Lab 6: 2D Image Convolution

In this Lab, we are going to using Vivado High Level Synthesis (HLS) tool to create a hardware accelerator for 2D image convolution. You will need both this document and the lecture recording to be able get this lab done.

1. Vivado HLS

This section will walk you through creating a new Vivado HLS project

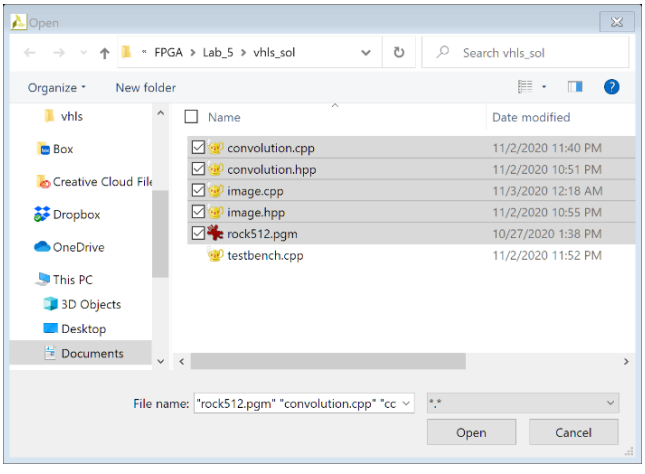
a. Download the lab6_template.zip and extract it.

b. Open Vivado HLS and click on New Project.

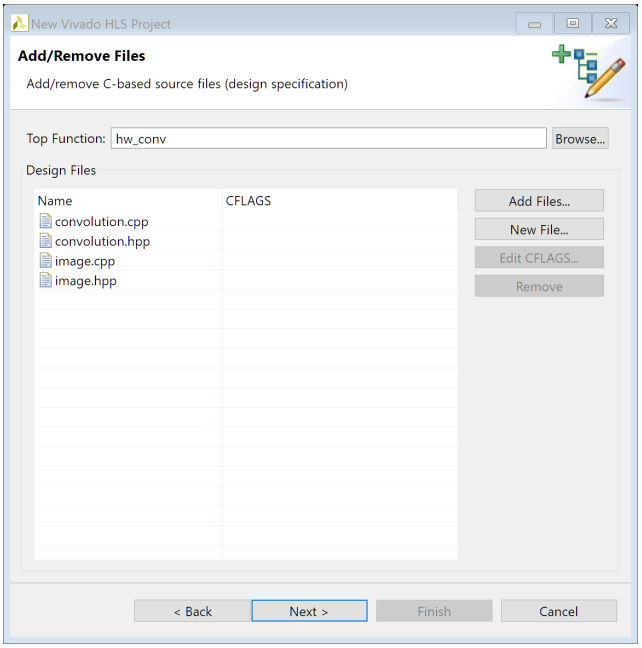
c. Choose a project name and make sure that the path to it doesn't have any spaces at all! Click Next.

d. Copy the folder name "vhls" under lab6_template folder and paste in the folder of the project.

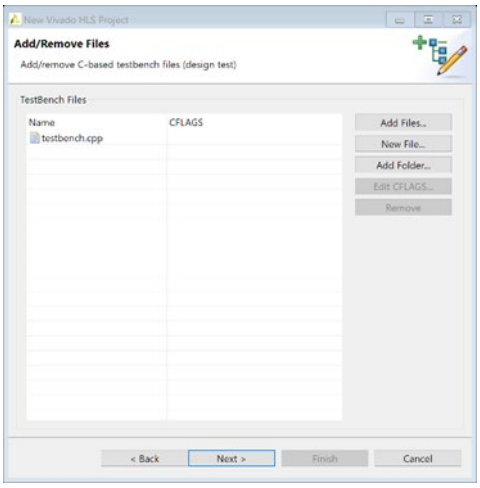
e. Choose the following files as your sources:



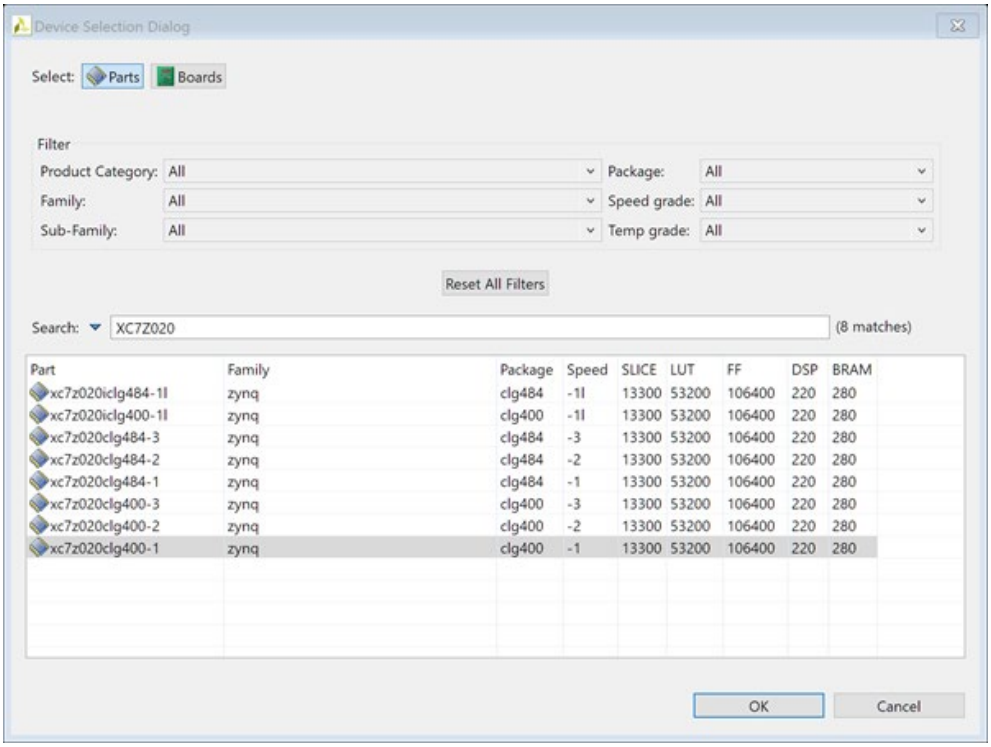
f. Next to Top Function, click on browse and choose hw_conv (convolution.cpp) as the top function. Click Next.
Note: this image is just missing the rock512.pgm file, but it should be there.



