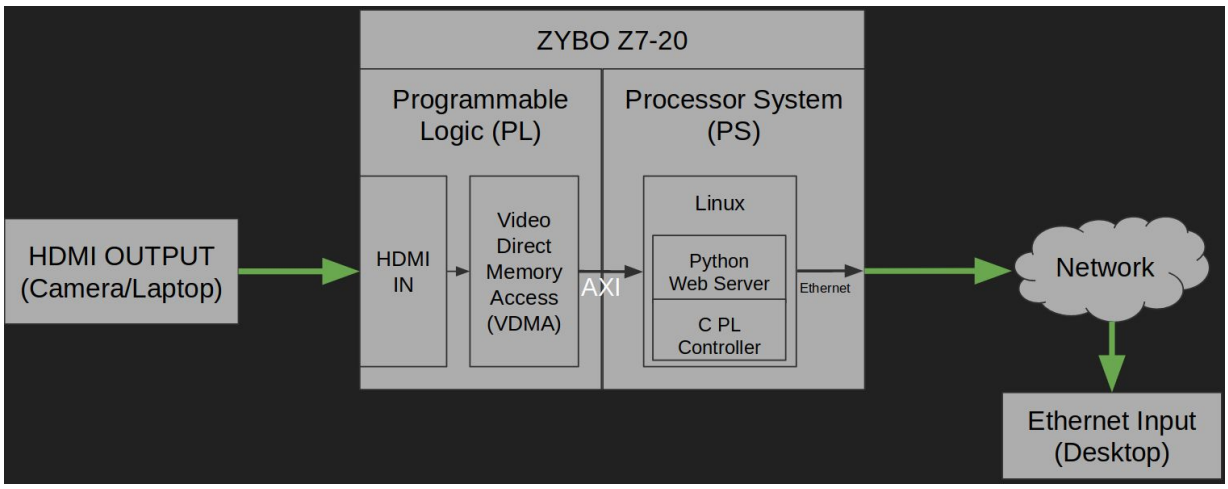
Austin Owens
12/9/2019
HW/SW Coprocessor Design w/ FPGAs

Summary

This project includes taking in video stream input via HDMI input or the Video Test Pattern Generator (VTPG) Xilinx ip core, sending that video stream to the PS side using VDMA, and displaying the video stream output with a webserver written in Python on the ZYBO Z7-20 development board. The overall goal of this project is to create an image processing testbed to facilitate computation-heavy image processing on the PL, as well as image processing on the PS side that may be too difficult to do on the PL side.



Requirements

- Create VDMA engine in PL that flows data to the PS
- Input data going through VDMA engine to PS will come from:
    1. generated data within the PL (for debugging) utilizing the Xilinx's Video Test Pattern Generator (VTPG) IP core
    2. an external source, such as HDMI input from a computer or an HDMI camera
- Incorporate Integrated Logic Analyzers (ILAs) for observing signals live time in PL
- Configure and initialize VTPG registers through AXI bus in software on PS
- Configure and initialize VDMA engine registers through AXI bus in software on PS
- Build custom rootfs that contains various software packages for development testing (devmem2, python, etc.)
- Build boot.bin with first stage boot loader capable of automatically loading the bitstream on fabric
- Build web server on PS to display captured data from VDMA engine as a picture the Zybo's IP address is queried in a web browser

Deliverables

- Pictures of the data being forwarded to the web browser that originated from Xilinx's VTPG (VTPG -> VDMA -> Linux)
- Pictures of the data being forwarded to the web browser that originated from laptop output (Laptop HDMI Output -> DVI to RGB data -> VDMA -> Linux)
- Pictures of the data being forwarded to the web browser that originated from camera output (Camera HDMI Output -> DVI to RGB data -> VDMA -> Linux)
- Screenshot of signals captured with ILA's to show data passing through the PL
- Screenshot of data being mapped from physical to virtual memory and show via stdout on Linux shell console
- Screenshot of Vivado block diagram project
- Video recording

Demonstration

The following section shows pictures of the deliverables.

