computer Engineering

University of New Mexico

Contents

Motivation

- DRASTIC work in ivPCL which is mainly focused on the development of adaptive video processing systems.
- Need to prepare tutorials for DRASTIC.



Introduction

- Partial Reconfiguration is the modification of an operating FPGA design by loading a partial configuration file which will reduce configuration time and save memory.
- This is a tutorial which describes how to create one reconfigurable partition which implements two reconfigurable module design i.e. Sobel Edge Detector and Gaussian Filter.
- ZedBoard is used to verify the design in hardware using a SD card.

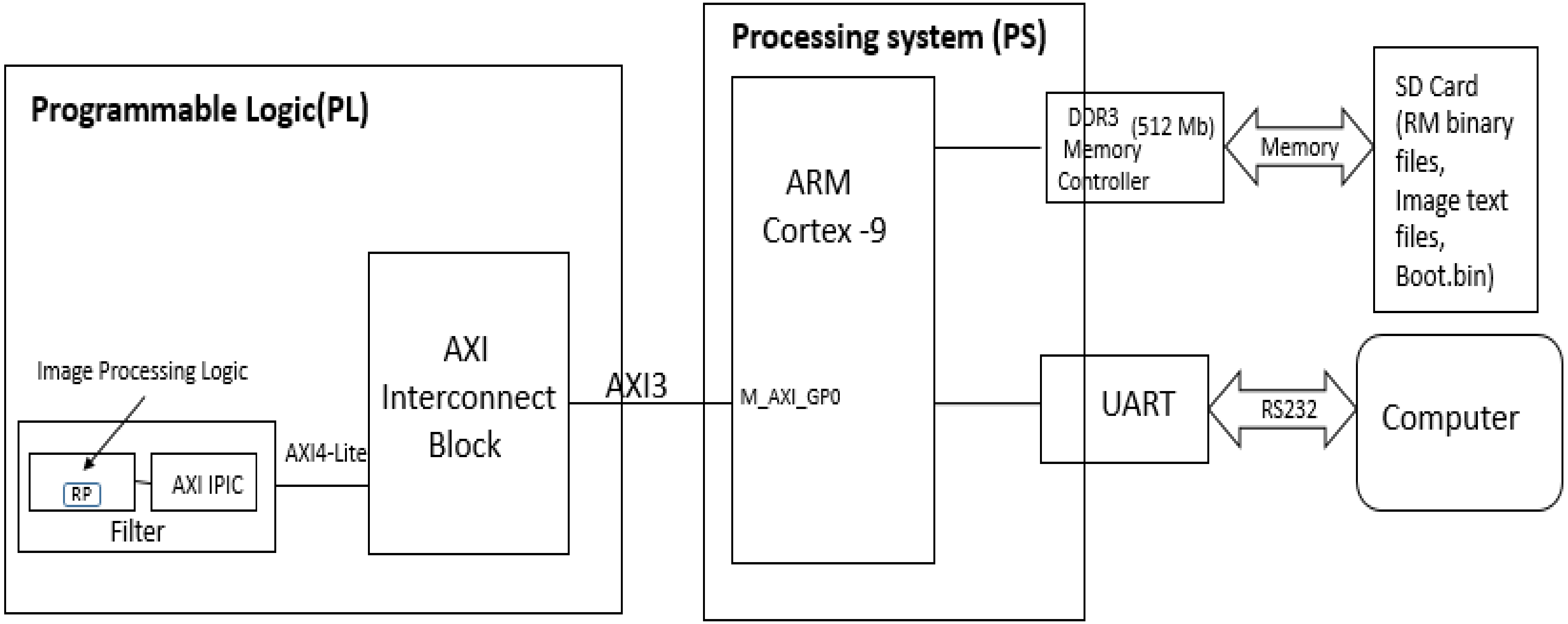
Contribution

- Designed a website where step-by-step Instructions to perform Partial Reconfiguration is given with Video Tutorials.
- This tutorial can be found under http://ivpcl.unm.edu/ivpclpages/Research/drastic/drastic.php
- Modified and Implemented Xilinx Partial Reconfiguration using Vivado project on ZedBoard.
- Designed and Implemented Partial Reconfiguration using an IP which has Reconfigurable modules of Image processing blocks.