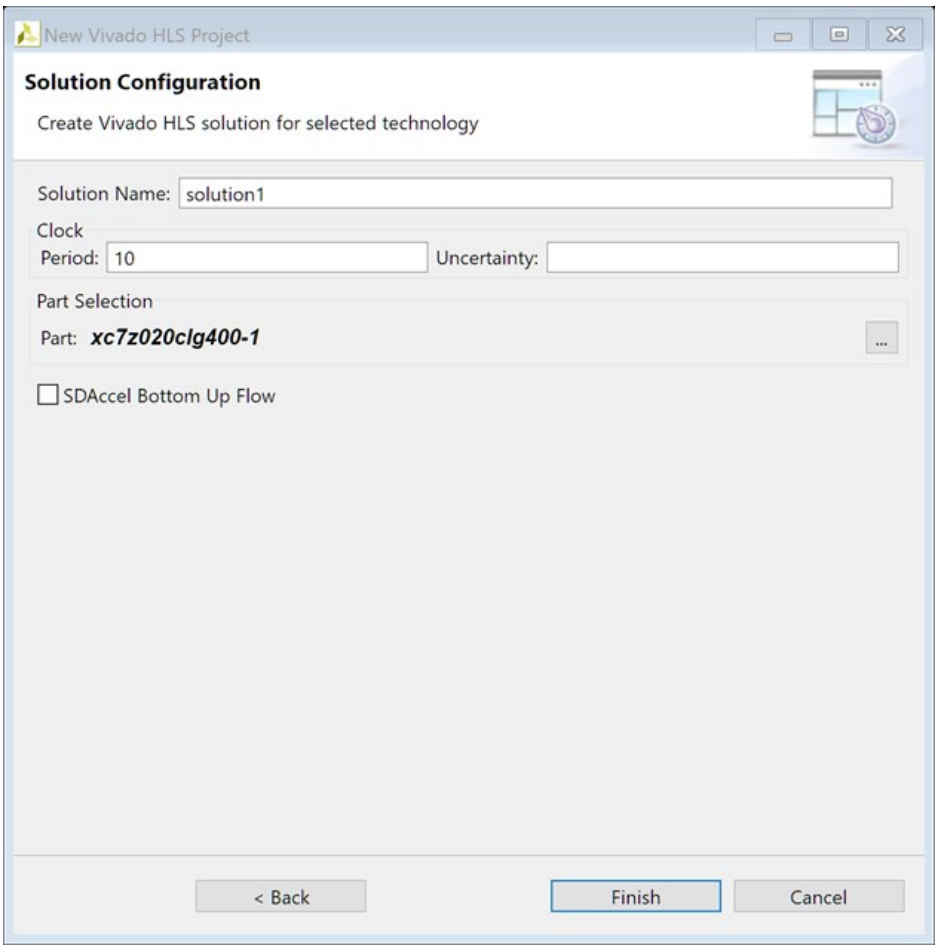
g. Choose the testbench.cpp as your simulation file. Click Next.



h. In this window, we are going to choose ZYNQ Part number: search for: xc7z020clg and choose the following part number: Click Ok.



i. Click Finish here



2. Synthesizing Top Function

Now you should open the convolution.cpp and start implementing the function named hw_conv. Note that you would need the hw_conv to be completed before running the synthesize step. In this step, Vivado HLS will try and generate hardware (.vhd) file representing this Top Function. It will take into consideration all the directive written and will try to optimize your design as it can. Note that different C code for this part will yield to different implementation and VERY different resources utilization, so make sure to optimize your C code as much as you can.

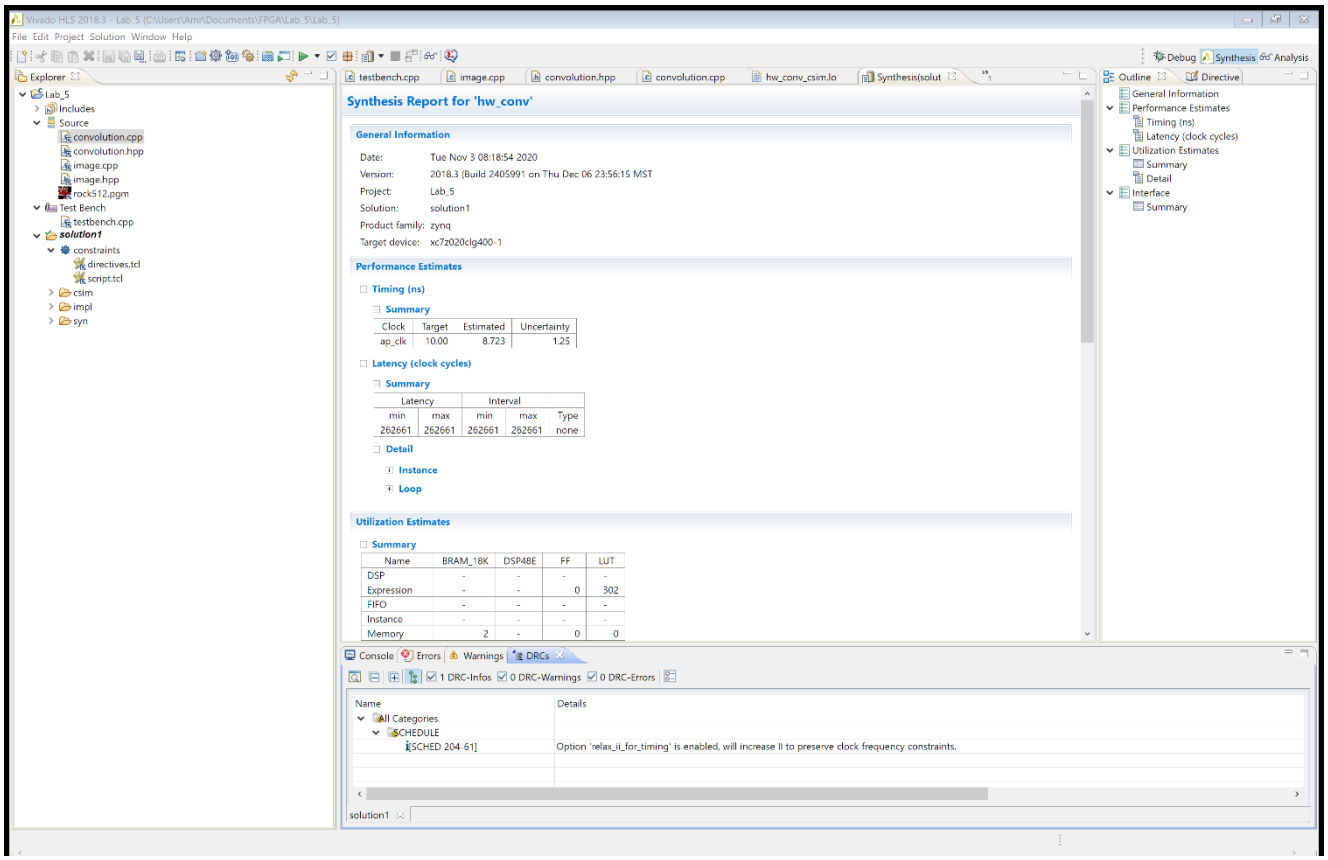
You will be implementing two different versions of this function:

1. Line buffers implemented as shift register
2. Line buffers implemented as ring buffers.

Make sure to take screenshots in both cases of the utilization and timing reports.

a. When you are done implementing your C function, click on the Synthesize button.

b. If there were no errors in the synthesis and implementation, you should see the following window. Any errors will be displayed in the console window, you can go back and fix them.

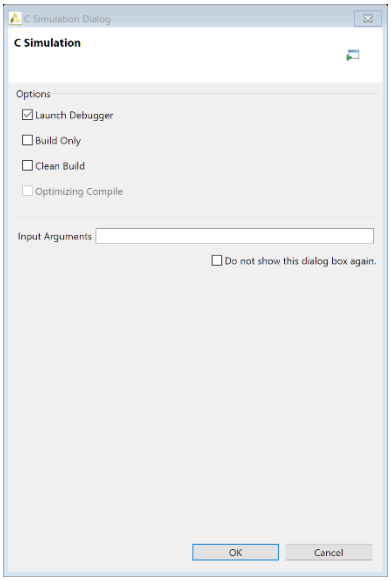


c. This is the generated Synthesis Report, which has an estimate of the Performance (Timing, maximum Clock frequency, Utilization, no. of each of the resources used, and the Interface ports).