VTPG -> VDMA -> Linux

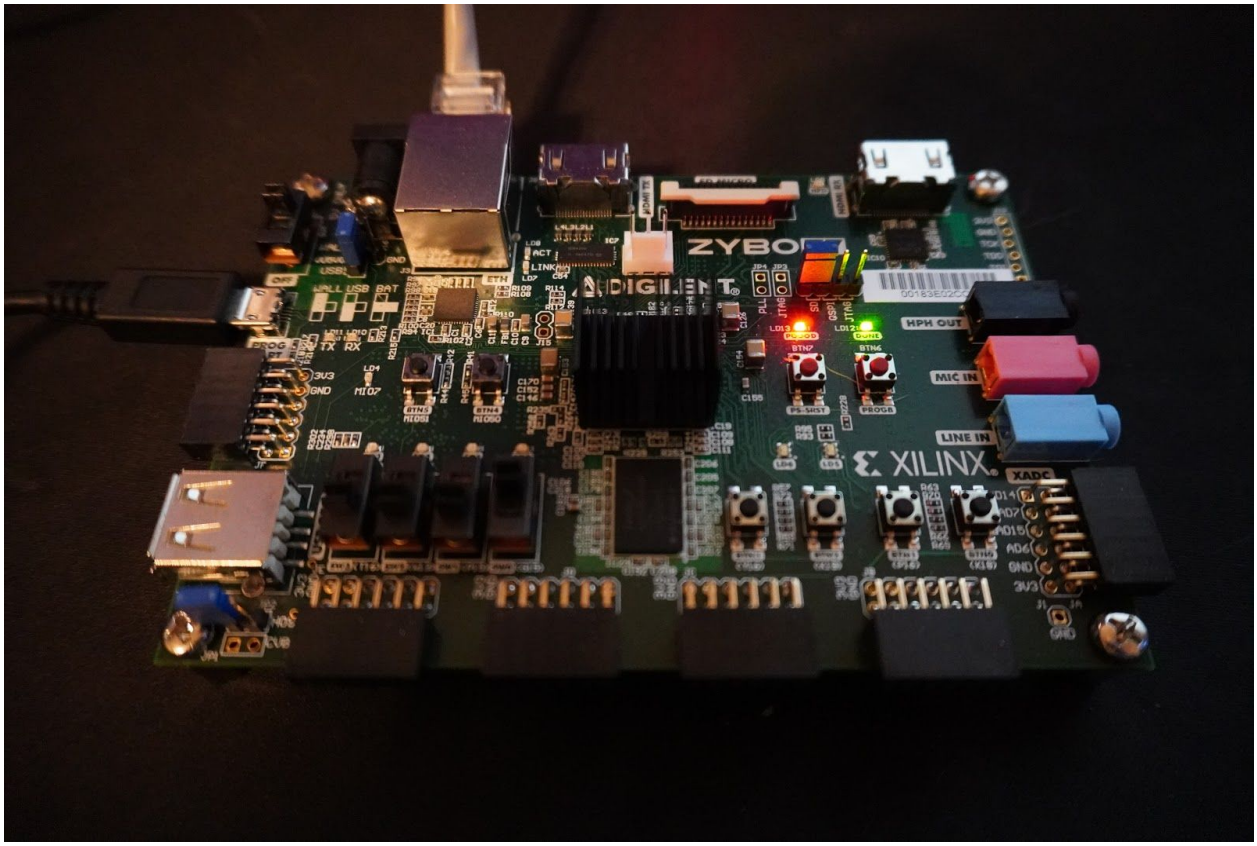


Figure 1: In order to use the Video Test Pattern Generator (VTPG) inside the PL, you must first put the first switch in the **on** position.

```
root@zybo-zynq7:~/img_proc# ./img_proc.elf 1920 1080 3
WIDTH: 1920 | Height: 1080 | Pixel Length 3


Available Commands:
'p' to print section of frame buffer
'v' to turn on VTPG
'q' to quit
>> v
Started VTPG

Available Patterns:
'p': Passthrough
'rgbhr': RGB Horizontal Ramp
'rgbvr': RGB Vertical Ramp
'tr': Temporal Ramp
'r': Red
'g': Green
'b': Blue
'k': Black
'w': White
'cb': Color Bars
'zp': Zone Plate
'tcb': Tartan Color Bars
'ch': Cross Hatch
'cs': Color Sweep
'vhr': Vertical Horizontal Ramp
'chkr' Black and White Checker Board
'ps': Pseudorandom
'dpcr': Display Port Color Ramp
'dpbwv': Display Port Black and White Vertical Lines
'dpcs': Display Port Color Square
>> tcb
```

Figure 2: Controlling VTPG to output Tartan Color Bars (TCB) test pattern.

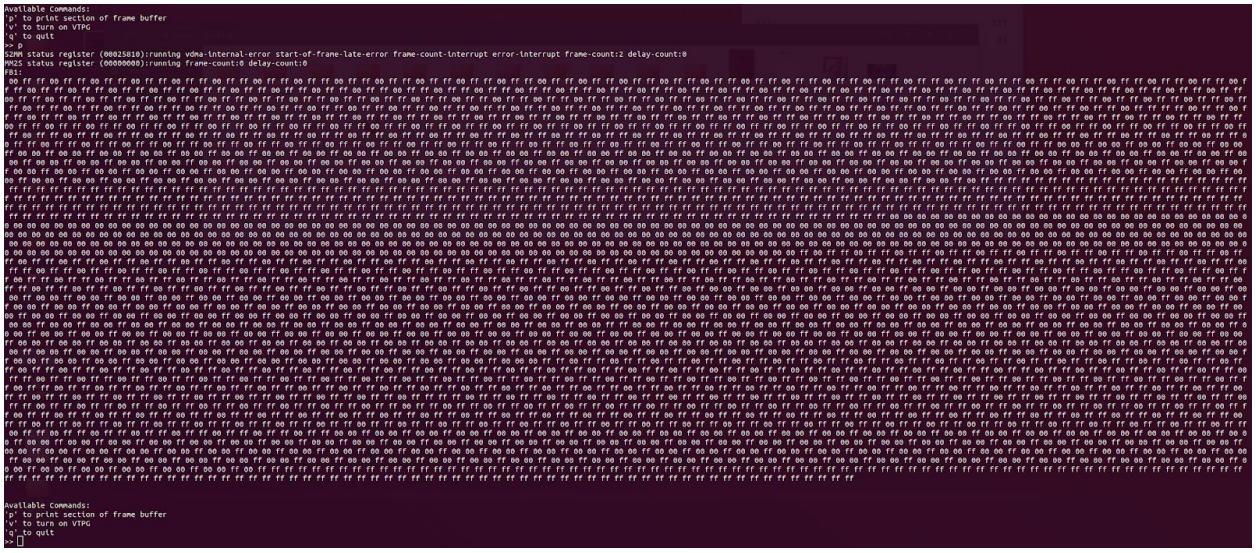


Figure 3: Shows snippet of VTPG output via VDMA to PS when TCB test pattern is selected for the VTPG.

