International Institute of Information Technology
Bangalore

VLS 505
System Design with FPGA Project Report

Team member : Achintya Harsha (IMT2021525)

Team Member : Akash Perla (IMT2021530)

November 8, 2024

Contents

1 Project Overview

The primary goal of this project was to interface the OV7670 camera with FPGA and displaying the image on the VGA monitor.

2 Project Implementation

The Project was implemented using Digilent Basys3 Xilinx Artix-7 FPGA Board, OmniVision OV7670 Camera Module and using a monitor having a resolution of 1920 X 1080.

2.1 Video Graphics Array (VGA)

VGA, or Video Graphics Array, is an analog video standard developed by IBM in 1987. The purpose of VGA (Video Graphics Array) was to provide a standardized way to display graphics on computer monitors, allowing for a higher resolution and better color depth than previous video standards.

A VGA monitor can be visualized as a grid of pixels, where each pixel is a picture element that can be assigned a specific color. The VGA interface operates serially, meaning the color information for each pixel is transmitted sequentially, rather than simultaneously. Colors are defined by a triplet representing the intensity levels of the three primary colors: red, green, and blue.[1]

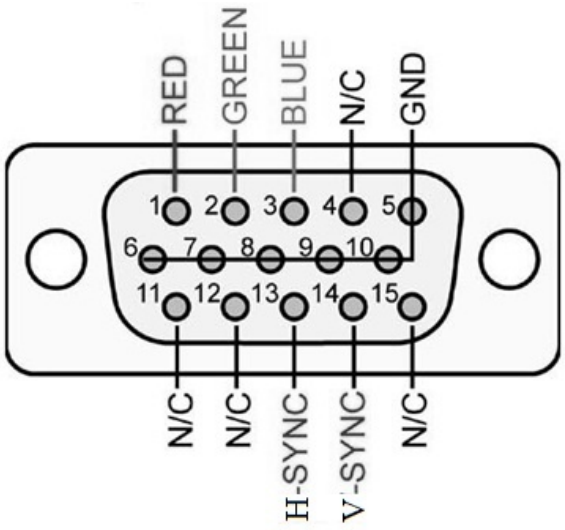


Figure 1: VGA Connector Pin Configuration. [2]

Figure 1 shows the overall VGA Connector Pin Configuration. We would need to provide the signals H-SYNC and V-SYNC to the VGA controller.

Horizontal Sync (HSYNC): This pin provides a pulse that indicates the start of a new row of pixels. It controls the horizontal timing and is used to synchronize the display to ensure each row starts at the correct time.

Vertical Sync (VSYNC): This pin provides a pulse to indicate the start of a new frame (from top to bottom). It controls the vertical timing, synchronizing the start of each new frame.

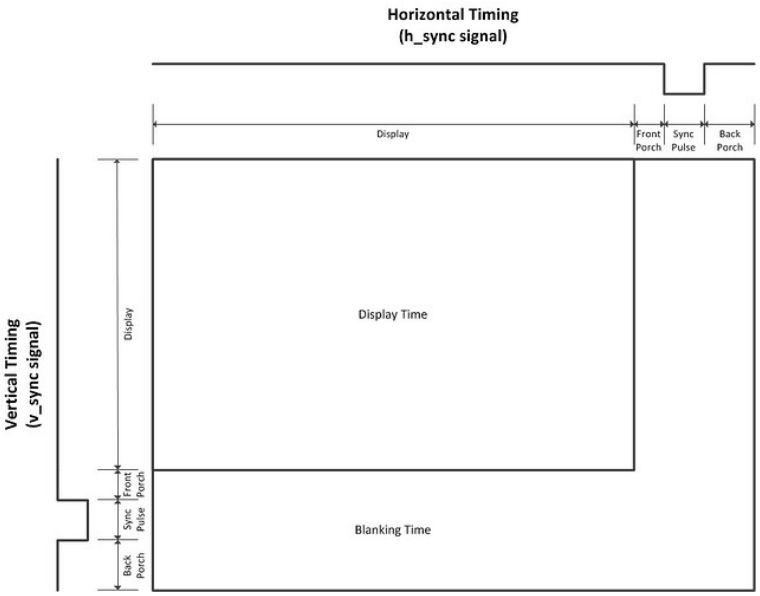


Figure 2: VGA Signal Timing Diagram. [3]

Parameter	Horizontal (hsync)	Vertical (vsync)
Display	1920	1080
Front Porch	88	4
Sync Pulse	44	5
Back Porch	148	36

Table 1: hsync and vsync signal parameters

This layout provides a clear and organized view of the hsync and vsync values. The controller has two counters to manage synchronization signals. One counter increments with each pixel clock and controls horizontal sync (h_sync). It starts display time at counter value 0, so the counter tracks the pixel's column position. After display, a blanking period occurs, including a front porch, sync pulse, and back porch. At the end of each row, the counter resets.