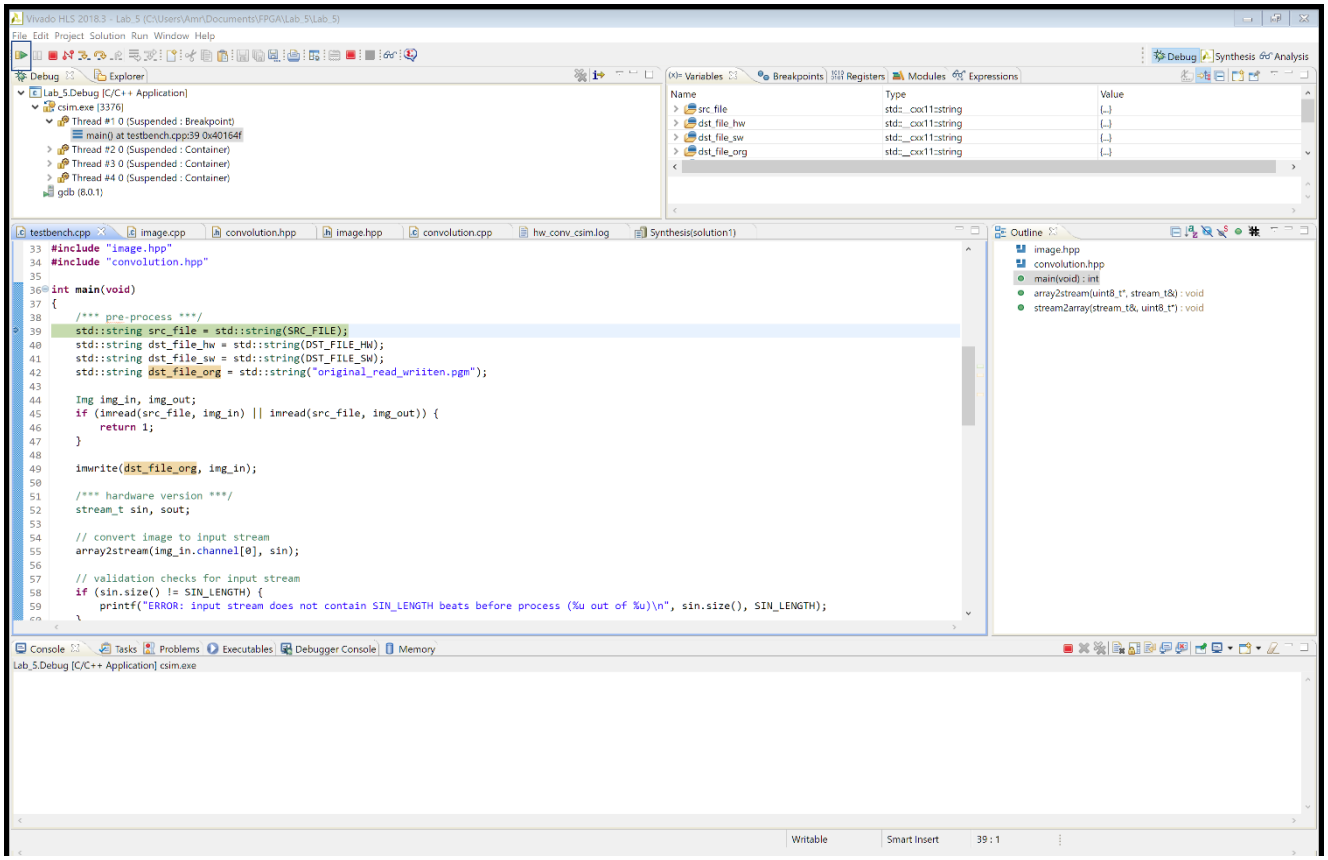
Familiarize yourself with this report, expand different parts and read it. You will be needing to

report the Timing and Utilization Estimates for your design. Again, different designs will have very different utilization summary.

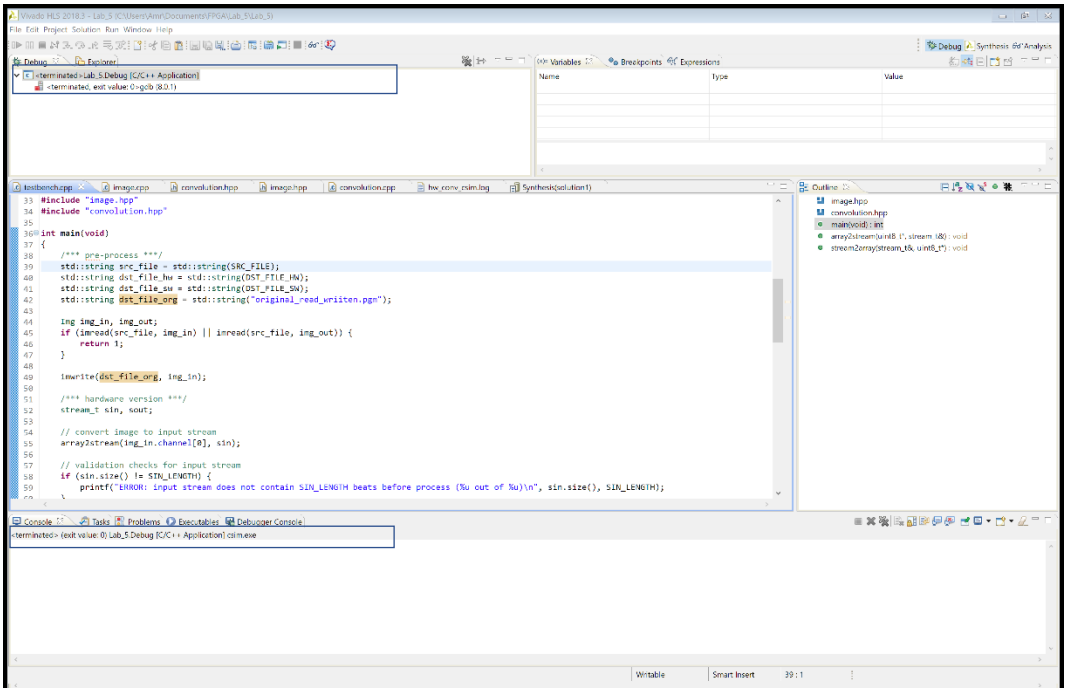
d. Now you are ready to Simulate. Click on the Run Simulation Icon, and Check Launch Debug in the following window.
Note: you might also want to check Clean Build in between runs, just to make sure that nothing gets written over.



e. If the simulation was able to start successfully, then you the view will be switched to the Debug view as in the following window. This is the Debug mode, so you are able to go through your code step by step. If you wish to run your code entirely, Click on Resume (the first icon to the left) or hit F8. In the following Simulation runs, if you don't wish to go in the Debug mode and run the code directly, uncheck the "Launch Debugger" in the previous step.



