**IMAGE PROCESSING & EDGE DETECTION USING VIVADO TOOL:**

**Step1: Line Buffer Design :**

* One line buffer will be going to store one line of the image.
* It is some kind of RAM where data will be initially filled from the original image and data will be read out & processed later.
* Each pixel is only one byte**.**

Writing the Verilog code for line buffer and then we have to perform simulation:

Code\_Link 🡪LineBuffer.v code

Simulation of line Buffer :

Note: Xilinx provide different kind of Ips ,but as per our requirement we have to write our own IP because we can customise them as per our need.

**Step2:** **Design of module which will do multiplication and summation operation**.

[MAc module🡪 Multiplication & Accumulation]

* This module will get the pixels & it will also get the kernels & it will basically multiply pixels with corresponding kernel value & add them together.
* Writing the verilog code for the module.

Code\_Link 🡪conv.v code

**Step3:** **Design of Control Logic**.

* Design a module which instantiate all the line buffers and these multiplexers ,

Which control storing the data into line buffers as well as which control sending data from these module to the MAc module

* Here we will write our imageController verilog code.

Code\_Link 🡪imageControl.v code

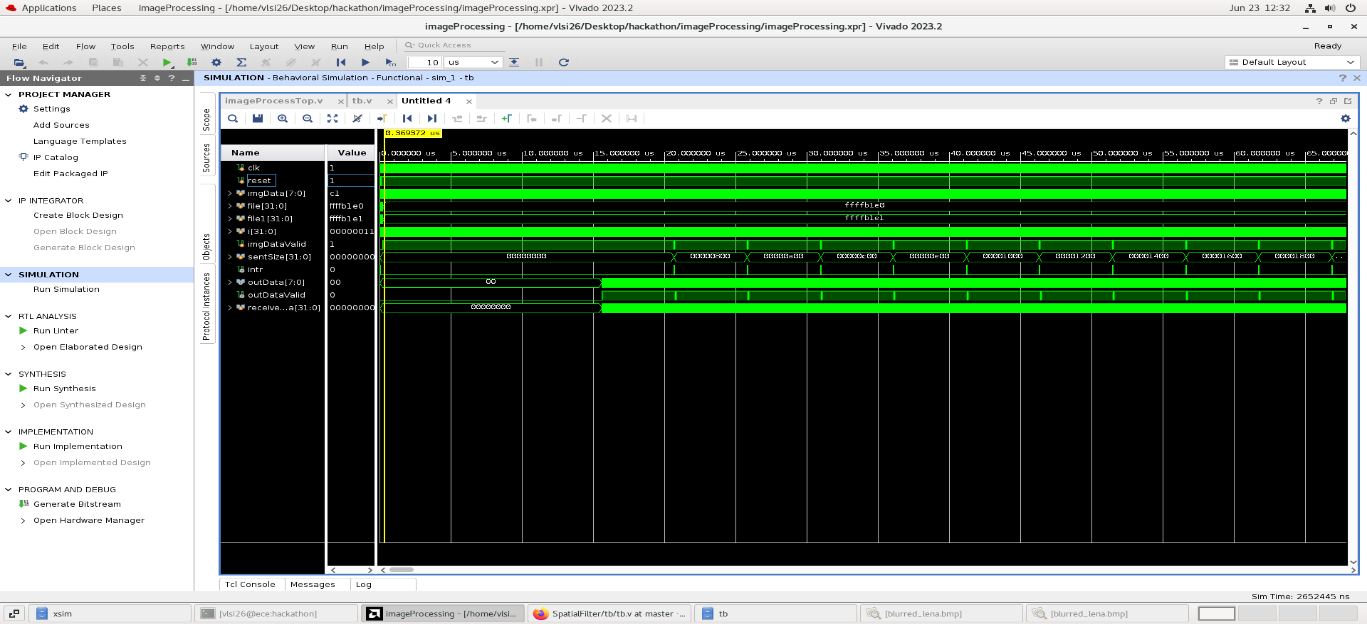
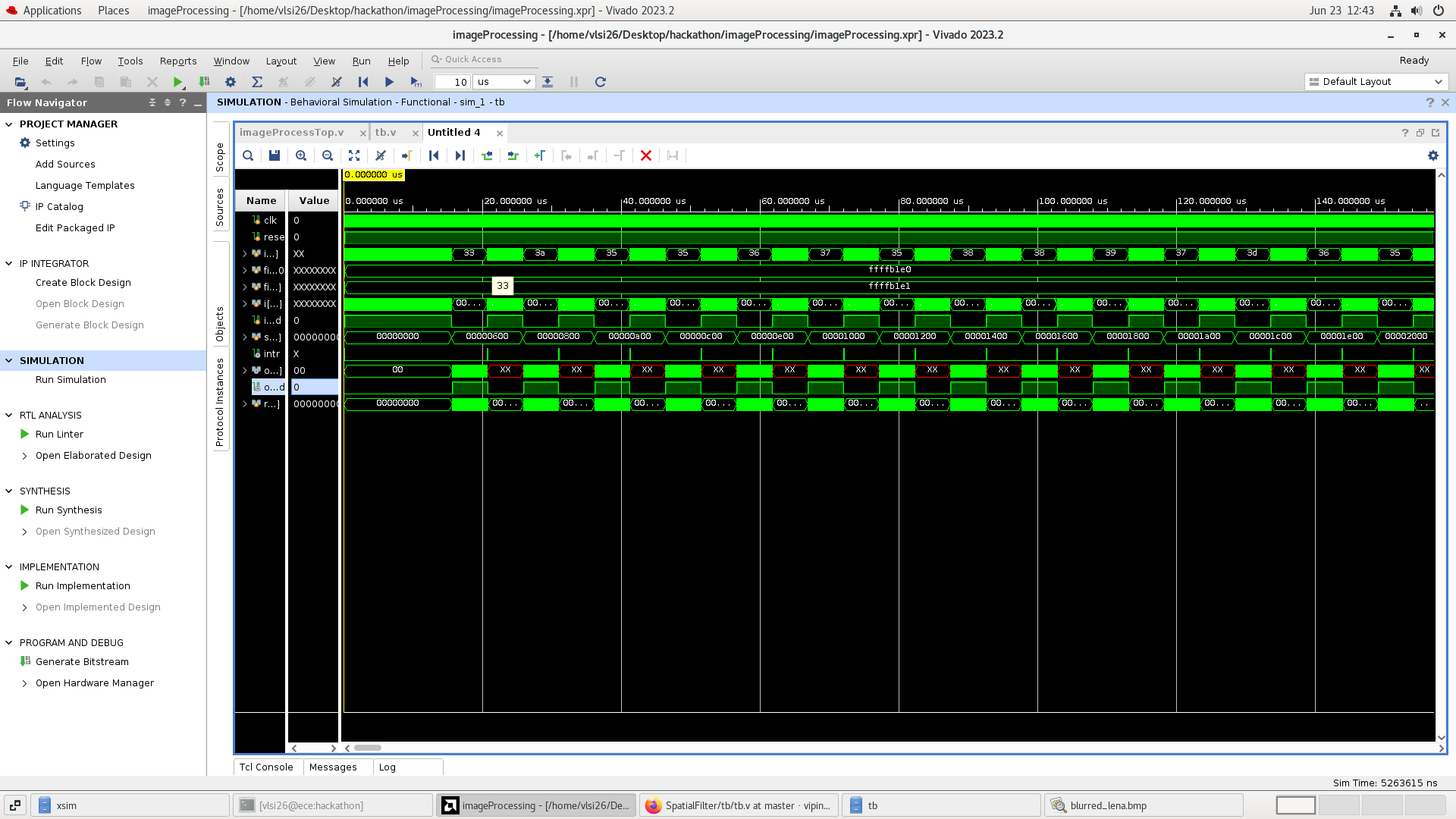
**Step4: IP Packaging:**

* We will combine all the modules we have developed till now and we will complete hardware design for our IP. So that we can package our own IP.