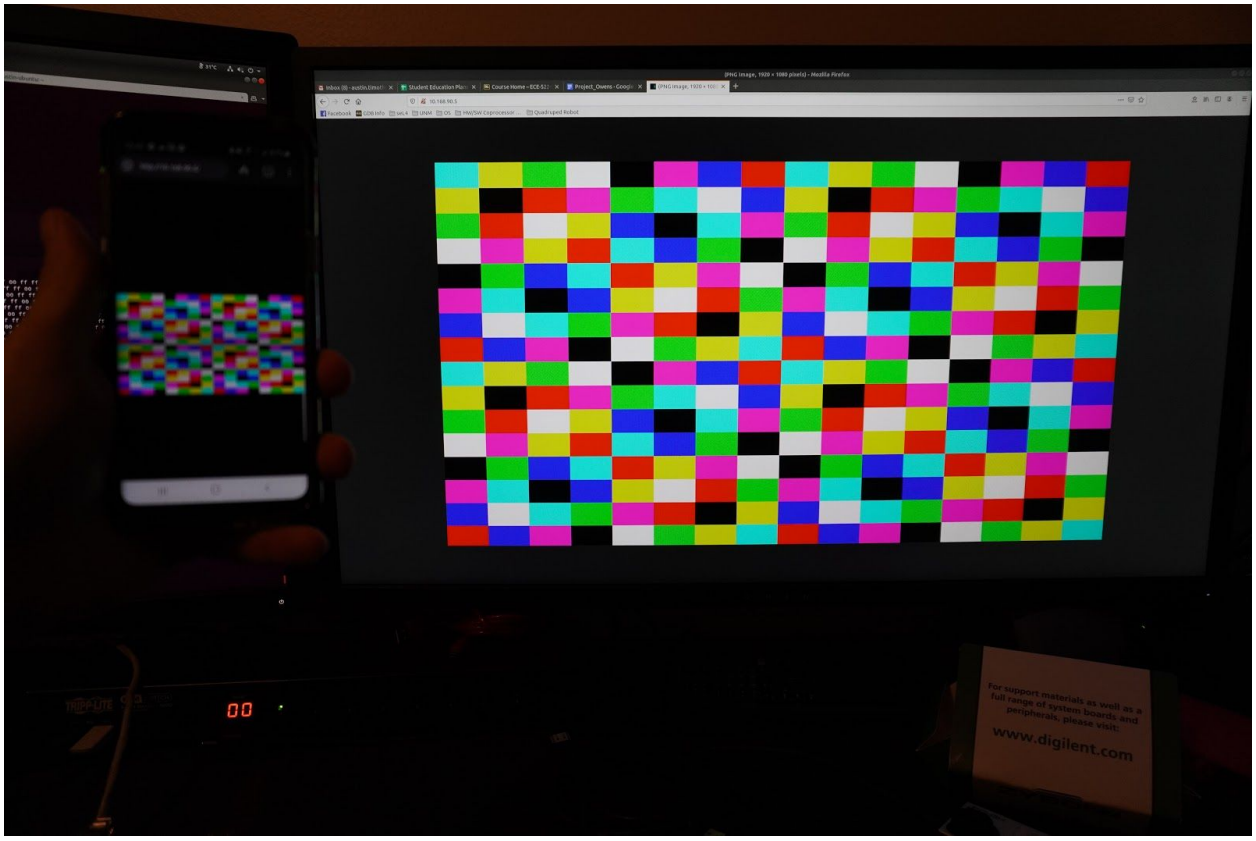


Figure 4: TCB pattern from VTPG being displayed via python web server on the PS side. Showing multiple devices on the network entering the IP address of the Zybo in the web browsers URL to get VTPG output.

Figure 5: Controlling VTPG to output a black and white checkerboard test pattern.

Figure 6: Shows snippet of VTPG output via VDMA to PS when the black and white checkerboard test pattern is selected for the VTPG.

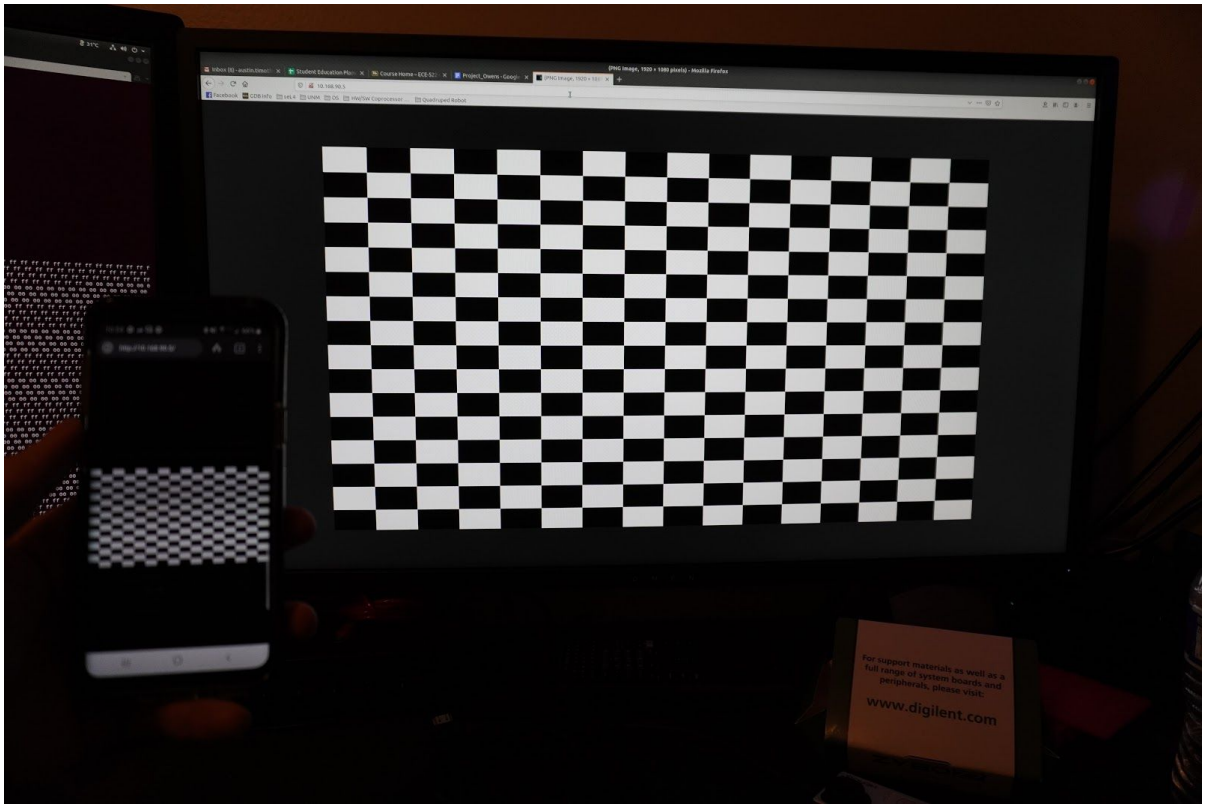


Figure 7: Black and white checkerboard pattern from VTPG being displayed via python web server on the PS side. Showing multiple devices on the network entering the IP address of the Zybo in the web browsers URL to get VTPG output.