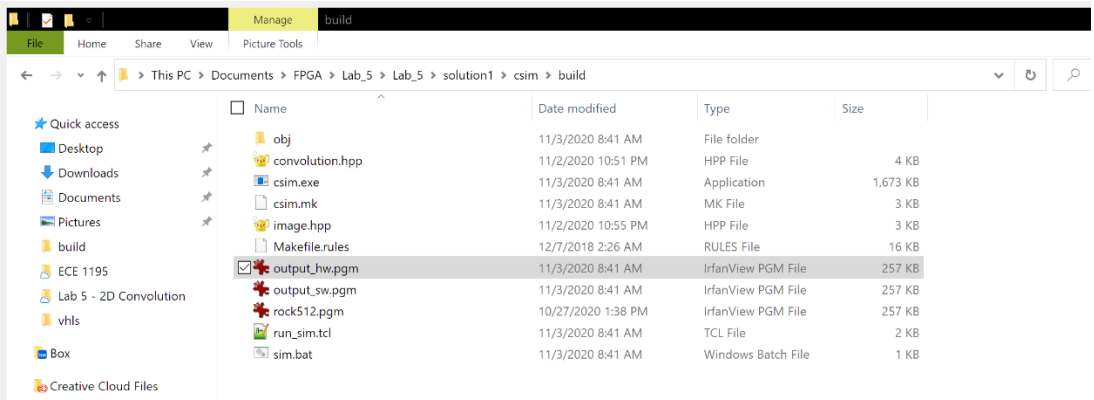
f. Any errors, will appear in the console window, you can go back and fix them. If the code was run in its entirety with no errors, you will see the following window. Note that the exit code is zero, meaning no issues.



g. This code will generate 2 images, one from the sw_conv (output_sw.pgm) and one form the hw_conv (output_hw.pgm). You can view these files by navigating to: “project dir”/“solution name”/csim/build.
Note that solution name in my case is solution1.
rock512.pgm is the original picture.

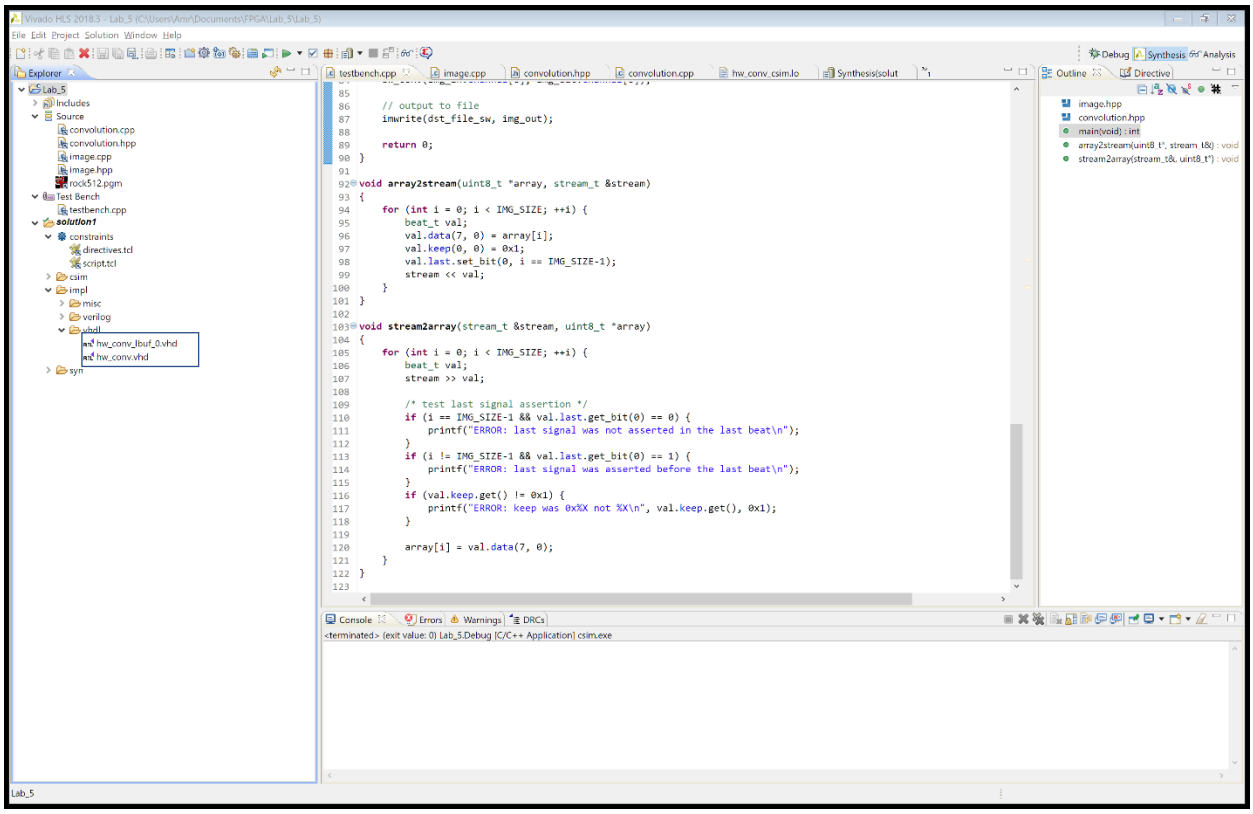
In order to view those pictures, you would need to download the following image viewer from this link: https://www.irfanview.com/

You will find that output_sw.pgm is the edges detected in the original picture. If your code for hw_conv is correct, both output_hw.pgm and output_sw.pgm should be exactly the same.