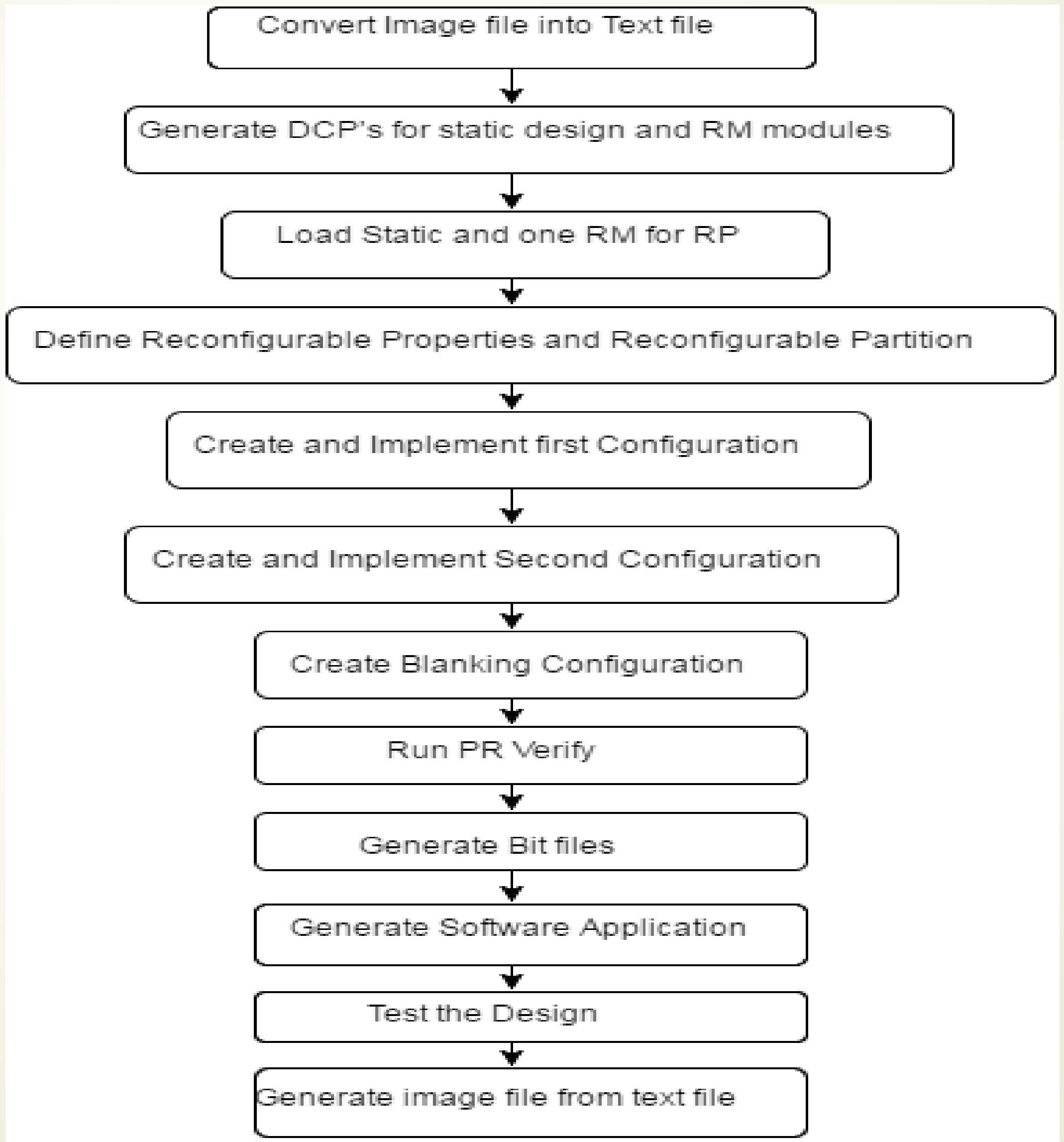
Requirements

- Software Tools:
  - Vivado Design Suite 2015.2
  - Xilinx Software Development Kit 2015.2
  - MATLAB
  - Terminal program (Teraterm)
- Hardware Tools:
  - ZedBoard (Zynq™ Evaluation and Development)
- Licensing:
  - Xilinx Partial Reconfiguration

Design Description



Design Procedure

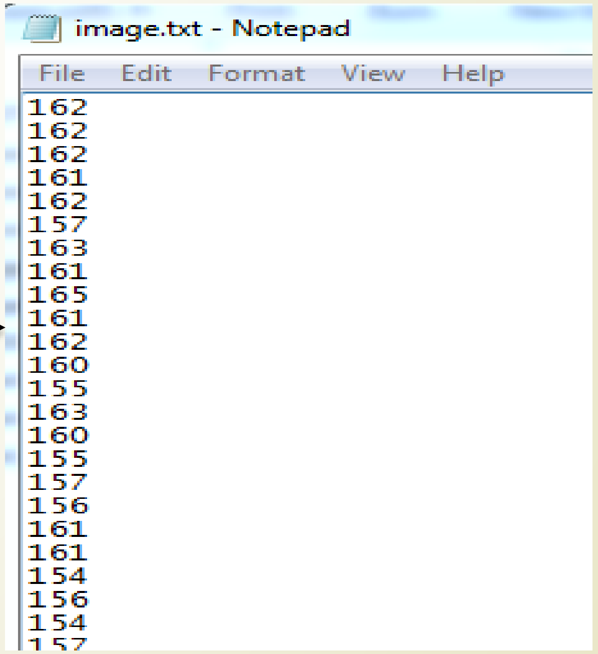

Convert Image file into Text file
Generate DCP's for static design and RM modules
Load Static and one RM for RP
Define Reconfigurable Properties and Reconfigurable Partition
Create and Implement first Configuration
Create and Implement Second Configuration
Create Blanking Configuration
Run PR Verify
Generate Bit files
Generate Software Application
Test the Design
Generate image file from text file

Directory Files

- Extract the PRLab.zip, then the Design Directory Structure is as follows:
  - Bitstreams
  - Checkpoint
  - Implement
  - Sources – IP, source for RM, Software Application, XDC
  - Synth
  - TCL files

Step 1: Convert Image file into Text file

- Read a image file
- Convert it into grayscale image
- Create a image.txt file which is written with the image pixel values.

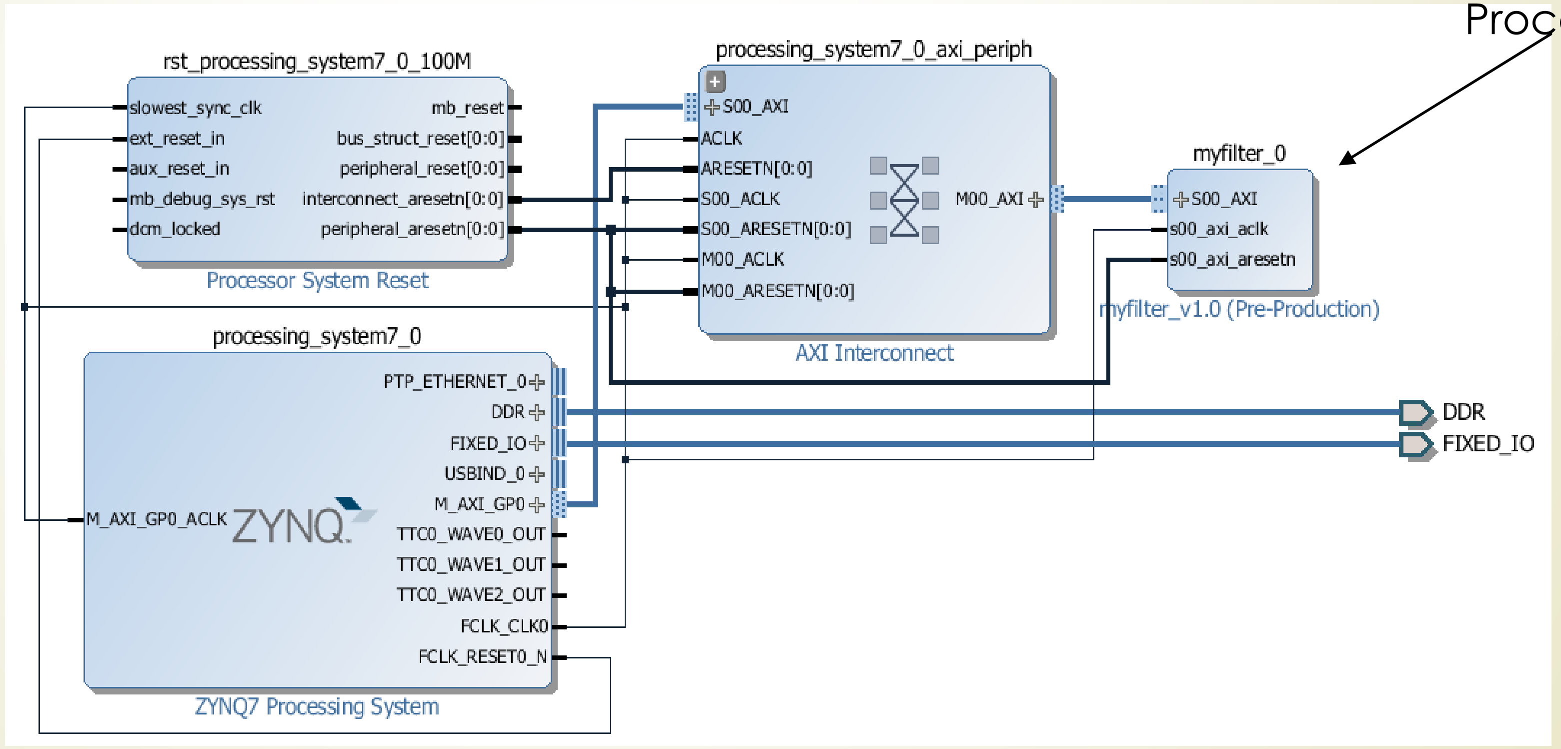


```
162
162
162
161
162
157
163
161
165
161
162
160
155
163
160
155
157
156
161
161
154
156
154
```