Code\_Link 🡪ImageProcessTop.v code

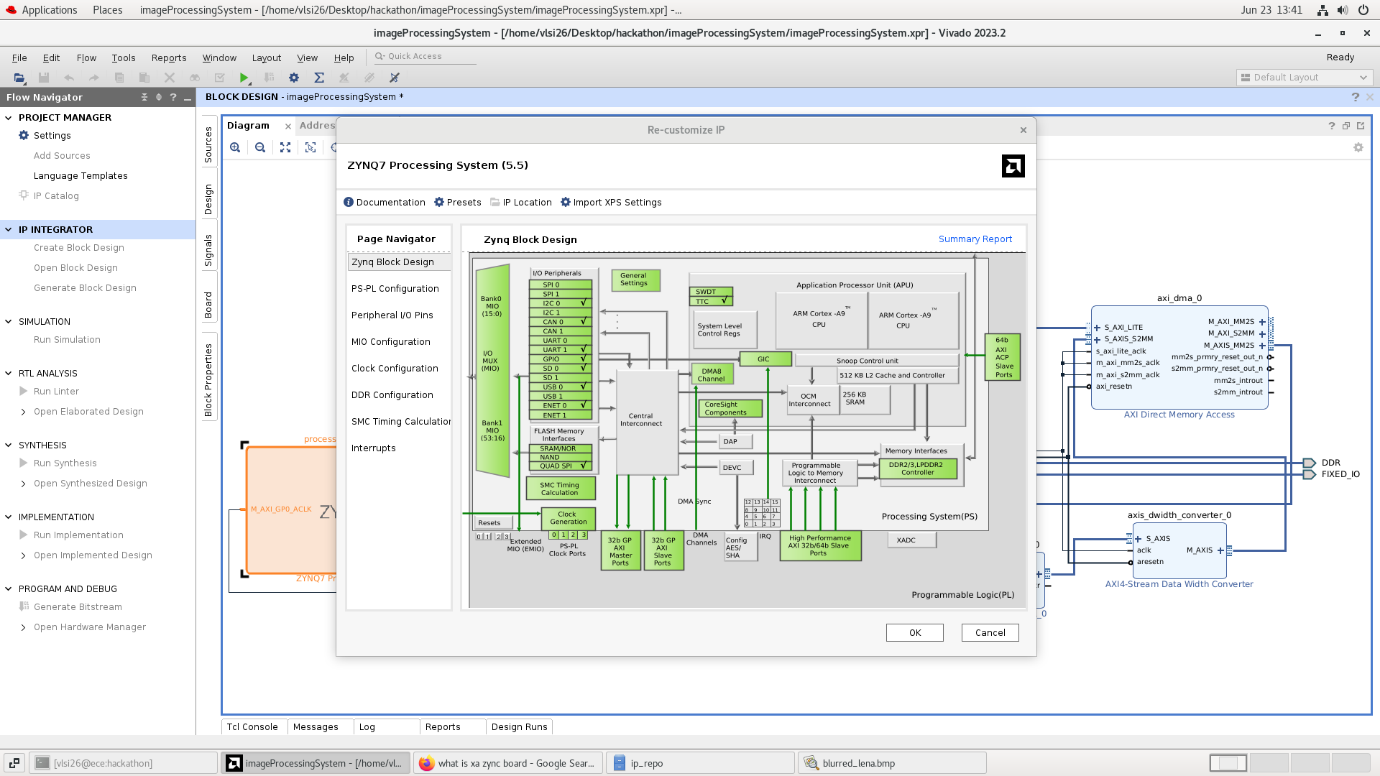
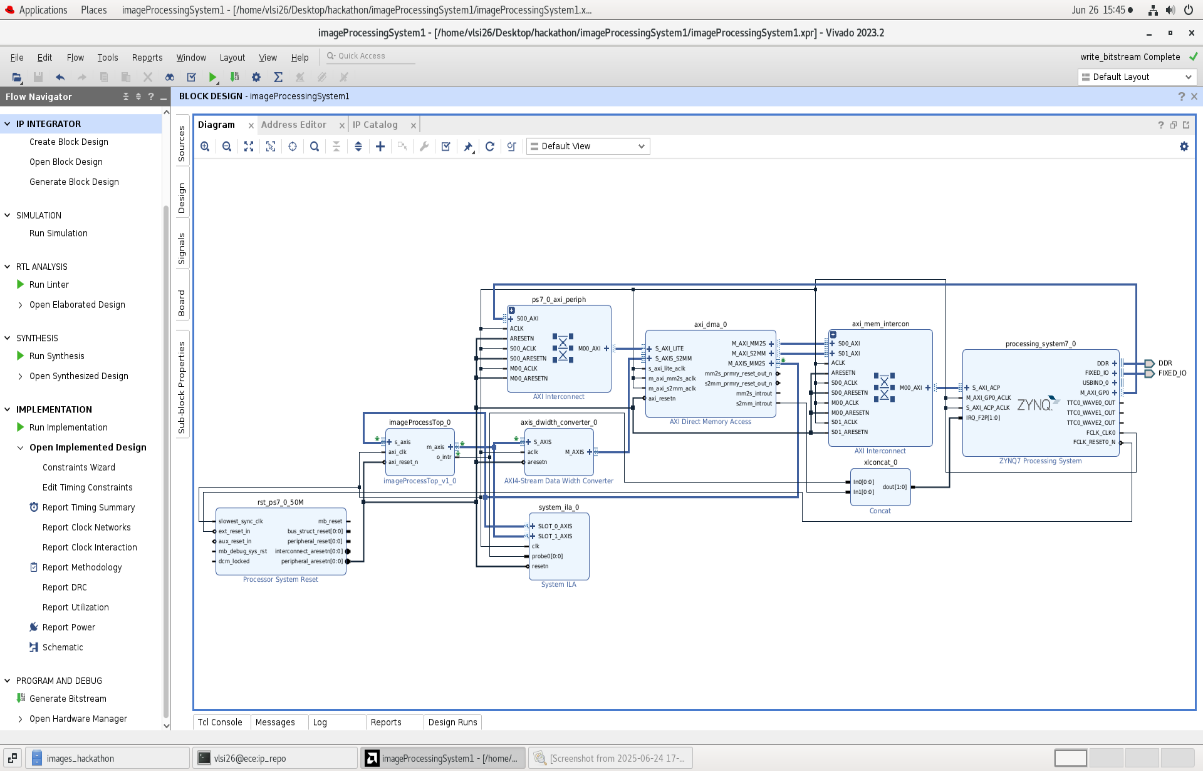
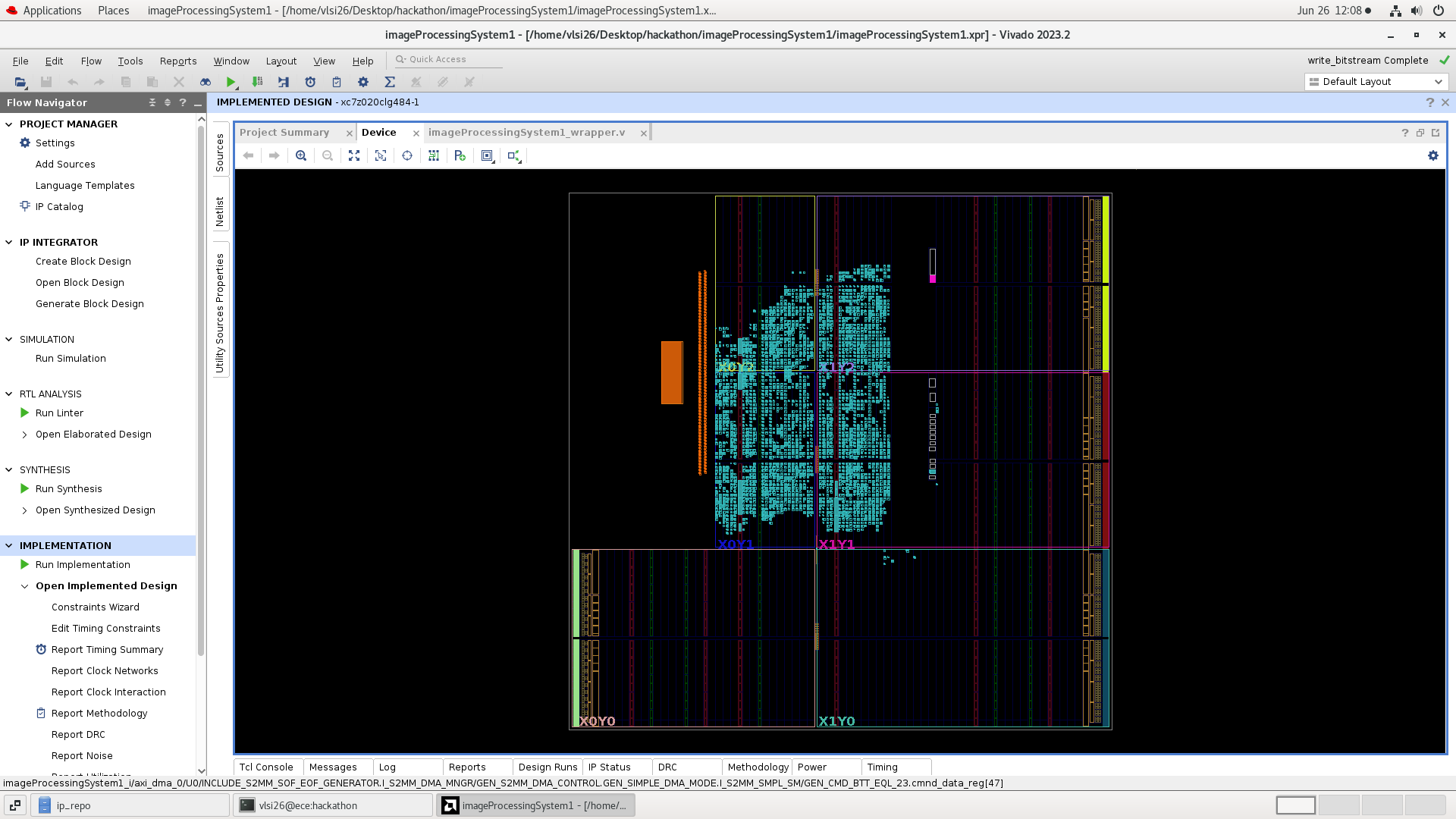
* Then click IPCatalog 🡪 select FIFO Generator 🡪 we are going to use AXI stream FIFO 🡪click OK 🡪click Generate.
* Instantiate this into my ImageProcessTop.v file 🡪 Code\_Link 🡪ImageProcessTop.v code
* Now we will convert it into IP AXI format . So that we can use this IP in our block design.
* Click on Tools 🡪Create Package IP 🡪Package your current project 🡪Next🡪Finish