Laptop HDMI Output -> DVI to RGB data -> VDMA -> Linux

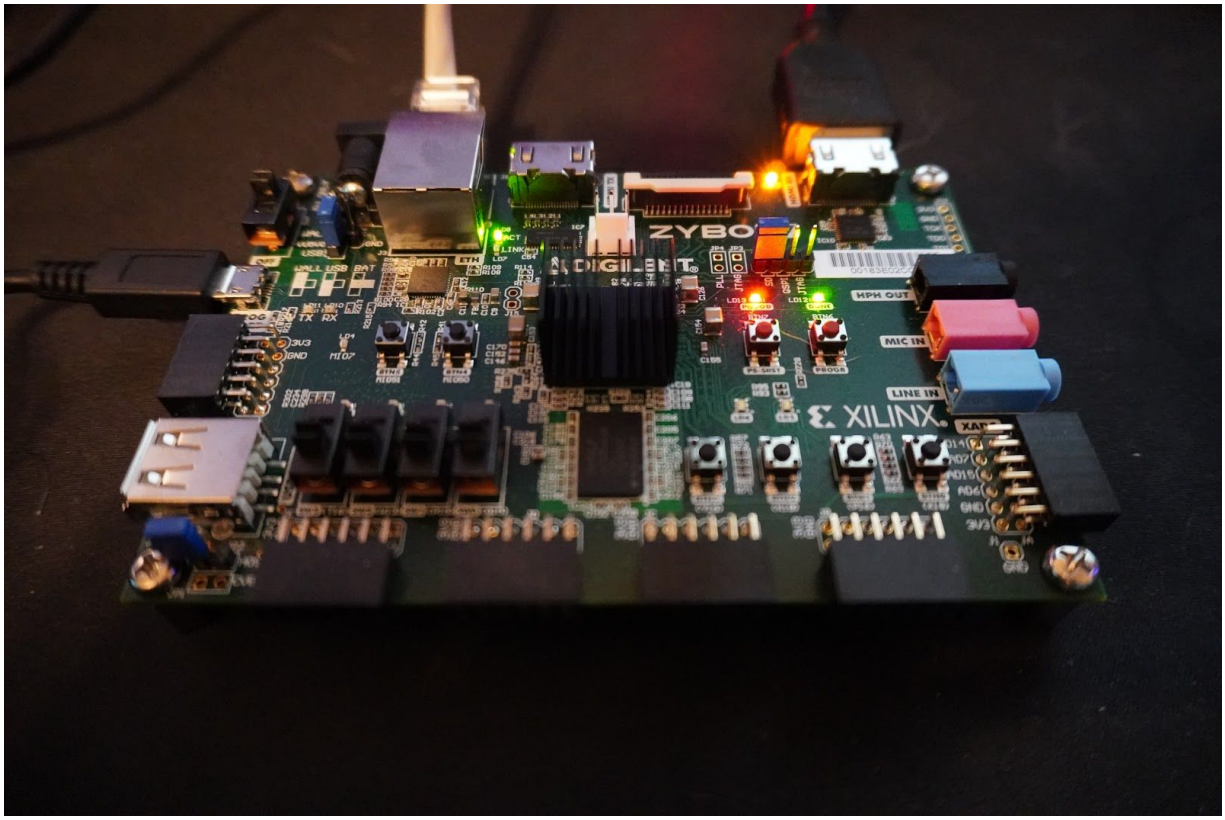


Figure 8: In order to feed HDMI input into the Zybo board, you must first put the first switch in the **off** position.

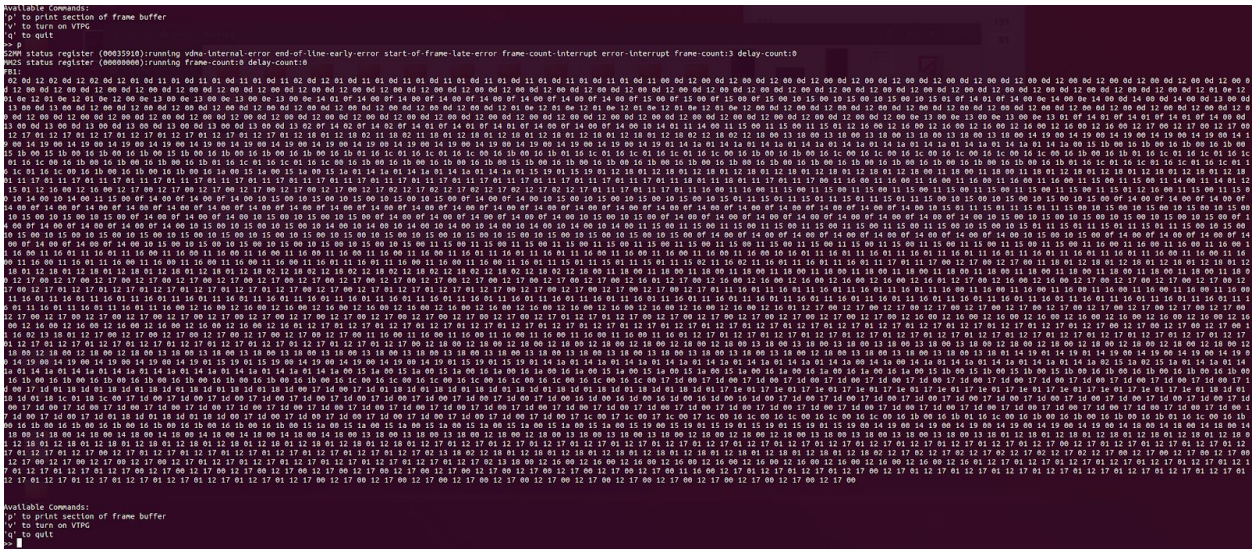


Figure 9: Shows snippet of HDMI RX input via VDMA to PS when the laptop is connected.