The second counter increments after each row, handling vertical sync (v_sync). It also starts display time at 0, representing the pixel's row position. Following display, there's a vertical blanking period with its own front porch, sync pulse, and back porch. When the vertical blanking ends, the counter resets to start a new screen refresh. [4]

We have been using a display resolution of 1920 x 1080 pixels to achieve a refresh rate of 60 frames per second.

$$\begin{aligned} 1 \text{ Frame} &= \big((h_{\text{display}} + h_{\text{front\_porch}} + h_{\text{sync\_pulse}} + h_{\text{back\_porch}}) \\ &\quad \times (v_{\text{display}} + v_{\text{front\_porch}} + v_{\text{sync\_pulse}} + v_{\text{back\_porch}})\big) \\ &= \big((1920 + 88 + 44 + 148) \times (1080 + 4 + 5 + 36)\big) \text{ pixels at 60 Hz} \\ &= 2475000 \text{ pixels at 60 Hz} \end{aligned}$$

Therefore, 1 pixel = $2475000 \times 60 = 148.5$ MHz, which is generated using the Clocking Wizard IP.

We have used the RGB 444 color format in our program as the Basys 3 board uses 14 FPGA signals to create a VGA port with 4-bits per color[5]. RGB 444 refers to a color format where the color components—Red, Green, and Blue—are represented using 4 bits each. This means that each of the three primary color channels

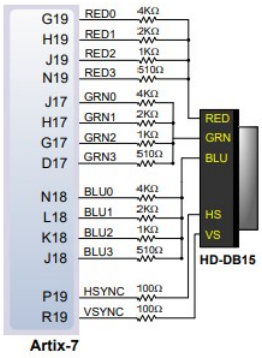


Figure 3: VGA Connections [5]

can have one of 16 possible intensity levels, since $2^4 = 16$. The total number of colors that can be represented in this format is: $16 \times 16 \times 16 = 4,096$. Thus, RGB 444 can represent a total of 4,096 colors.

We have tried testing our VGA module by using a static image of resolution 1920 x 1080 which was stored in laptop. The Basys3 Block RAM has a size of 1,800 Kbits[5] and does not have the required space to store the full 1920 x 1080 image.

To optimize image storage in Block RAM, we resized the image, resulting in some resolution loss. We selected an image size of 384 x 216, which is derived from scaling down the original dimensions (1920 x 1080) by a factor of 5. Despite this compression, there is still sufficient space in Block RAM. We used the RGB 444 format, which provides 12 bits per pixel. This results in a total storage requirement of 384 * 216 * 12 = 995,328 bits, representing a 50-fold reduction in size compared to the original image.

To achieve this, COE files for the Block RAM were generated using a Python script that downsizes the original image and reduces its color depth from 8-bit to 4-bit per channel, producing the final 384 x 216 image at a 12-bit color depth, ready for storage in Block RAM.