Step 2: Generate DCP's for static design and RM modules

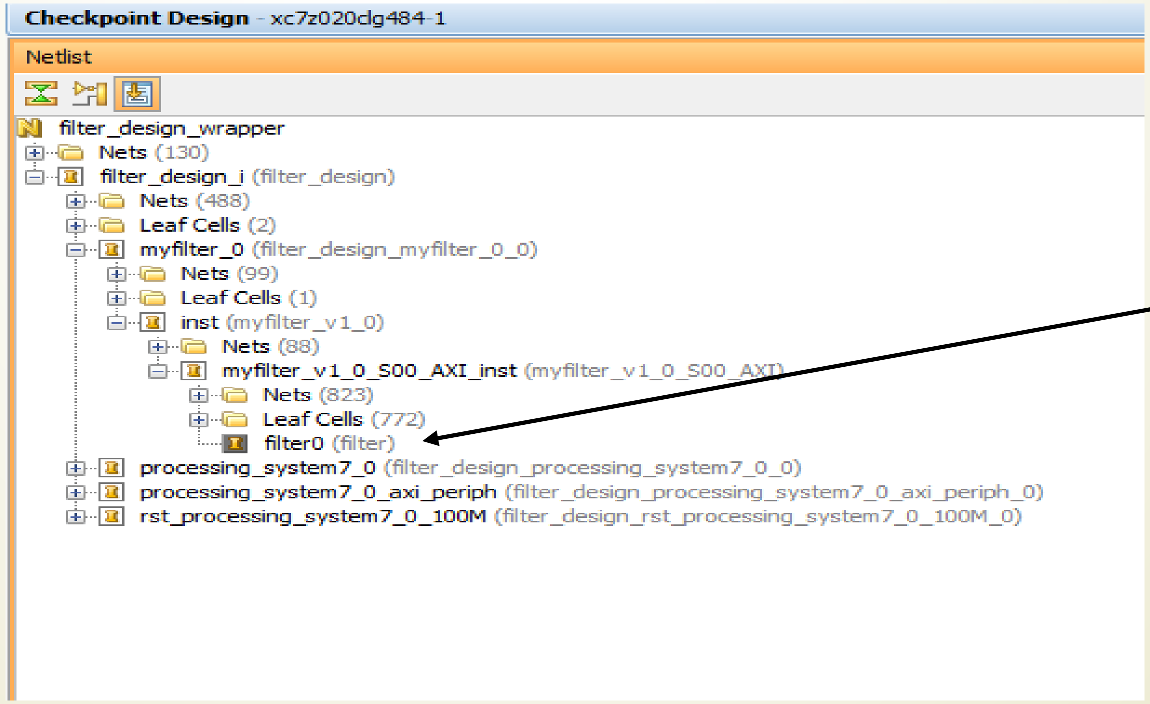
- Creating the block design called filter_design, instantiate ZYNQ PS along with filter IP.

User Logic (Image Processing Logic)



Step 3: Load Static and one RM for RP

- Since all required netlist files (dcp) for the design are now available, we can use Vivado to floorplan the design, define Reconfigurable Partitions and add Reconfigurable Modules.
- We need to the open the static check point created.



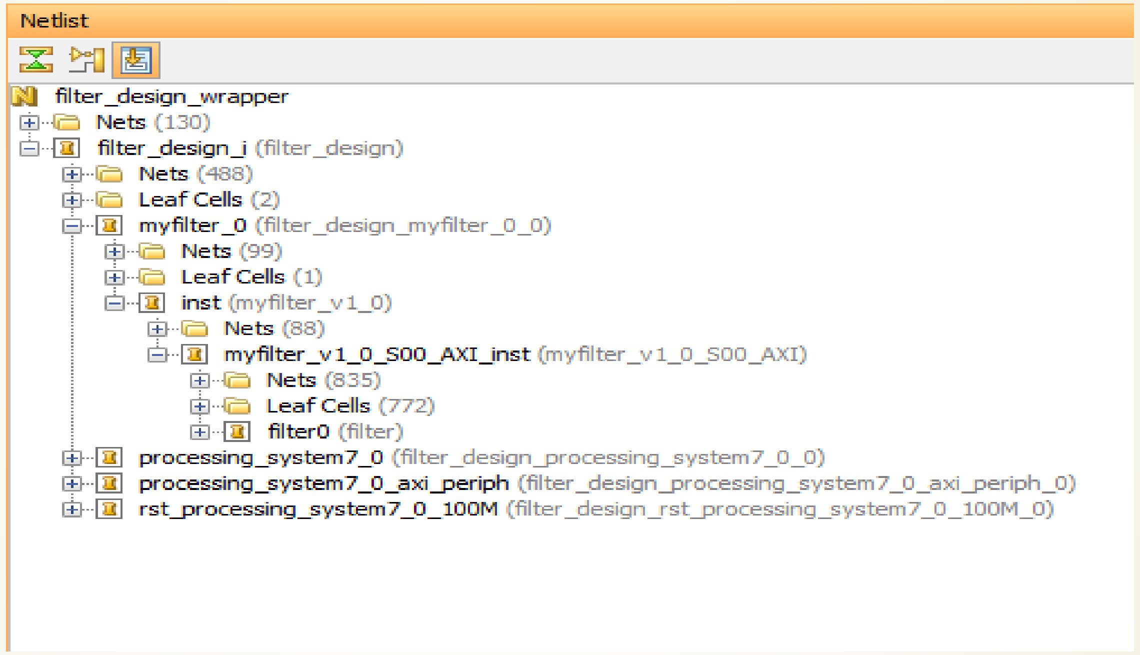
Checkpoint Design - xc7z020clg484-1

Netlist

```
filter_design_wrapper
  Nets (130)
  filter_design_i (filter_design)
    Nets (488)
    Leaf Cells (2)
    myfilter_0 (filter_design_myfilter_0_0)
      Nets (99)
      Leaf Cells (1)
      inst (myfilter_v1_0)
        Nets (88)
        myfilter_v1_0_S00_AXI_inst (myfilter_v1_0_S00_AXI)
          Nets (823)
          Leaf Cells (772)
          filter0 (filter)
    processing_system7_0 (filter_design_processing_system7_0_0)
    processing_system7_0_axi_periph (filter_design_processing_system7_0_axi_periph_0)
    rst_processing_system7_0_100M (filter_design_rst_processing_system7_0_100M_0)
```

Black Box

- Load one RM for the RP by using the read_checkpoint command.
- Now you can see the Black box would be no more as you loaded the reconfigurable module

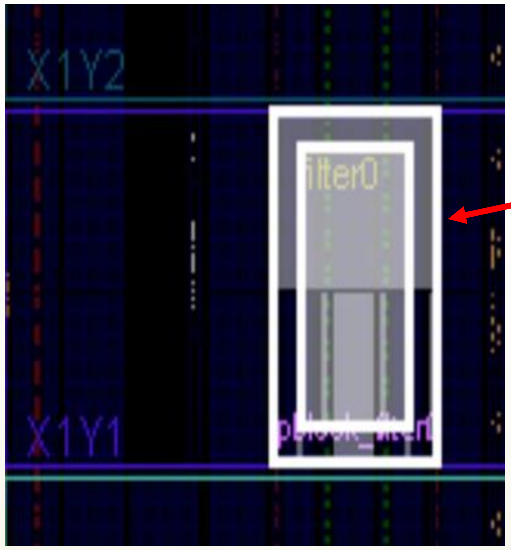


```
Netlist
filter_design_wrapper
  Nets (130)
  filter_design_i (filter_design)
    Nets (488)
    Leaf Cells (2)
    myfilter_0 (filter_design_myfilter_0_0)
      Nets (99)
      Leaf Cells (1)
      inst (myfilter_v1_0)
        Nets (88)
        myfilter_v1_0_S00_AXI_inst (myfilter_v1_0_S00_AXI)
          Nets (835)
          Leaf Cells (772)
          filter0 (filter)
    processing_system7_0 (filter_design_processing_system7_0_0)
    processing_system7_0_axi_periph (filter_design_processing_system7_0_axi_periph_0)
    rst_processing_system7_0_100M (filter_design_rst_processing_system7_0_100M_0)
```

- Make a note of the Resources utilized under the statistics tab.