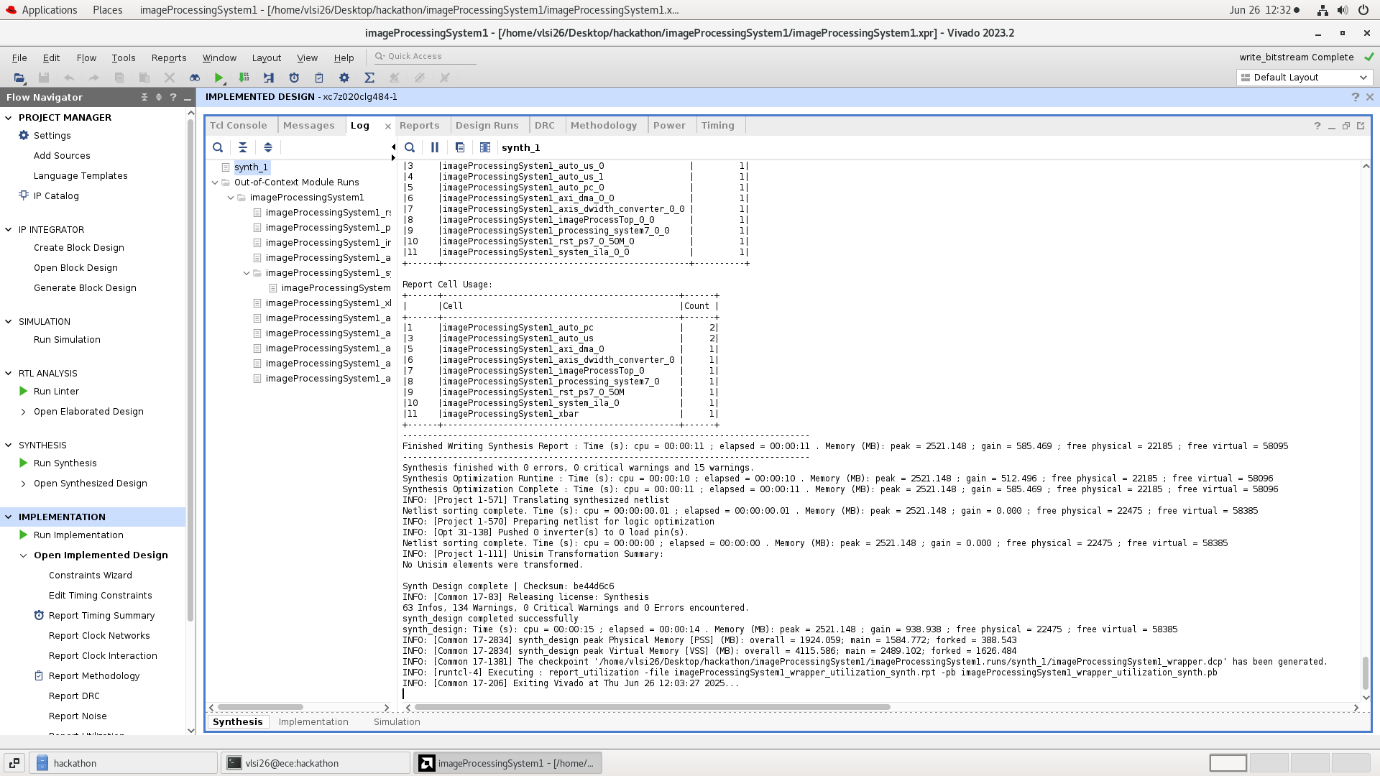
**Step 5: Simulation.**

* Till now we have packaged our IP but not yet verified.So we have to do simulation and we cannot verify it by waveform here.
* We have to realy send an image(.bmp format) & check whether the output is coming properly or not.
* So we have to write our own test bench [which will read the image & send it to our IP and it will receive the data coming from the IP and I will be able to see the actual output.]
* Code\_Link 🡪 tb.v file
* After that go to settings 🡪simulation 🡪OK🡪run simulation