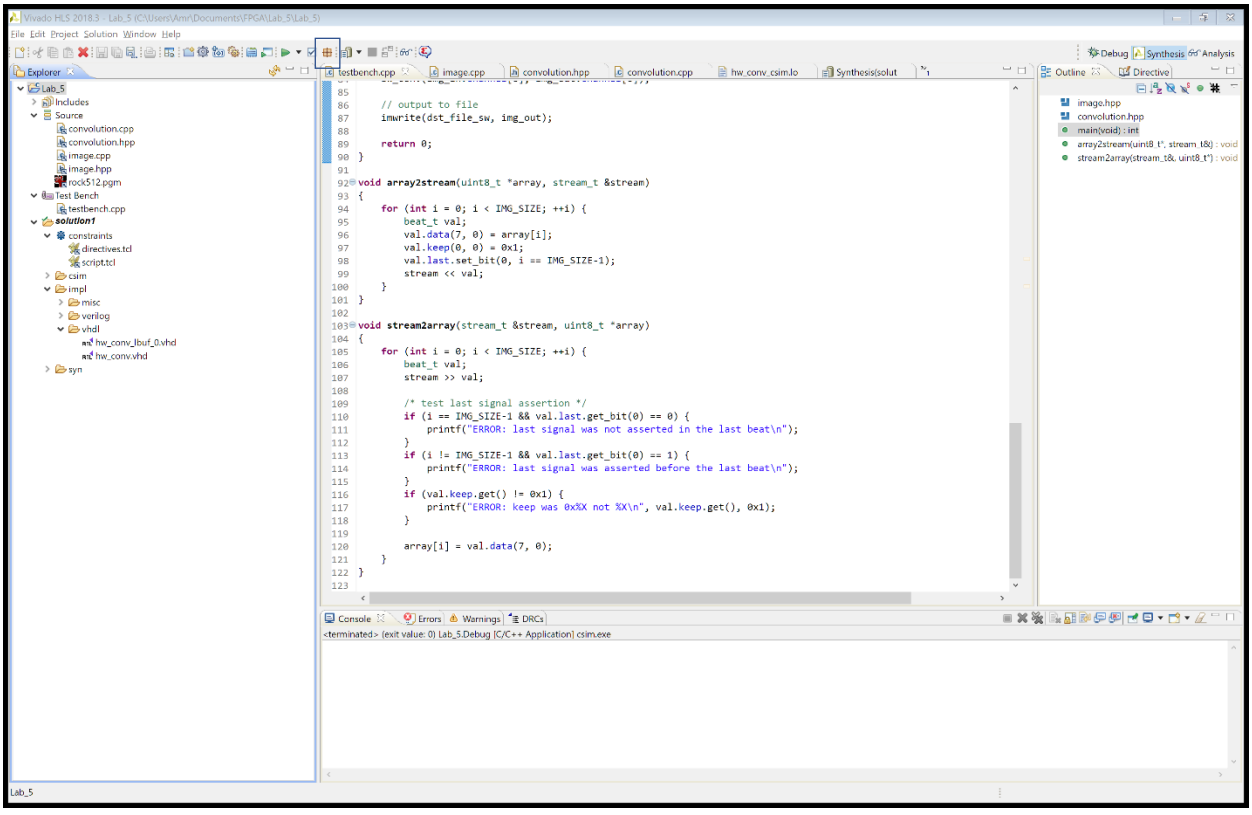


3. Generating an IP for your hw_conv function

After the whole process, you will see VHDL files generated by the HLS under solution1->impl->vhdl in the Explorer window. Please open these VHDL files and take a look. You might have different number of files depending on your implementation.

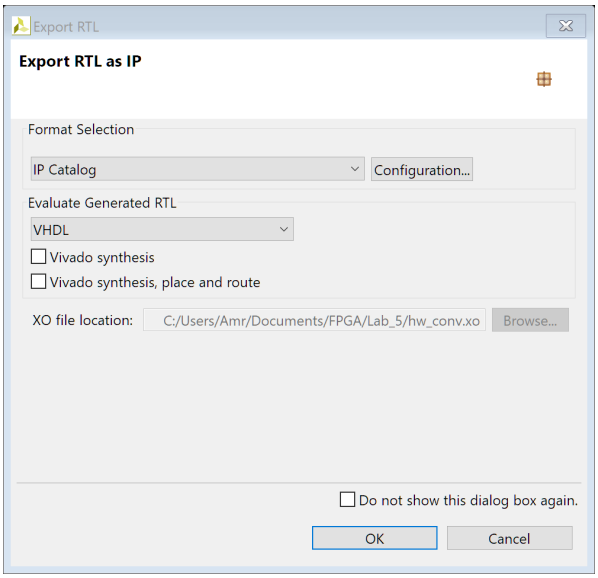


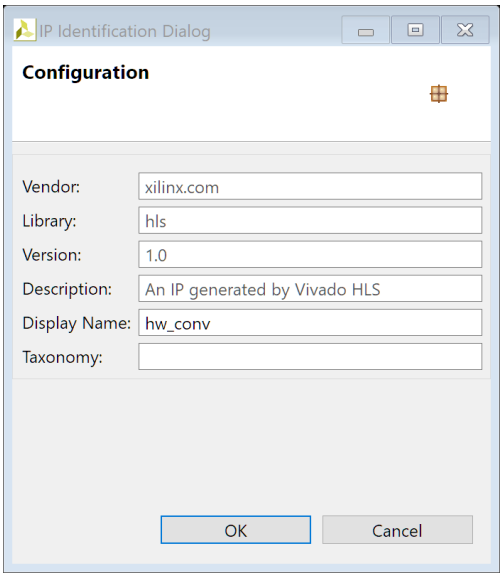
a. Click “Export RTL” button as labeled below to export RTL as IP.

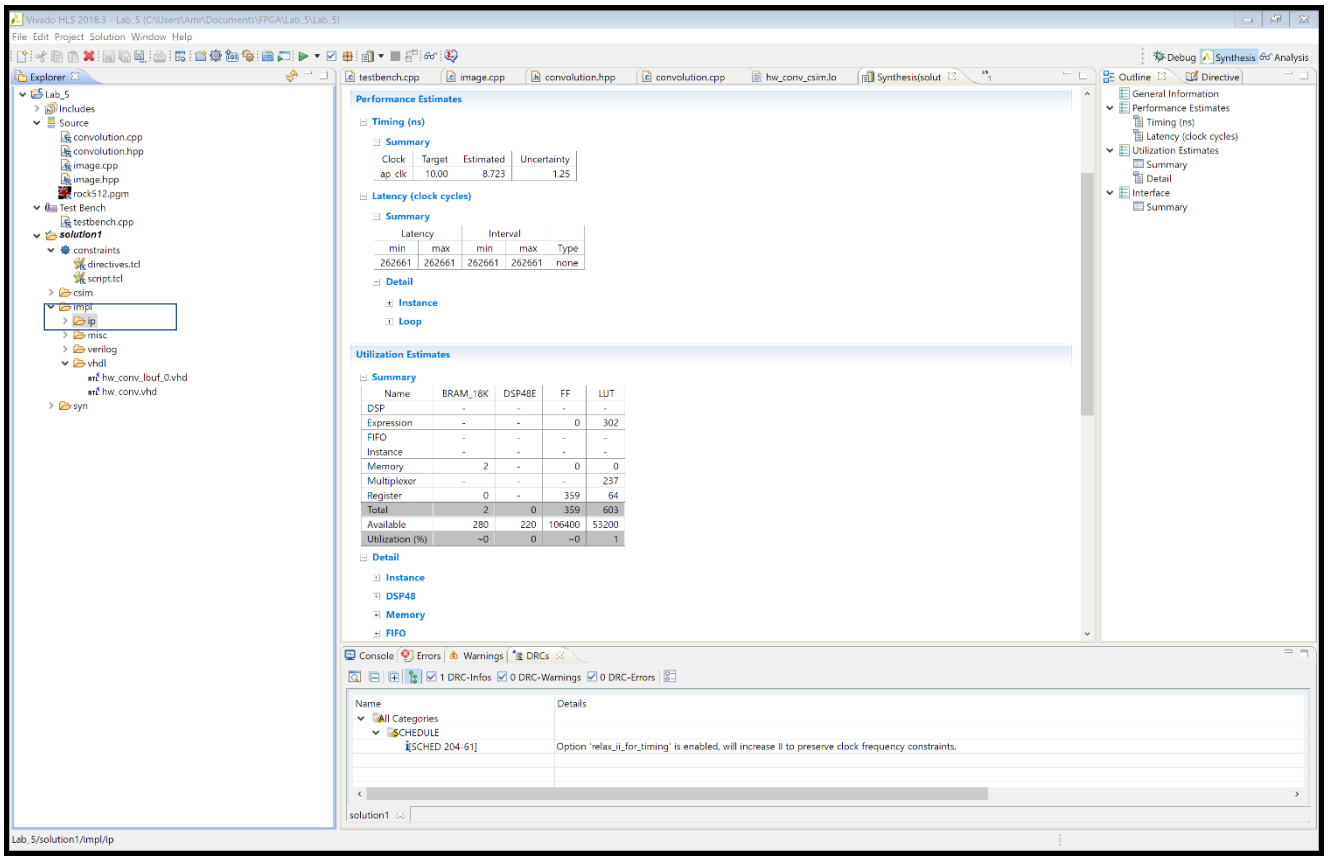


b. We will use VHDL as RTL. You can change the name of the IP by editing “Configuration” and editing the “Display Name” property. Click “OK”, and then it will generate the IP into the “impl”

folder. Now, you have the conv_hw accelerator successfully designed.