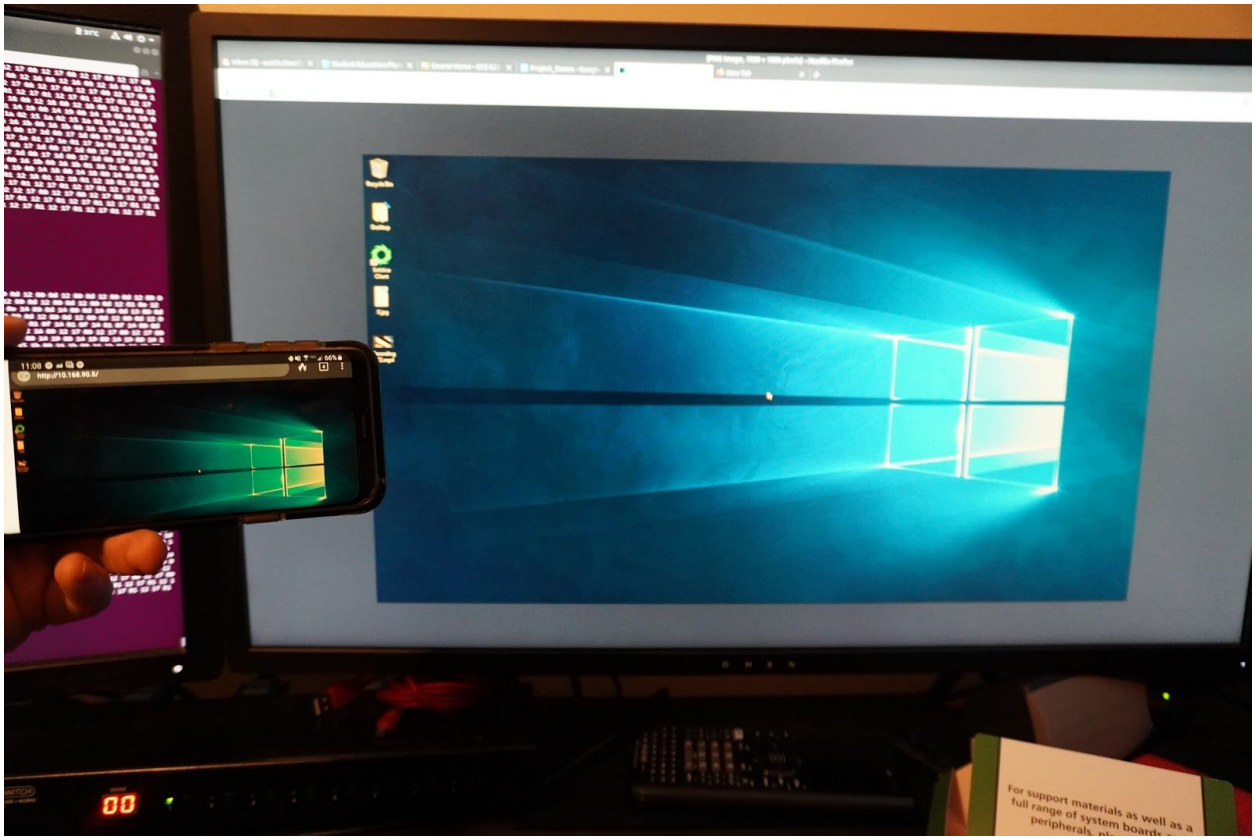


Figure 10: Laptop screen being displayed via python web server on the PS side. Showing multiple devices on the network entering the IP address of the Zybo in the web browsers URL to get what is on laptop screen.

Camera HDMI Output -> DVI to RGB data -> VDMA -> Linux

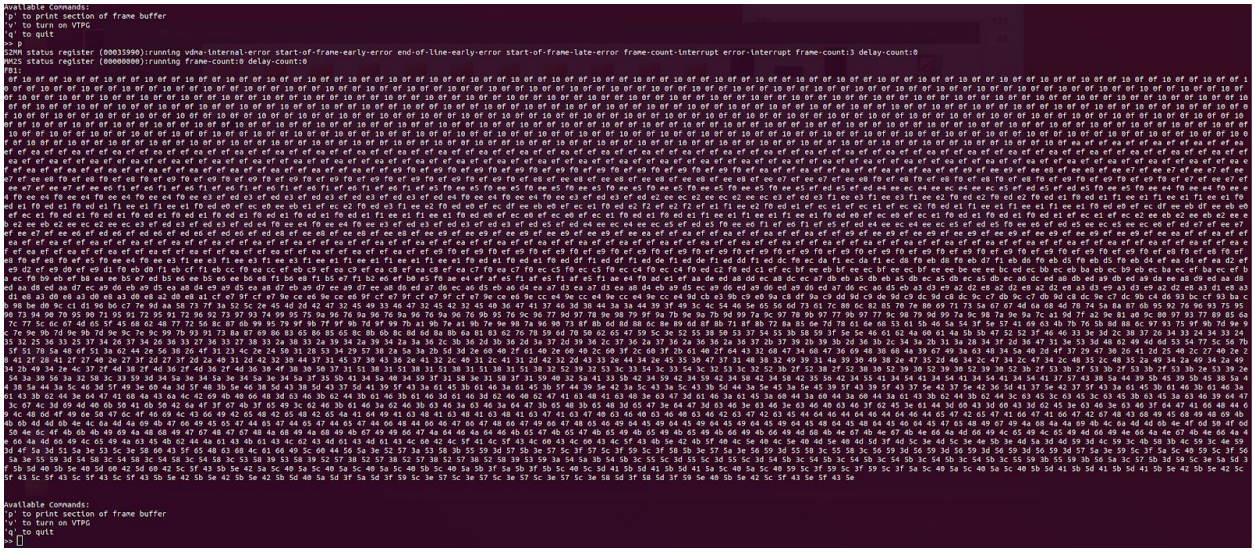


Figure 11: Shows snippet of HDMI RX input via VDMA to PS when GoPro camera is connected.