Step 4: Define Reconfigurable Properties and Reconfigurable Partitions

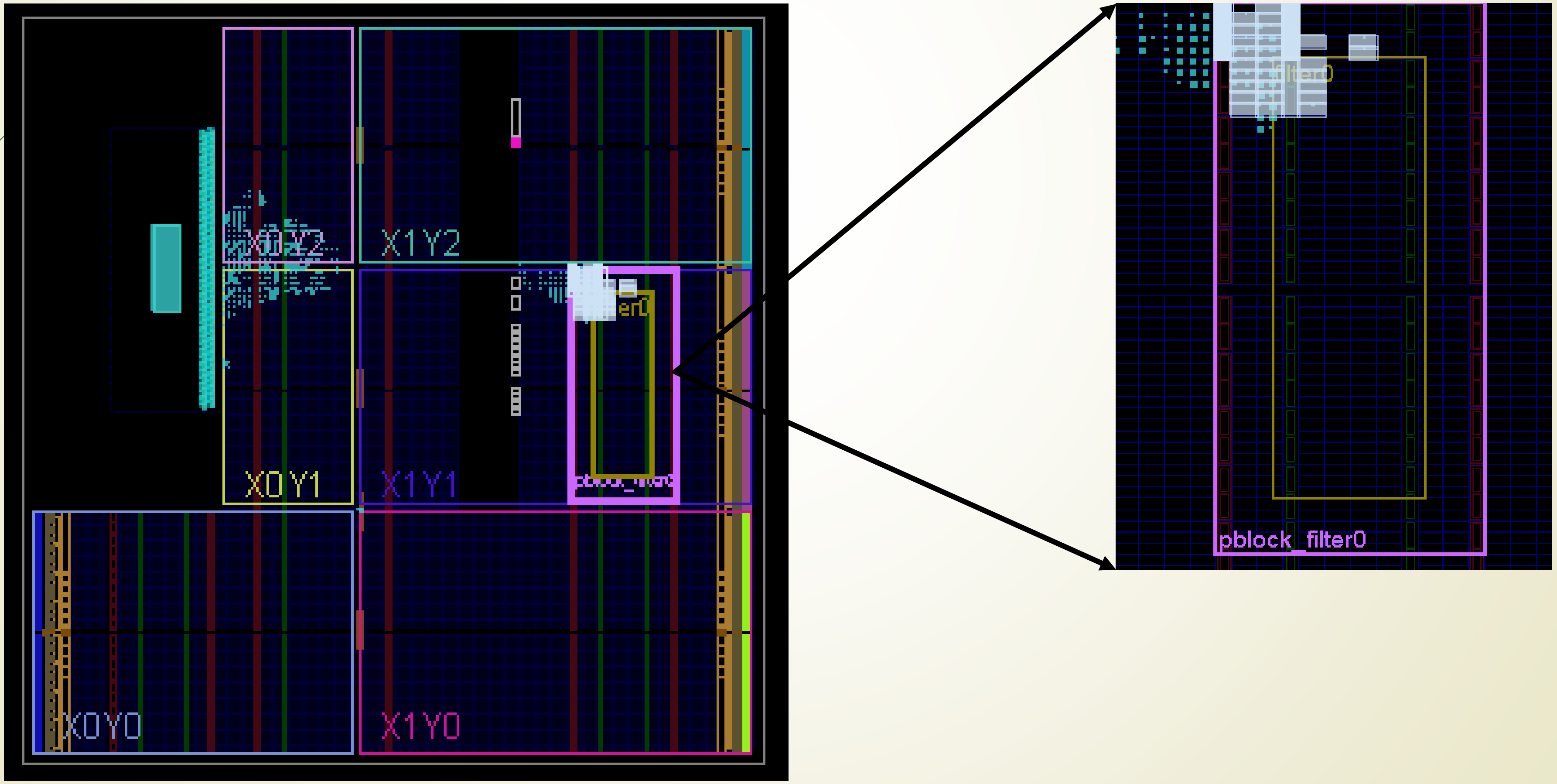
- Define the reconfigurable properties to the loaded RM by setting the HD.RECONFIGURABLE property.
- Next you must floorplan the RP region.