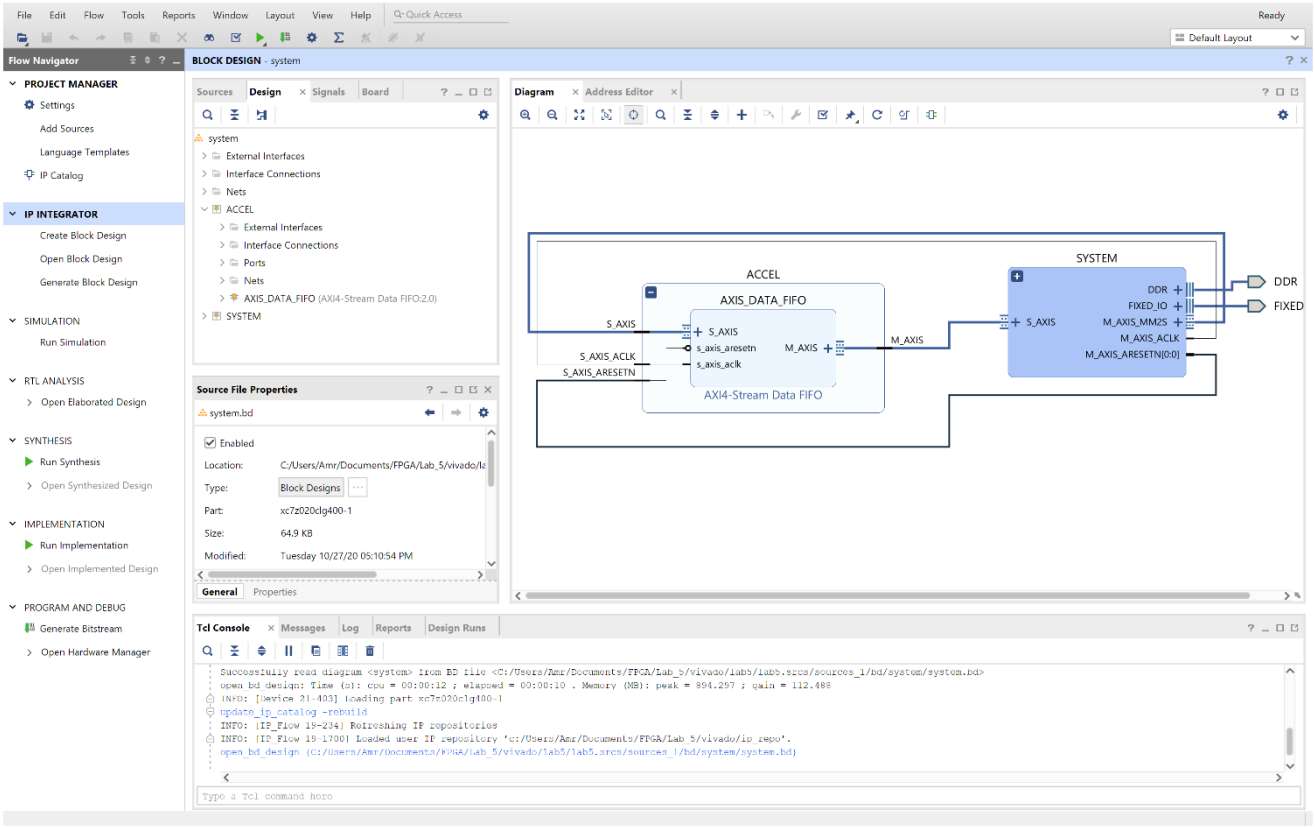


4. Final Bitstream Generation using Vivado

With generated IP, now we are ready to combine the accelerator with existing library modules such as DMA and ZYNQ together to implement the architecture to implement the architecture and deploy it on the board. In the provided "lab6_template" folder, a folder named "vivado" has a template Vivado project that has all the peripherals and components needed to generate the bitstream, except for the conv_hw generated IP. You will be using this project to include the ip you just generated and generate the bitstream.

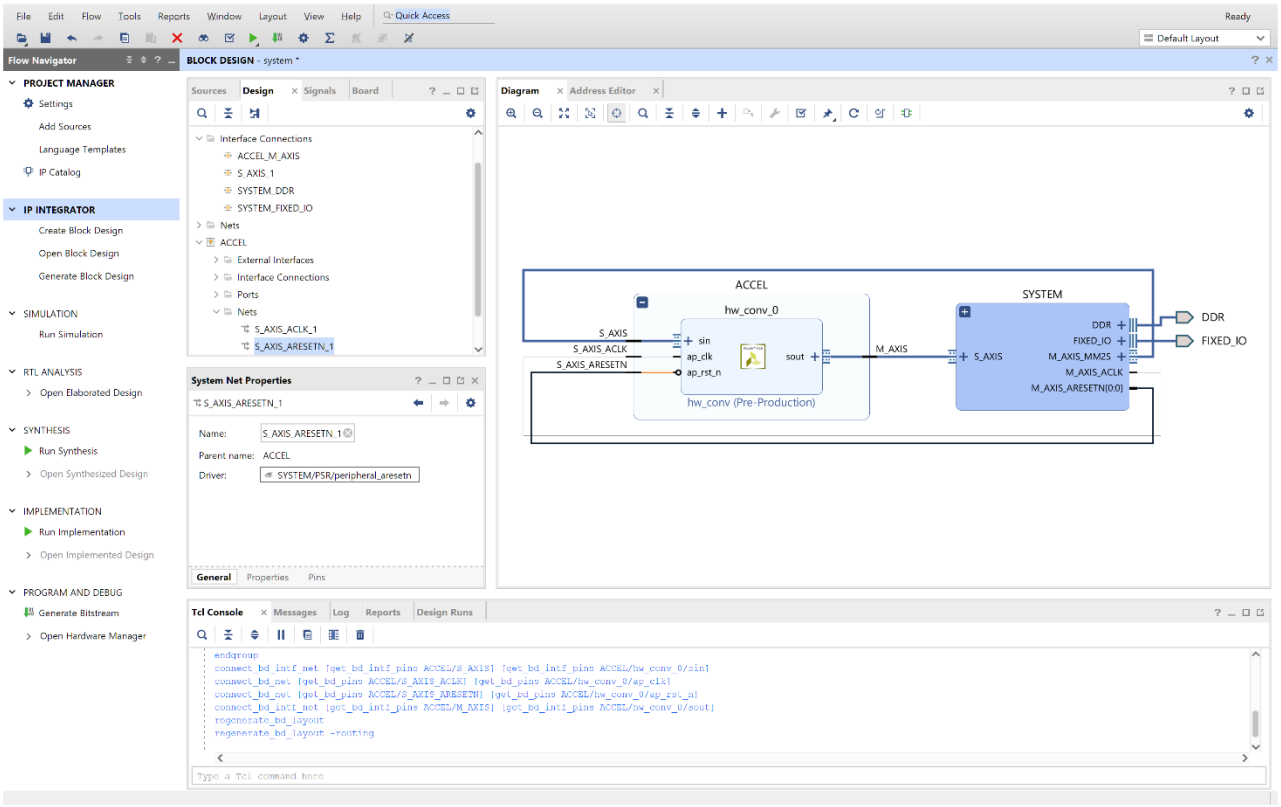
a. Copy the provided folder named “vivado” into the project directory you created for lab 6.