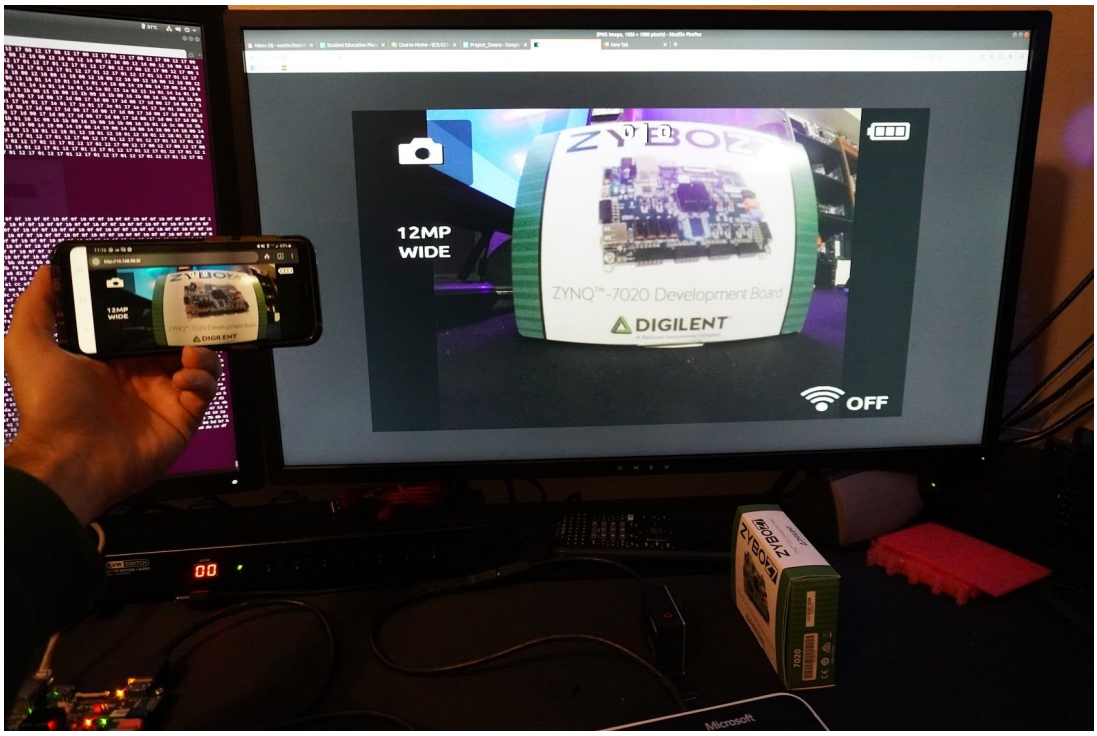


Figure 12: GoPro camera video stream being displayed via python web server on the PS side. Showing multiple devices on the network entering the IP address of the Zybo in the web browsers URL to get what is on GoPro camera.

Integrated Logic Analyzers (ILAs)

ILAs there were used over the course of the project to help with debugging.

Figure 13: ILA located within Digilent IP core (dvi2rgb).

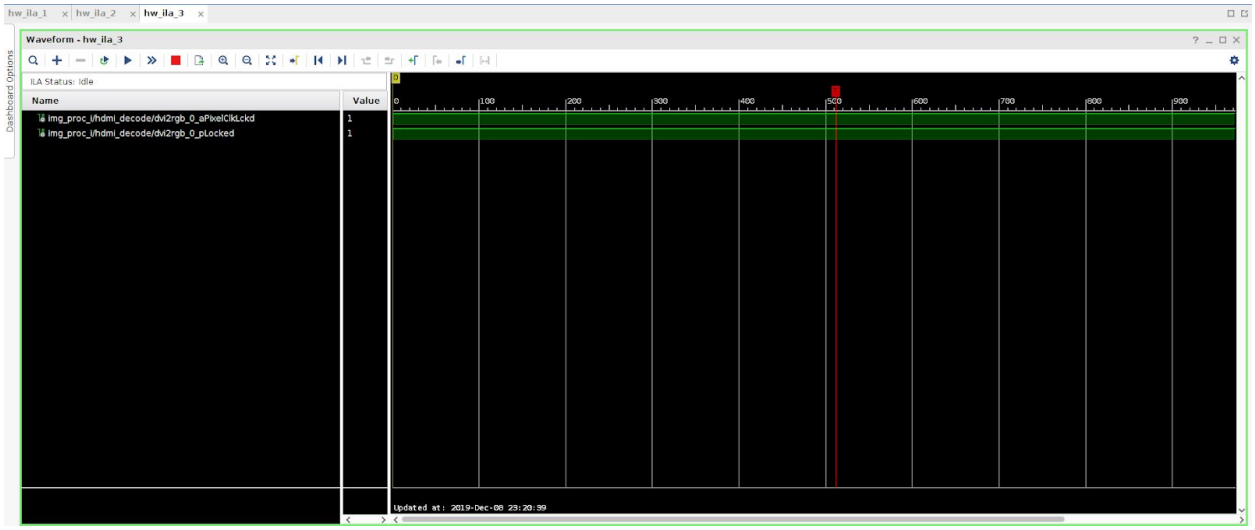


Figure 14: An ILA core I added to further troubleshoot when the dvi2rgb IP core has locked onto the clock.

Vivado Block Diagram

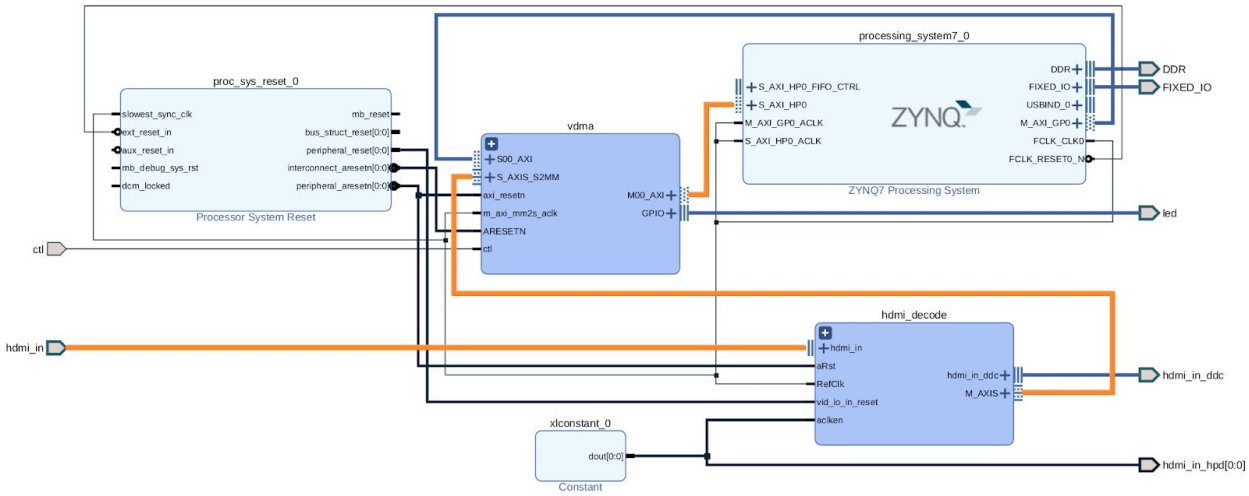


Figure 15: High-level overview of system. Highlighted line is the path of the video stream.