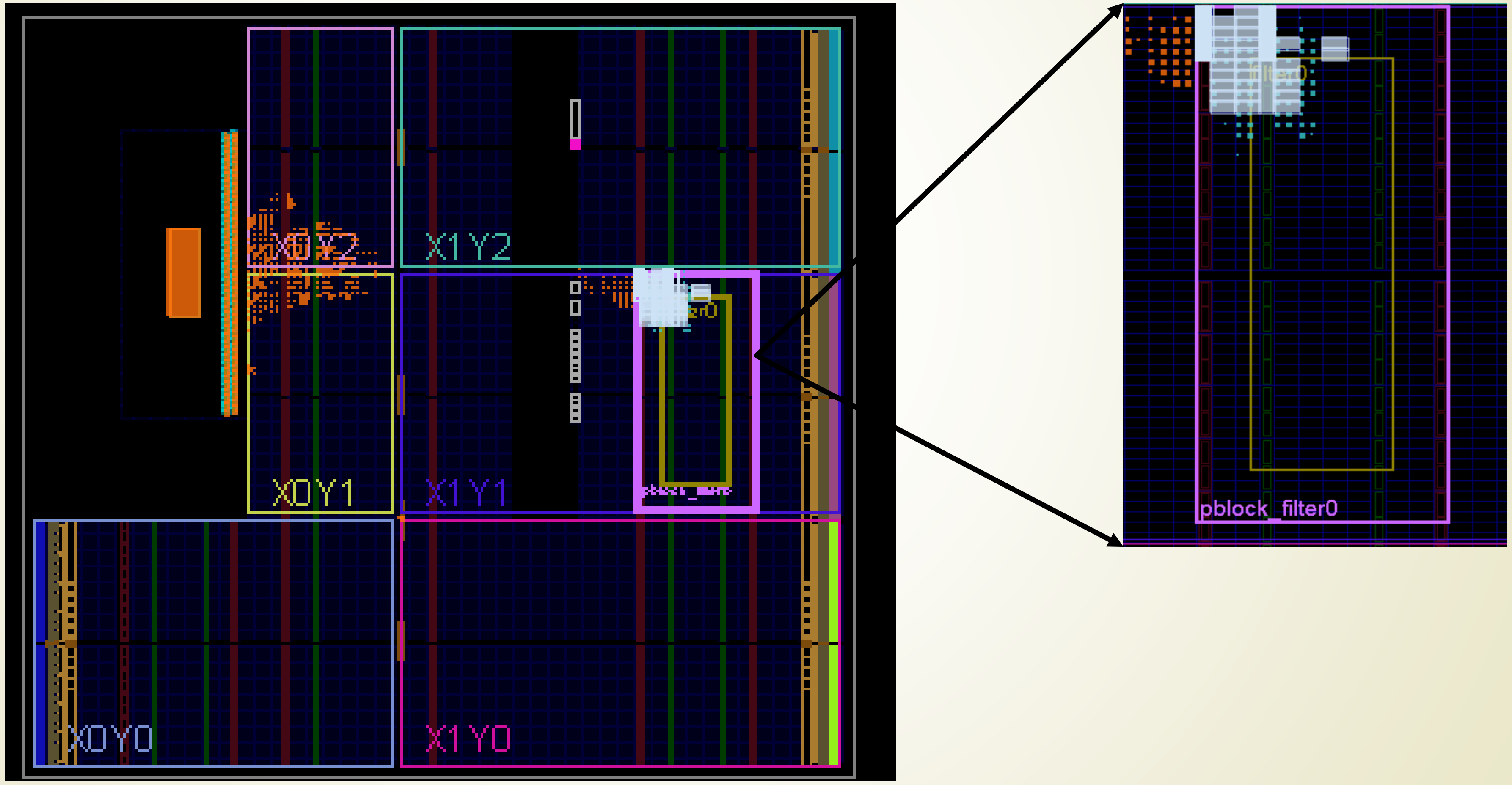
Designed pblock

Step 5: Create and Implement first Configuration

- Optimize, place and route the design.

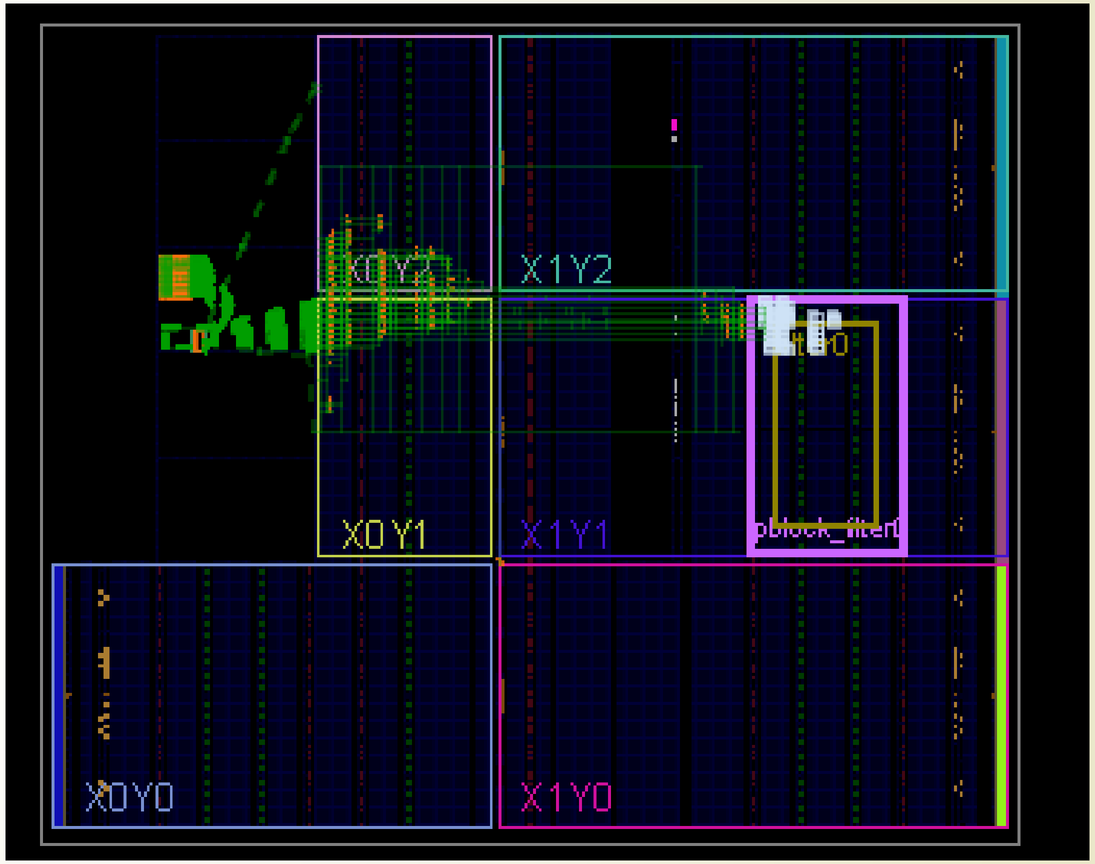


Step 6: Create and Implement second Configuration



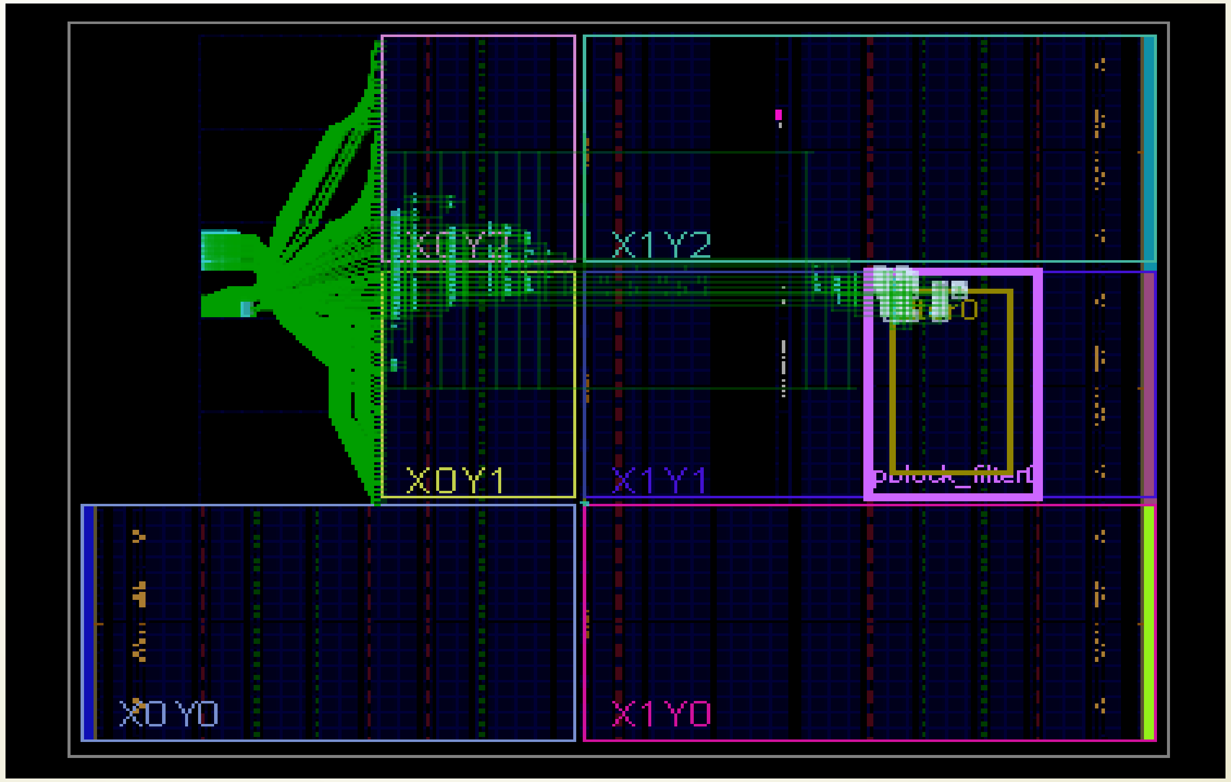
Step 7: Create Blanking Configuration

- This has no logic for either reconfigurable partition, simply outputs driven by ground. Outputs can be tied to VCC if desired, using the HD.PARTPIN_TIEOFF property.