b. Open the project file under “vivado/lab6”. Open system.bd diagram. Expand the Accel hierarchy. This is what it looks like. The AXIS_DATA_FIFO is just a first in first out accelerator that we are going to replace with the generated conv_hw function.



c. We need to add the ip repo of the generated conv_hw. Add the following folder “your project directory\”solution name”\impl\ip” to your ip repositors directories.

d. Delete the AXIS_DATA_FIFO and replace it with the conv_hw component. The project should look like this. Note the icon in the middle of the component which indicates that it’s an HLS generated component.



e. Now, generate the HDL wrapper and the bitstream. If everything went well, you should be able to successfully generate the bitstream.

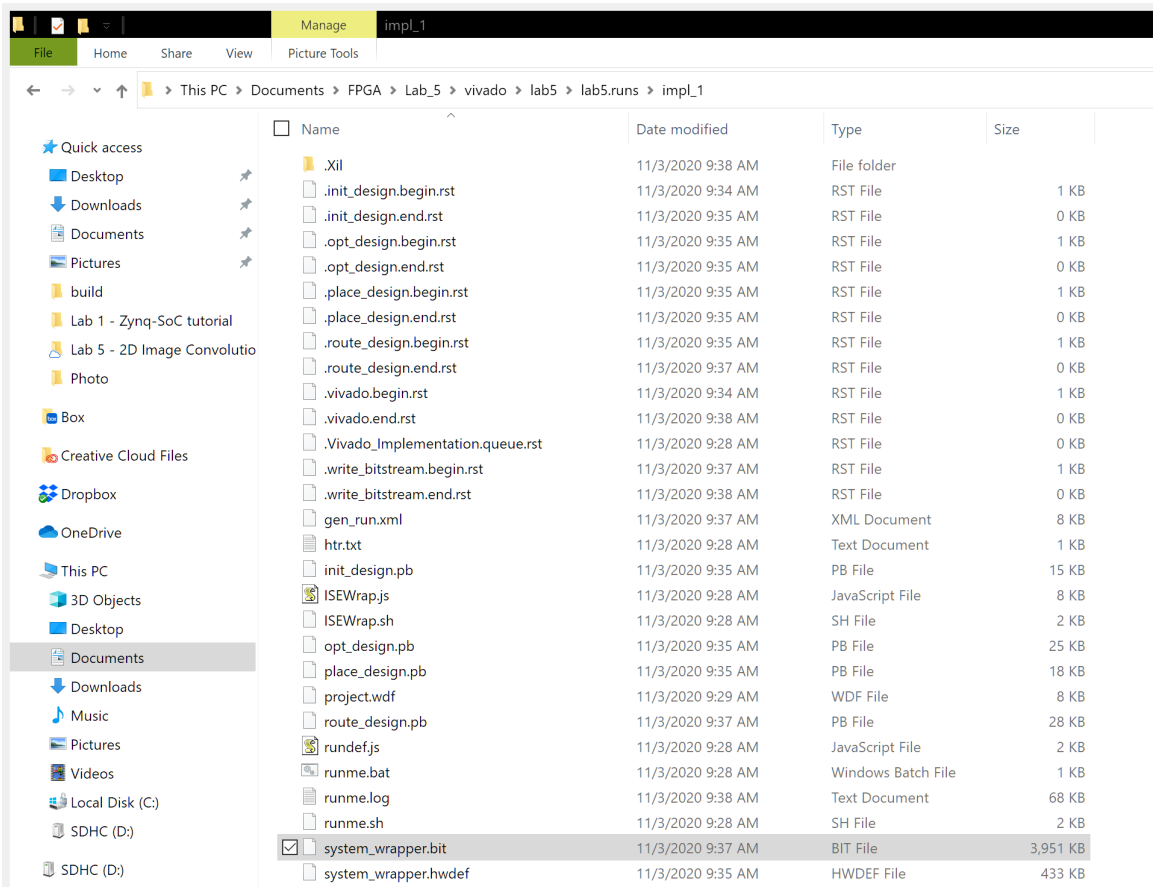
5. Running the accelerator on your FPGA board

For this part, you are going to use the sd card that comes with the kit, the usb cable, Putty, and the generated bitstream from Vivado.

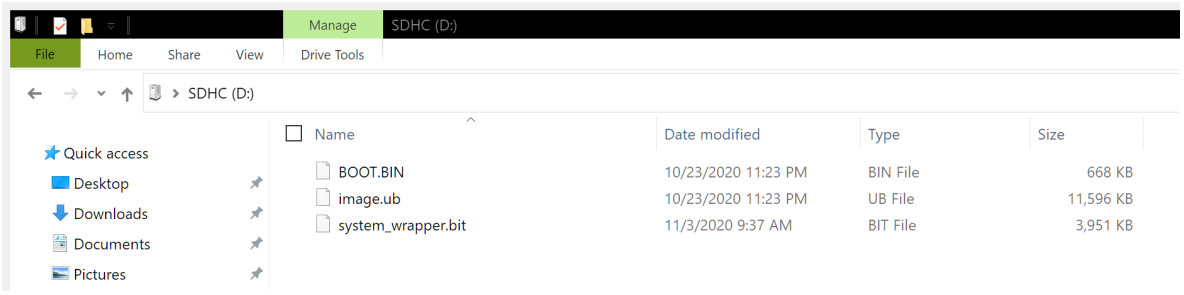
1. Use an sd card reader and the sd card that comes with you kit to format it. From Windows, right click on the sd card then click format, then start.

2. Copy the following two files “BOOT.BIN” and “image.ub” from the sd card folder provided under lab5_template, then past them directly under your sd card.

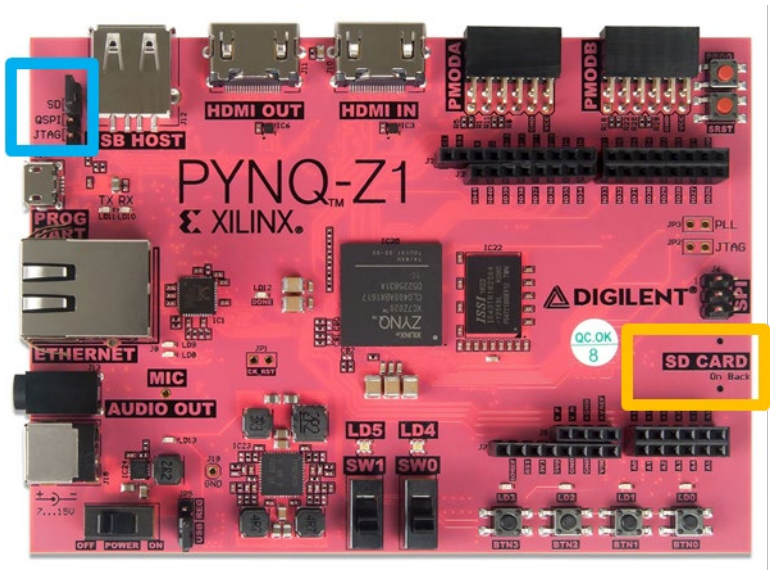
3. Navigate to your lab 6 Vivado working folder and copy the file called “system_wrapper.bit”. This is the generated bitstream. Paste this file also directly under your sd card. Follow the path in the following picture:



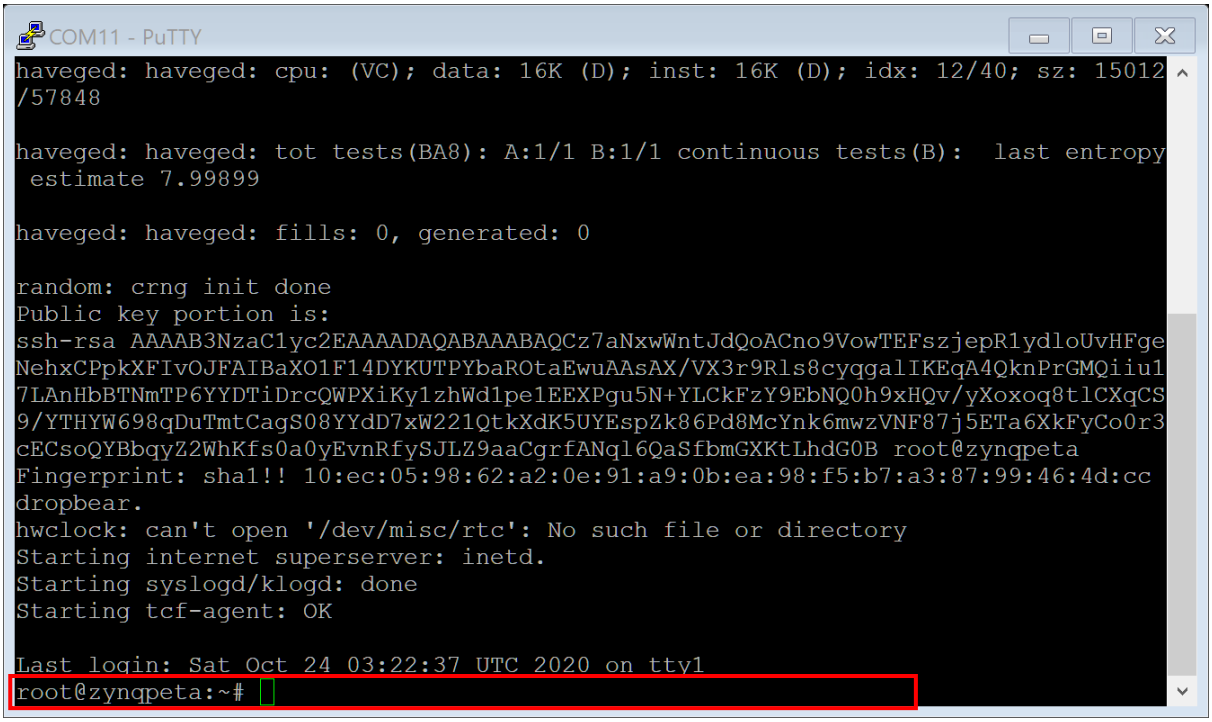
4. When you are done with these 3 steps, your final sd card directory should look like that:



5. Insert the sd card in the sd card slot in the board and switch J14 to SD.



6. Connect the board through USB cable, turn the board on, then wait till LD12 (Done) led turns green. Now start Putty. Note, you might a bunch of commands being executed on Putty, this is fine. Just wait till you see the following line. If you started Putty a while after the led turns green, then you might just see a blank screen.

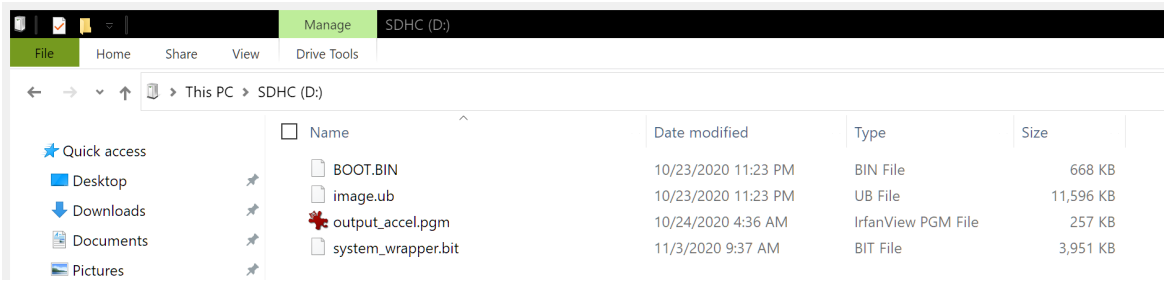


7. We are going to execute the following commands, one at a time, to instantiate the accelerator and save the image to the sd card. Any line that is in red is just a comment for you.

```
exit #DNE

# run CONV accelerator (input: rock512.pgm (512x512 Y), output: output_accel.pgm (512x512 Y))
axidma 1 /dev/axidma0 ./rock512.pgm ./output_accel.pgm 512 512 1

# sample output (when you run the above command, the execution time will be reported, take a note of that as you will be reporting it later on)
# time:  0.002669

# now we need to mount the sd card, copy the output picture to it to view it on the PC
mount /dev/mmcblk0p1 /mnt
cp output_accel.pgm /mnt/output_accel.pgm
sync
umount /mnt
```

8. Take a screenshot of your Putty Window. You can now turn off the board, extract the sd card and plug it back to your PC reader.

9. If everything went well, you should find the convoluted picture in the contents of your SD card.



10. Add a section in your report for part 5 (see below). You should be reporting the following:
    a. A screenshot showing the communication done over Putty and you executing these commands.
    b. Report the time your accelerator took to execute.
    c. Include the conv pic from your accelerator, and the one generated form the HW simulation. Add them next to each other, they should be identical.
    d. If anything went wrong, reflect on that and say what you think went wrong and how you might fix it.

What to do:

1. You should design the conv_hw function in the provided convolution.cpp template. You will be needing to watch the lecture recording along with this document in order to get the job done.
2. Synthesize the Top Function using both a ring buffer and a shift register for the line buffers. Generate the Synthesis Report. You will be writing a report for this Lab so make sure to get screenshots that would cover all parts of the report.
3. Simulate the c testbench and have a look at the generated picture, the hw and sw generated pic should be matching. Those pictures will be included in the report as well.
4. Generate the ip for conv_hw and use the provided Vivado template to generate the bitstream. A screenshot confirming that you have successfully generated the bit stream would be included in the report.
5. For part 5, please add a section in this report, following the description in step 10 in “Running the accelerator on your FPGA board”

Deliverables:

1. You will be delivering all the VHDL and C codes that you have written, including the design files, testbenches, and the C codes. The submission will be on Canvas Assignment for Lab 6. You can just Zip the top Vivado project folder that has all the files that was mentioned above. (please make sure that it has all the files required).
2. Check-offs will be done on the due date for this assignment. The rubric for the check-offs will be provided later.