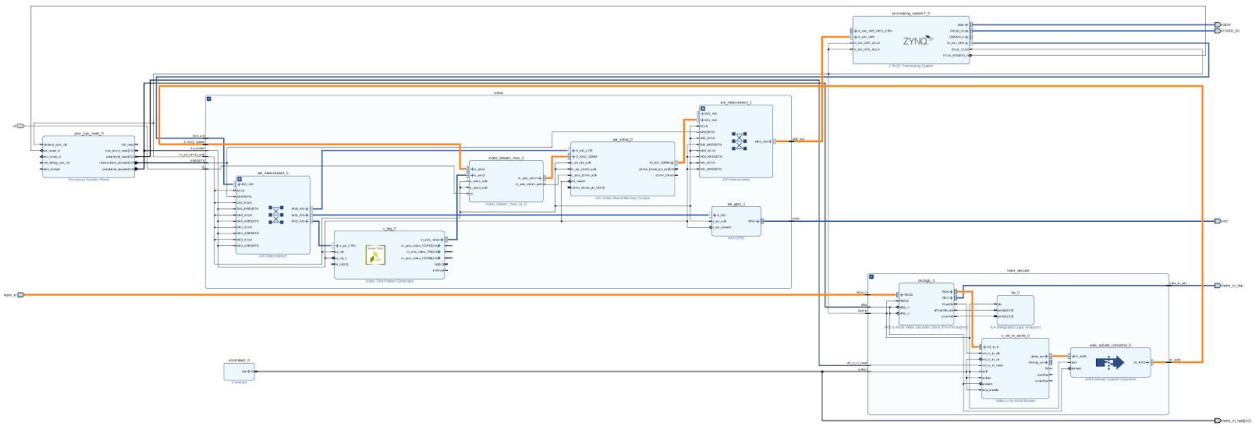


Figure 16: Detailed view. Highlighted line is the path of the video stream.

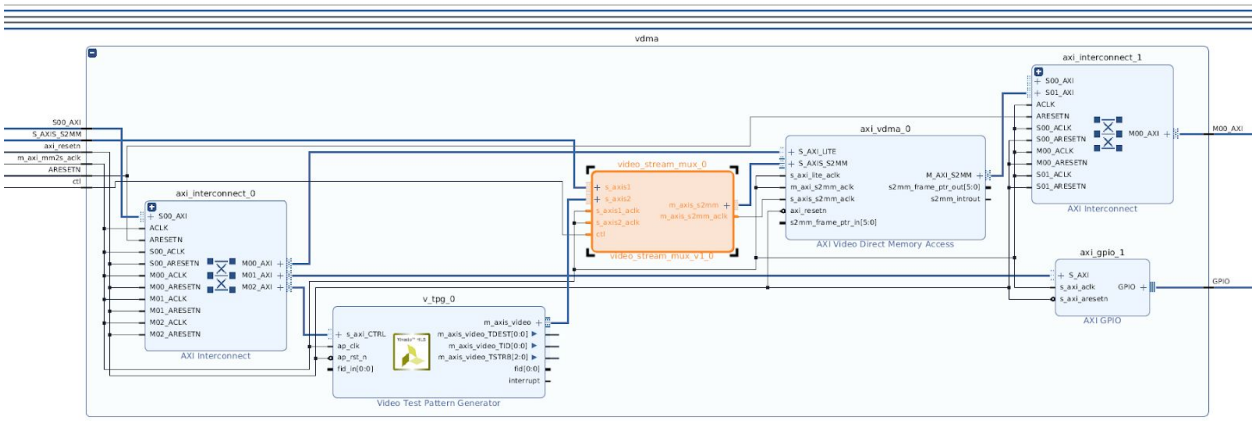


Figure 17: Highlighted IP core is VHDL that was custom made with vivado's IP core editor. I needed a way to stream Xilinx's VTPG and the HDMI input into a single S2MM (Slave to Memory Mapped) port, so I created a video stream mux.

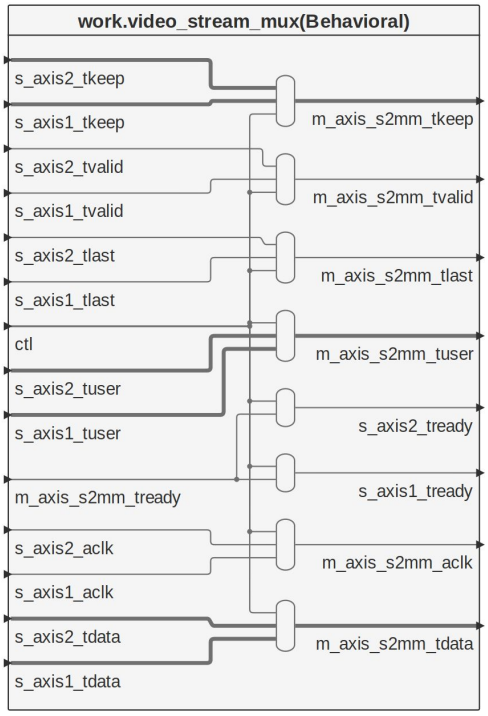


Figure 18: Internals of video stream mux.