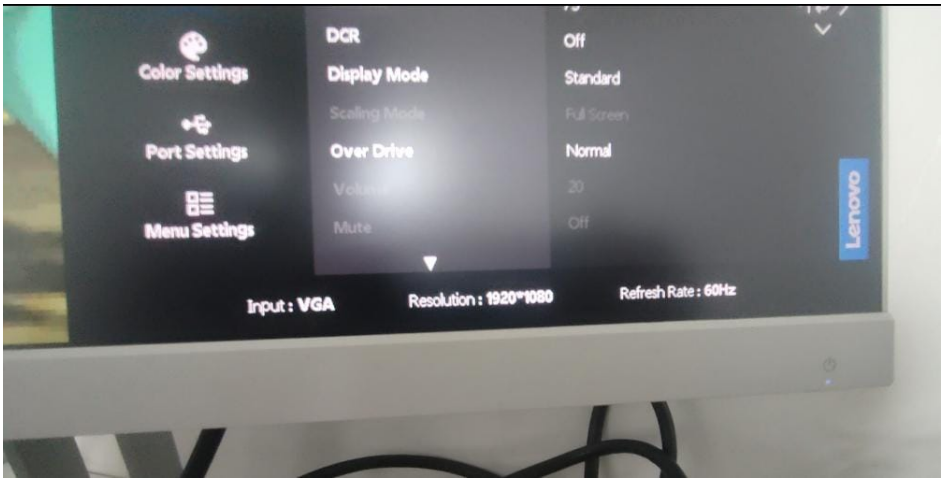
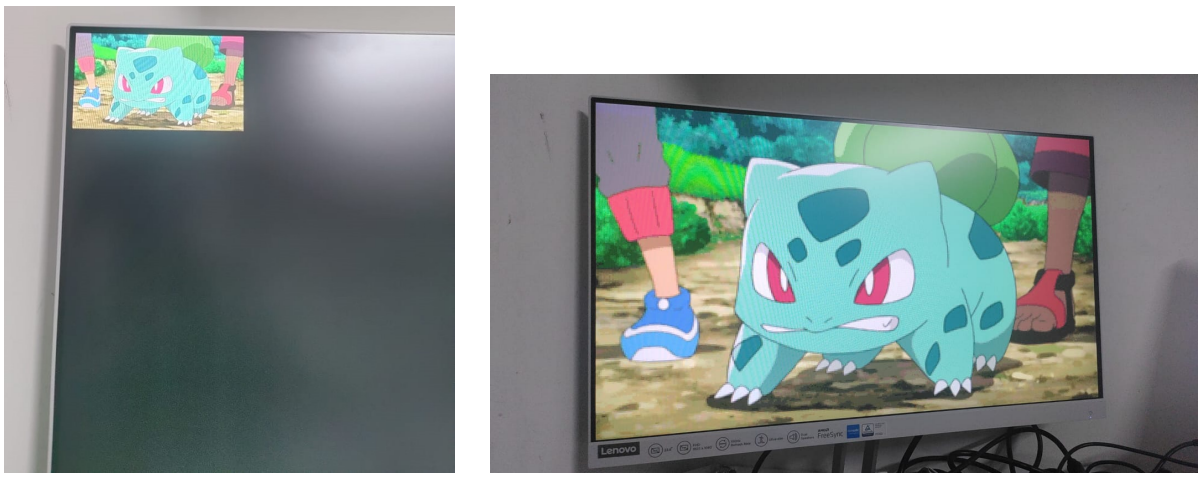


Figure 4: VGA configuration information shown on the monitor

2.2 Serial Camera Control Bus (SCCB) protocol

SCCB, or Serial Camera Control Bus, is a two-wire communication protocol used primarily to control camera modules, like the ones used in mobile devices. SCCB protocol is similar to I2C protocol. We have to use the two wire SCCB protocol to configure the OV7670 camera.



(a) Static image from laptop to VGA (b) Upsampled static image from laptop to VGA

Figure 5: Comparison of VGA display images

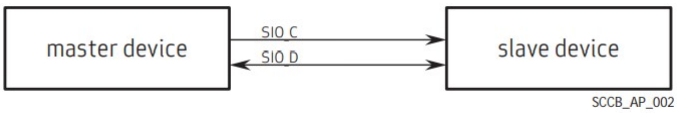


Figure 6: SCCB Block Diagram [6]

The SCCB uses two lines for communication:

- **SIO_C:** Carries the clock line
- **SIO_D:** Transfers the data between master and slave devices

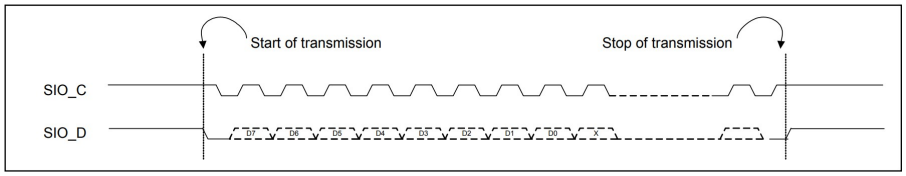


Figure 7: SCCB Transmission Timing Diagram [6]

The start of data transmission will occur when the **SIO_D** is driven to "0" and **SIO_C** is "1". A write or read operation will always be initiated by the master and only after the occurrence of the start condition.

A stop of data transmission will occur when the **SIO_D** is driven low or "0" to high or "1" while the **SIO_C** signal is high or "1".

Phase 1: Slave Address, Write Bit, Don't Care Bit

Phase 2: Camera Register Address, Don't Care Bit

Phase 3: Data to Register, Don't Care Bit

The Slave Address is this case is the OV7670 camera Address(i.e 7'h21)

2.3 OV7670 Camera

The OV7670 is a popular CMOS camera sensor module developed by OmniVision, which is used for capturing images and video. The OV7670 camera module has an Image Sensor Array of active pixels 640 x 480.