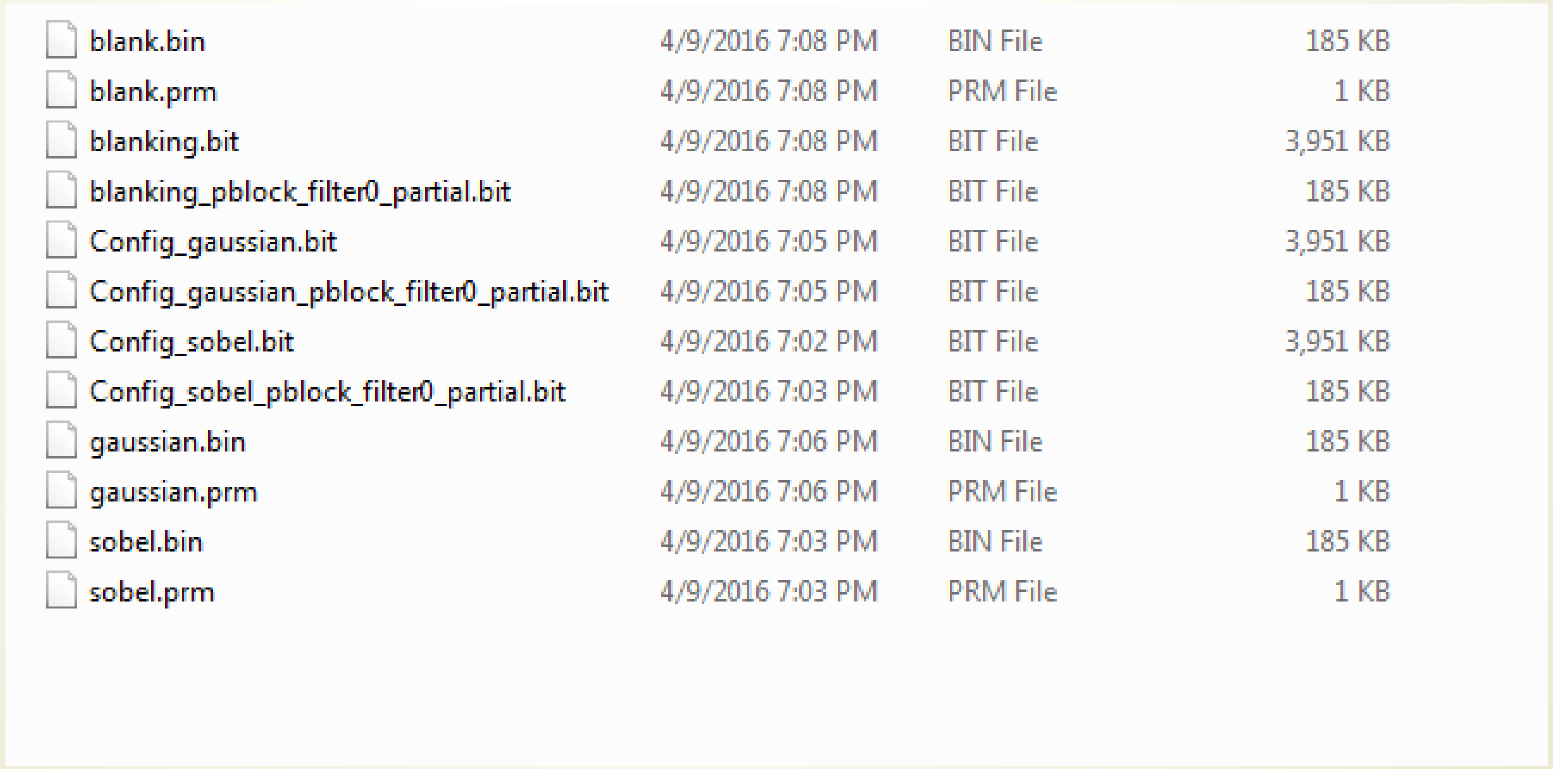
Step 8: Run PR Verify

- You must ensure that the static implementation, including interfaces to reconfigurable regions, is consistent across all Configurations. To verify this, you run the PR_Verify utility.



Step 9: Generate Bit files

- After all the Configurations have been validated by PR_Verify, full and partial bit files must be generated for the entire project.



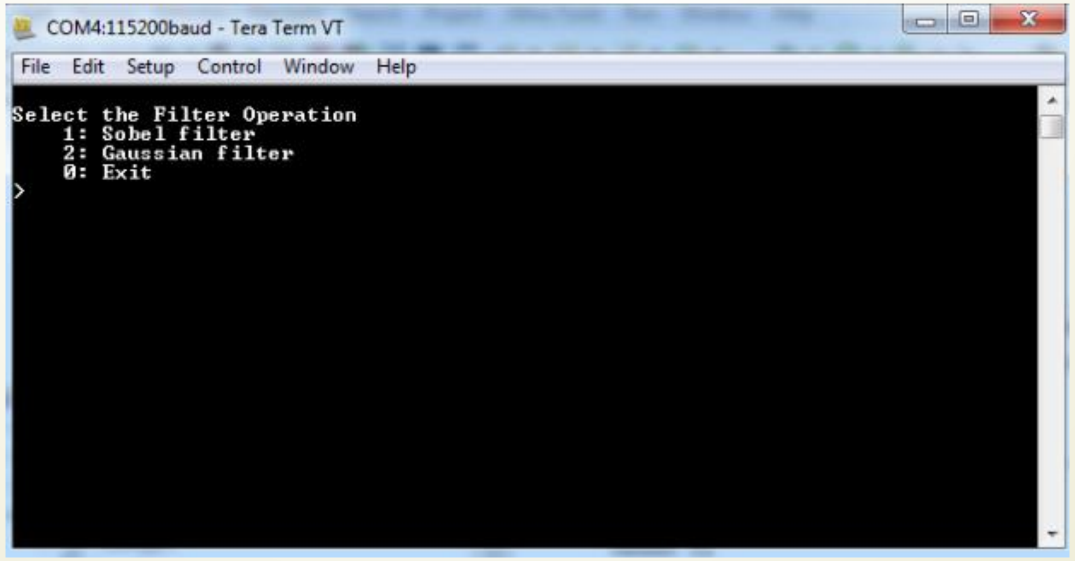
blank.bin	4/9/2016 7:08 PM	BIN File	185 KB
blank.prm	4/9/2016 7:08 PM	PRM File	1 KB
blanking.bit	4/9/2016 7:08 PM	BIT File	3,951 KB
blanking_pblock_filter0_partial.bit	4/9/2016 7:08 PM	BIT File	185 KB
Config_gaussian.bit	4/9/2016 7:05 PM	BIT File	3,951 KB
Config_gaussian_pblock_filter0_partial.bit	4/9/2016 7:05 PM	BIT File	185 KB
Config_sobel.bit	4/9/2016 7:02 PM	BIT File	3,951 KB
Config_sobel_pblock_filter0_partial.bit	4/9/2016 7:03 PM	BIT File	185 KB
gaussian.bin	4/9/2016 7:06 PM	BIN File	185 KB
gaussian.prm	4/9/2016 7:06 PM	PRM File	1 KB
sobel.bin	4/9/2016 7:03 PM	BIN File	185 KB
sobel.prm	4/9/2016 7:03 PM	PRM File	1 KB