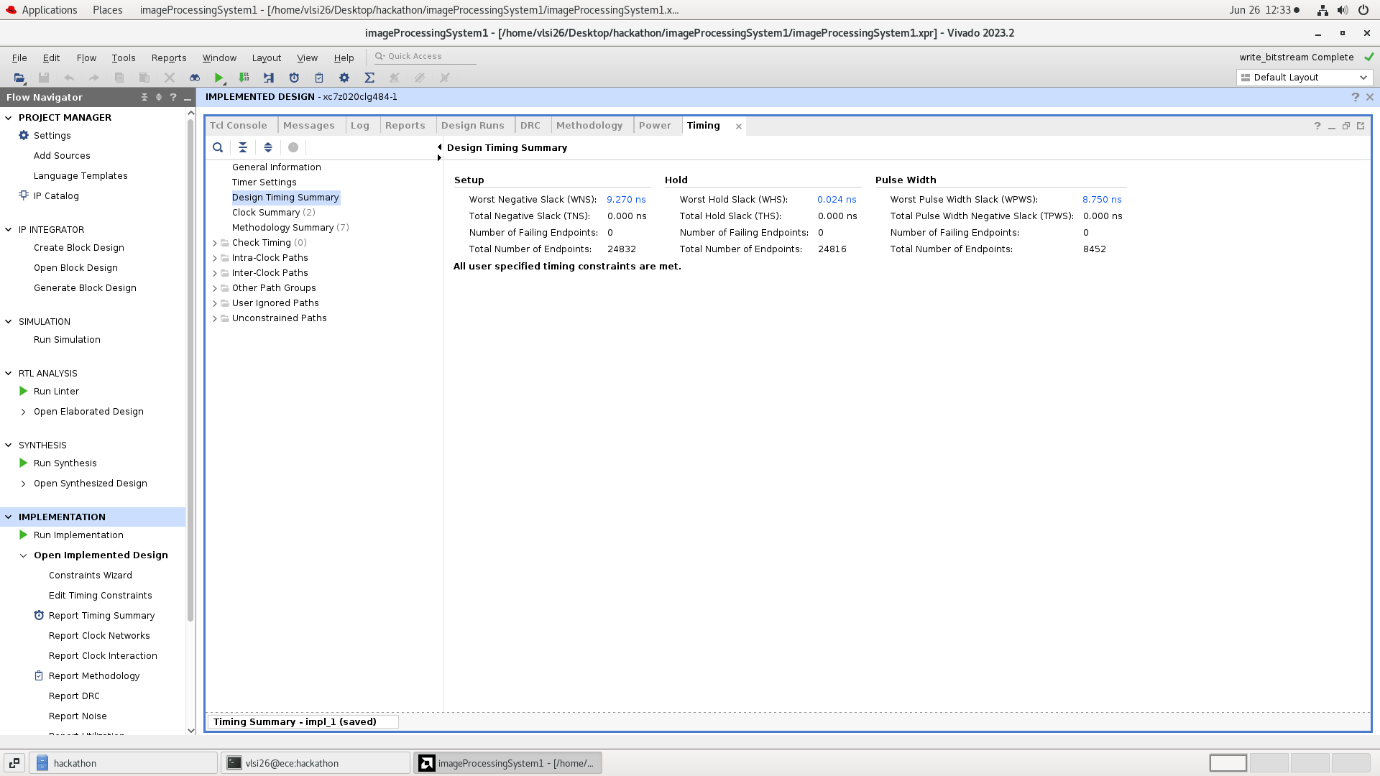
 Input image .bmp file  output blurred image .bmp file