Summarization of Build

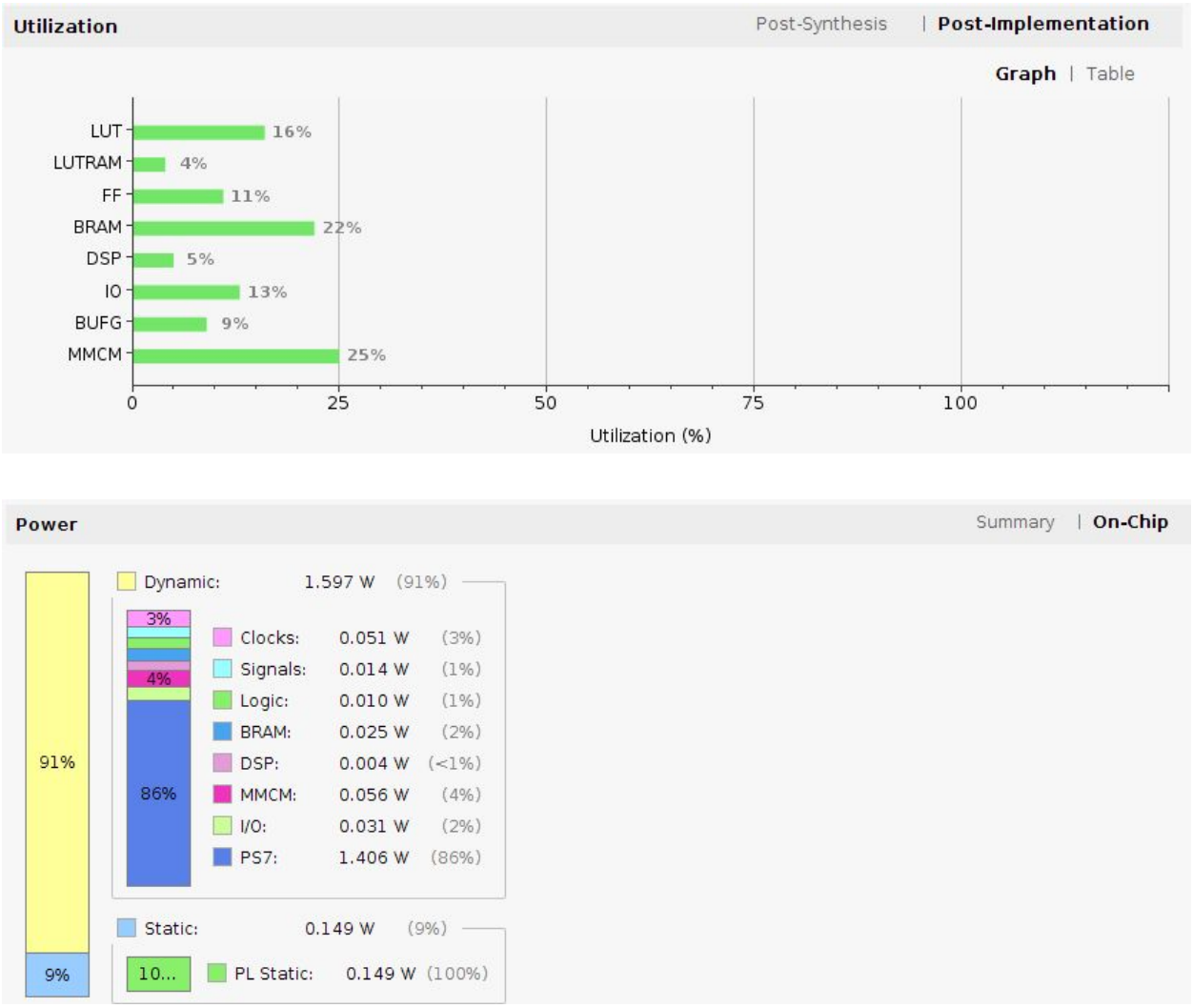


Figure 19: Resource and power utilization.

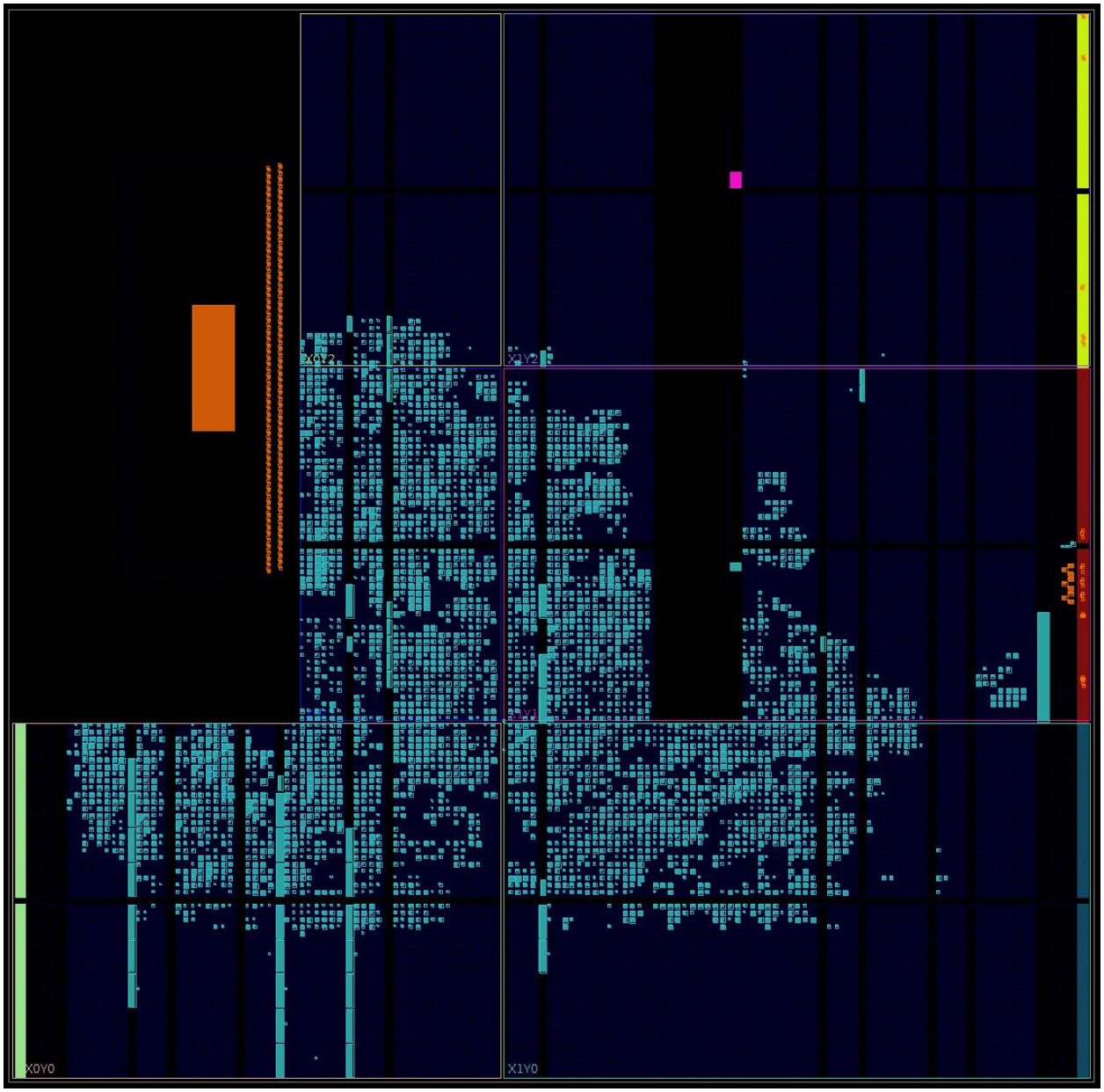


Figure 20: Implemented design on FPGA fabric. Highlighted areas indicate utilization.