Step 10: Generate Software Application

- Create a Board Support Package
- Create a FiterTest application project
- Create a zynq_fsbl application – Bootloader, Used to configure FPGA with bit stream
- Create a Zynq boot image

Step 11: Test the Design

- Copy the generated BOOT.bin and the bin files on the SD card along with it make sure the SD card has the image text files then place the SD card in the board. Power On the board.
- Start a terminal emulator program i.e. TeraTerm. Select an appropriate COM port. Set the COM port for 115200 baud rate communication.


COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help

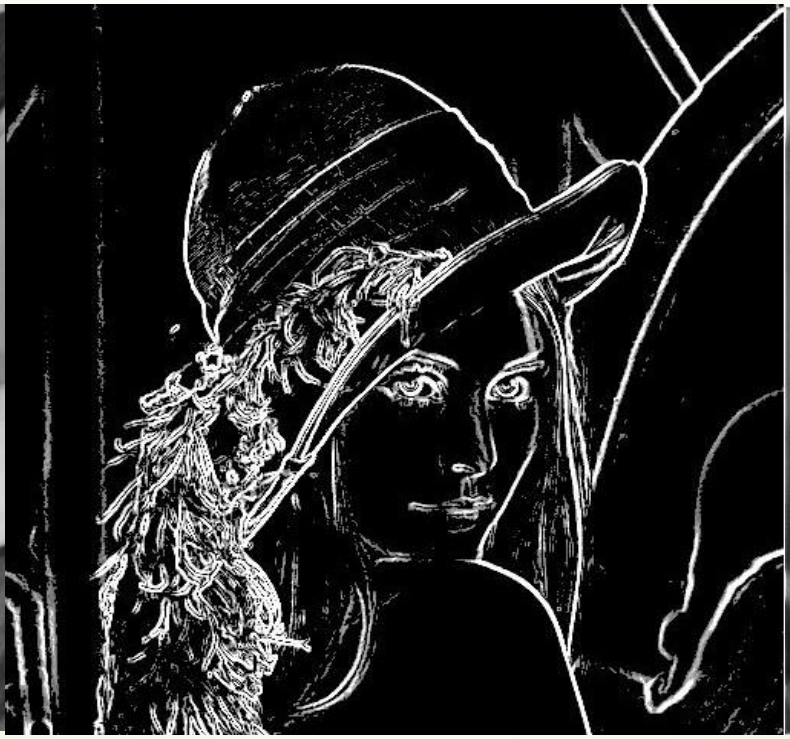
```
Select the Filter Operation
    1: Sobel filter
    2: Gaussian filter
    0: Exit
>
```

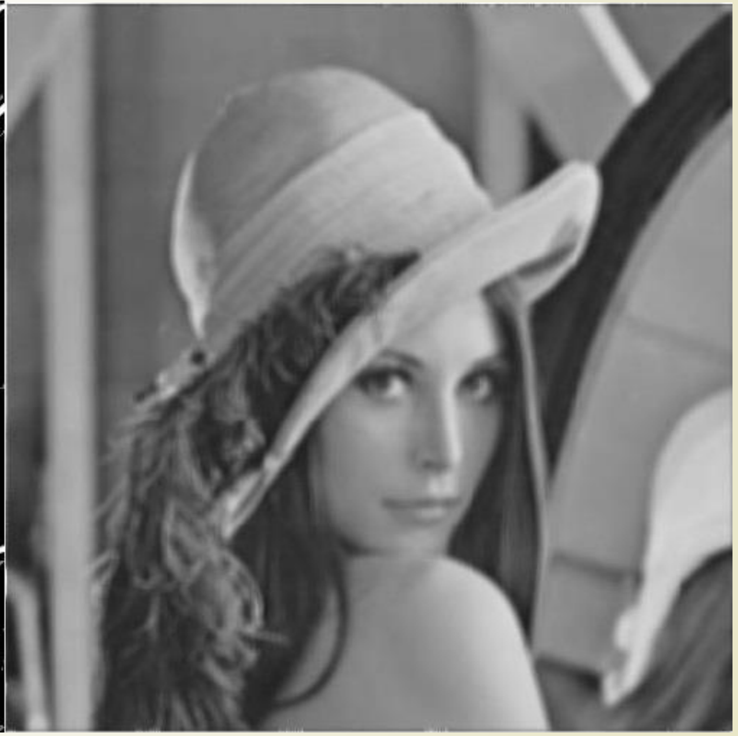
Step 12: Generate image file from text file

- Create a matlab file to convert text file to image file.
- Use the given sobel.m and Gaussian.m files in the PRLab.zip folder