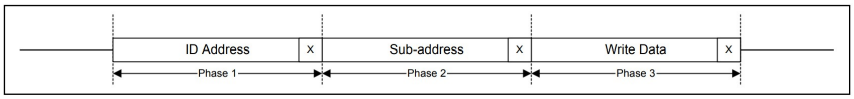


Figure 8: SCCB 3-Phase Write Transmission Cycle [6]

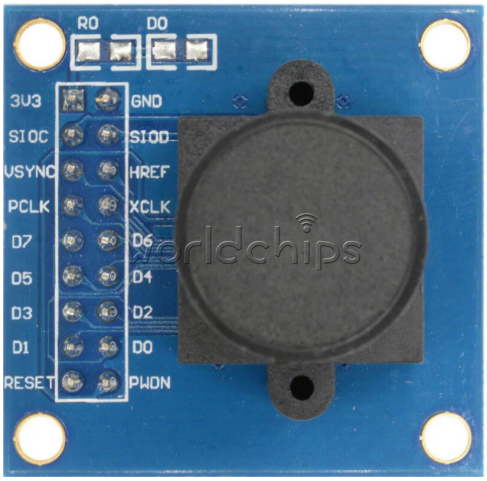


Figure 9: OV7670 Camera Module Pinout

When powered up, the OV7670 sensor is configured through SCCB by writing to its internal registers. The sensor captures light through its lens, converts it to an electrical signal, and then processes it into digital image data. This data can be formatted into RGB 444.

Pin Description: [7]

- The pixel data is sent out over an 8-bit parallel data bus (**D0-D7**).
- The pins **SIOC** and **SIOD** are used for SCCB communication between the camera module and Basys 3 board.
- The **XCLK** is an input clock signal. We have provided a clock with a frequency of 24 MHz to the camera module using XCLK.
- **PCLK**, is the output signal used to latch the 8-bit output data bus. With each rising edge of PCLK, 8 bits of pixel data are made available for reading.
- **HREF** is an output signal that stays high during the transmission of data for a row, indicating active pixel data.
- **VSYNC** is an output signal used to indicate the end of a frame, signaling the completion of one full image transmission.

3 Integration of FPGA with OV7670 Camera Module

3.1 Setup and Pin Mapping

The pmod pins provided by the BASYS-3 FPGA board was used to connect the camera module. Specifically the JB and JC set of ports were used to establish connection.

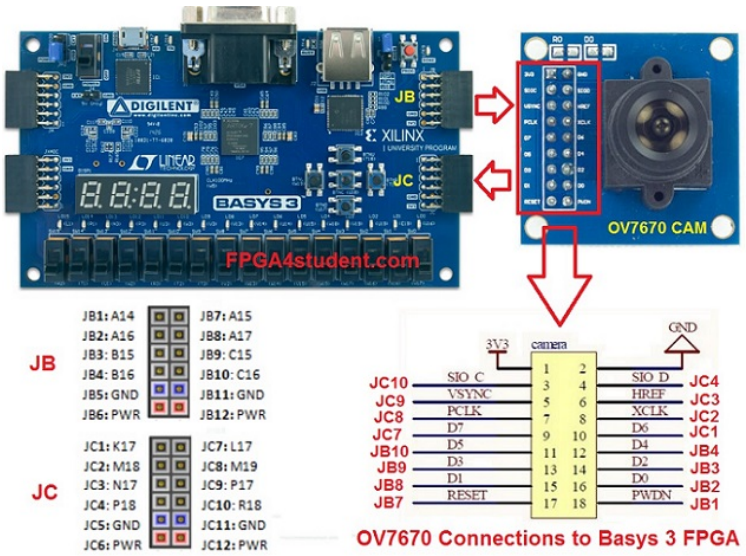


Figure 10: Camera connections [8]