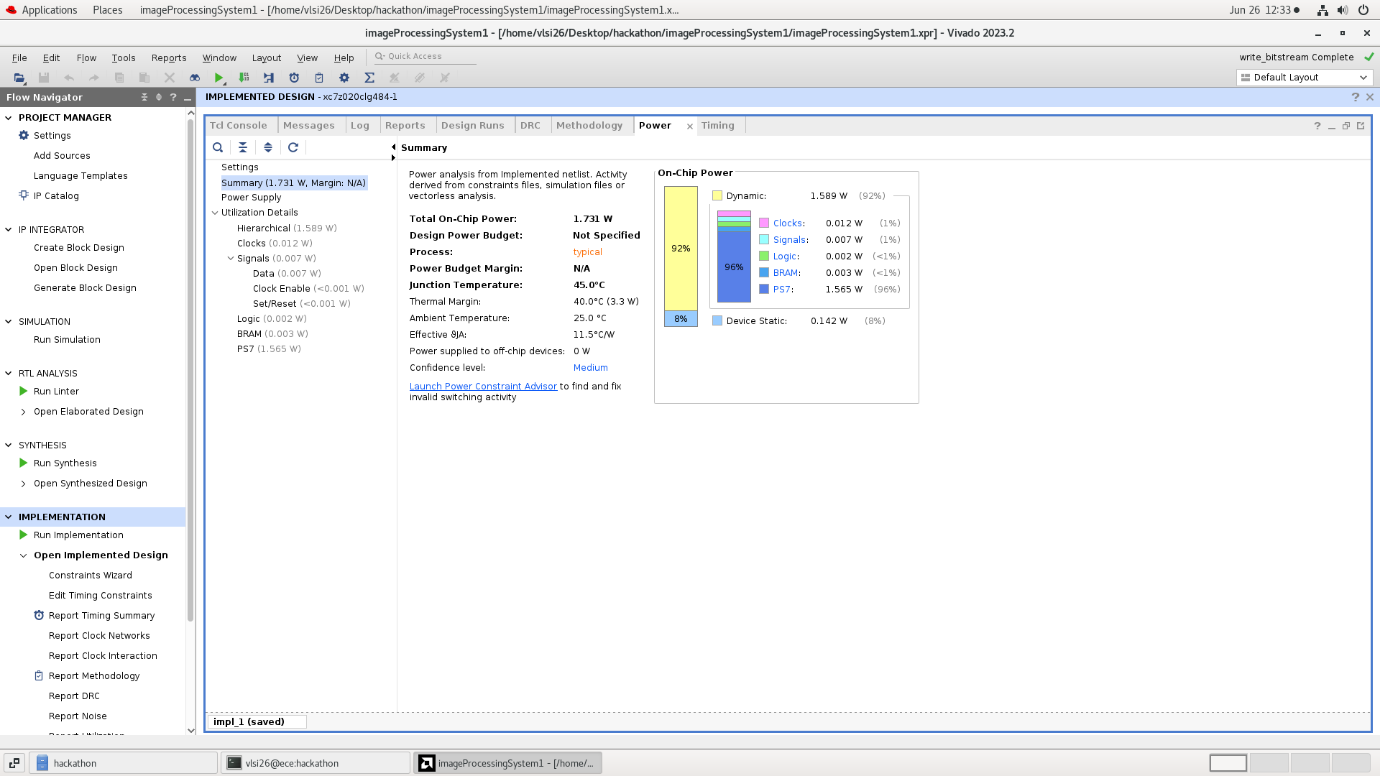
**Step 6:** **System Integration**

* We are going to build a system which is going to use this IP core.
* Create new project 🡪create block design .Select the following IP:
* Zynq Processing System | our IP(ImageProcessTop\_v1) |AXI DMA
* Then run connect automation | after that select some more IPs like AXI4 - Data width Converter| AXI Interconnect | Concat IP 🡪 then again run connect automation
* Verify design
* Generate Wrapper [Sorce 🡪select ImageProcessingSystem🡪create HDL wrapper]
* Generate BitStream

**Log window: showing successful synthesis completion**

**Timing Checks: Setup time & Hold Time**



**Power Analysis from the implemented netlist****:**

**Step7:** **Software Development & Hardware Evaluation**

* We have completed the hardware design part of our system which uses our image processing IP. Next we have to go to software part of the system in order to verify it on hardware.
* An image is simply a big array of numbers , so instead of sending image to external world we can create a static array of that big numbers & make it a part of our software source code.
* Verilog can do filehandling also. So I will modify my testbench(tb.v) file to get the header file (imageData.h).

Link\_of\_imageData.h\_file

* Export hardware & launch vitis IDE tool.
* Create a new application project & import file imageData.h file .Next go to source & create file imageTest.c file.
* Link\_of\_imageTest.c\_file

Now we will test it on hardware.

* Go to vivado & program the device

Now actually the data is coming & the entire processing is over . It is sending back the processed data .Now we will get our image filtered output similar to simulation.

Edge Detection Image :

Image must be stored in .bmp format



The edge detected image using Solebel