Results from ZedBoard

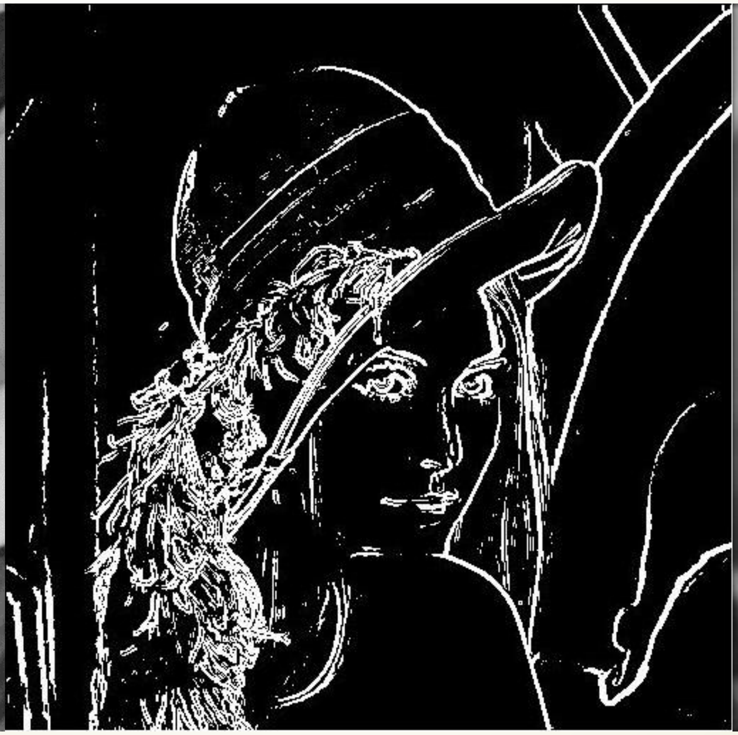
Original Image

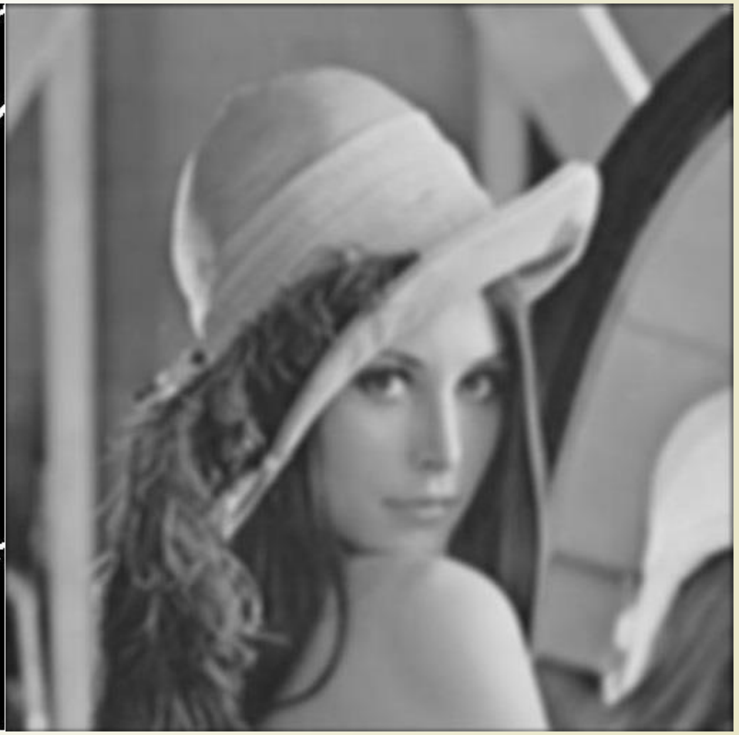

After Sobel Filter


After Gaussian Filter

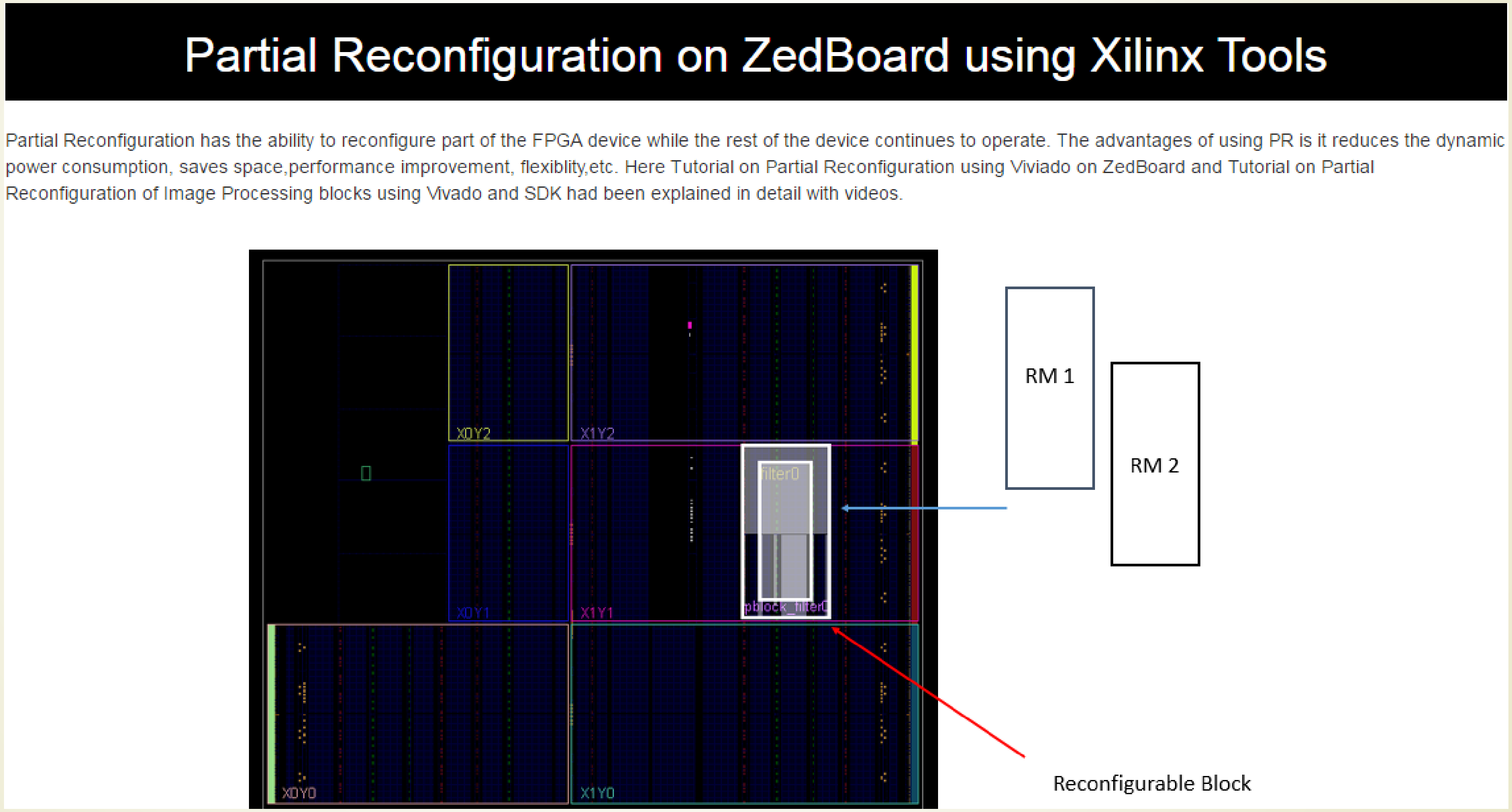
MATLAB Visual Verification

Original Image


After Sobel Filter


After Gaussian Filter

Web Tutorials



Partial Reconfiguration on ZedBoard using Xilinx Tools

Partial Reconfiguration has the ability to reconfigure part of the FPGA device while the rest of the device continues to operate. The advantages of using PR is it reduces the dynamic power consumption, saves space,performance improvement, flexiblity,etc. Here Tutorial on Partial Reconfiguration using Viviado on ZedBoard and Tutorial on Partial Reconfiguration of Image Processing blocks using Vivado and SDK had been explained in detail with videos.

Conclusion

- This tutorial helps in understanding steps involved in creating a processor system using Vivado IPI.
- Generating Full bitstreams as well as partial reconfiguration bitstreams along with bin files.
- Generating the boot image as well as verifying the functionality using ZedBoard.

Future Work

- To perform Partial configuration on large images and on real-time videos.
- The Ultimate goal is to be able to perform PR for heterogeneous video computing.

References

- Vivado Design Suite Tutorial: Partial Reconfiguration (UG947)
- Partial Reconfiguration User Guide (UG702) - For ISE Design Tool
- Vivado Design Suite Tcl Command Reference Guide (UG835)
- Vivado Design Suite User Guide: Designing with IP (UG896)
- Partial Reconfiguration User Guide (UG909)
- Partial Reconfiguration of a Hardware Accelerator with Vivado Design Suite (XAPP1231)
- Xilinx University Program on Partial Reconfiguration Flow on Zynq using Vivado
- Tutorials developed and taught by Prof. Daniel Llamocca http://www.secs.oakland.edu/~llamocca/EmbSysZynq.html