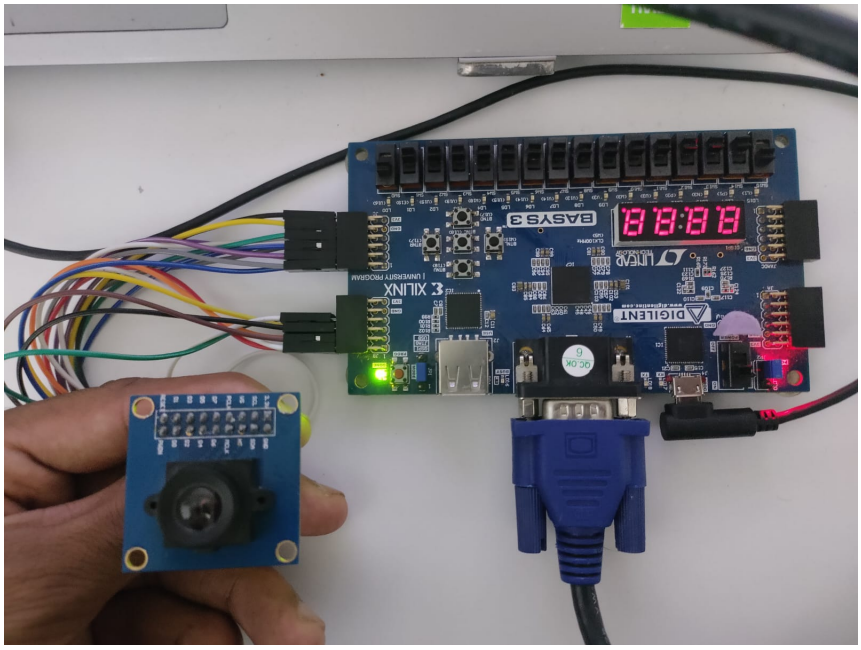


Figure 11: Image showing the camera module wired together with the BASYS-3 FPGA

Basys3: Pmod Pin-Out Diagram

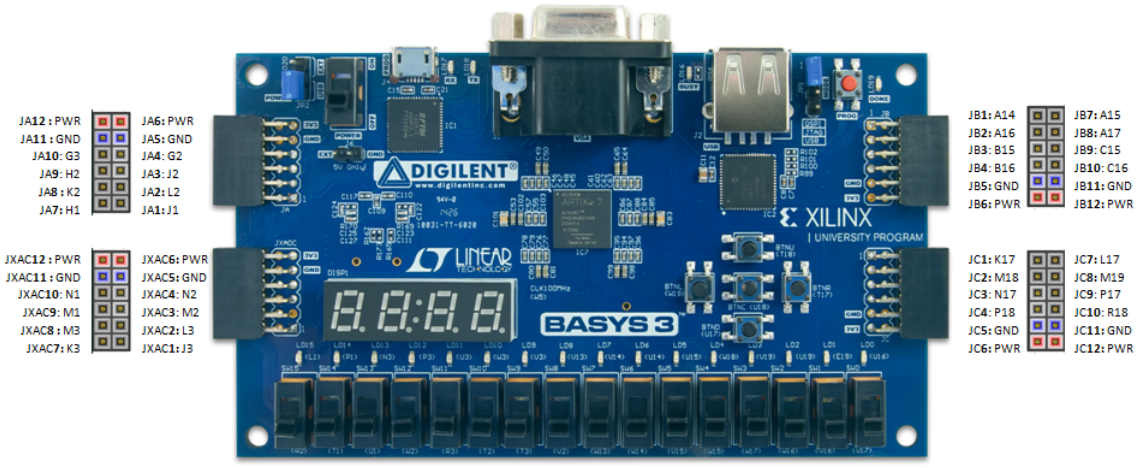


Figure 12: Pmod ports

Package Pin	Port	Package Pin	Port	Package Pin	Port
A16	D0	A15	Reset	A14	PWDN
A17	D1	M18	MCLK	M19	PCLK
B15	D2	N17	HREF	R18	SIOC
C15	D3	P17	VSYNC	P18	SIOD
B16	D4				
C16	D5				
K17	D6				
L17	D7				

Table 2: Pin to Port Mapping

3.2 IPs used

1. **Clocking Wizard:** As mentioned earlier, the VGA requires a 148.5 MHz clock to operate, as the display being used is 1920x1080 resolution. Additionally, the OV7670 camera requires a 24 MHz clock (MCLK) to operate. Both clocks are generated using the Clocking Wizard (MMCM).

2. **Block RAM:** It is important to note that we are crossing clock domains when using the Block RAM. Data is written to the Block RAM at a lower frequency than it is being read. Therefore, a simple two-ported Block RAM must be used. The memory specifications are as follows:

   - 384×216 address depth (corresponding to the maximum number of pixels)
   - 12-bit data width (corresponding to the depth of each pixel)
   - Simple two-ported memory with one read port and one write port

3.3 Configuration

We used the *OV7670 camera setup code* [9] to configure the camera module. We modified the code to fit our needs and combined it with our own VGA implementation.

4 Vivado Implementation Results

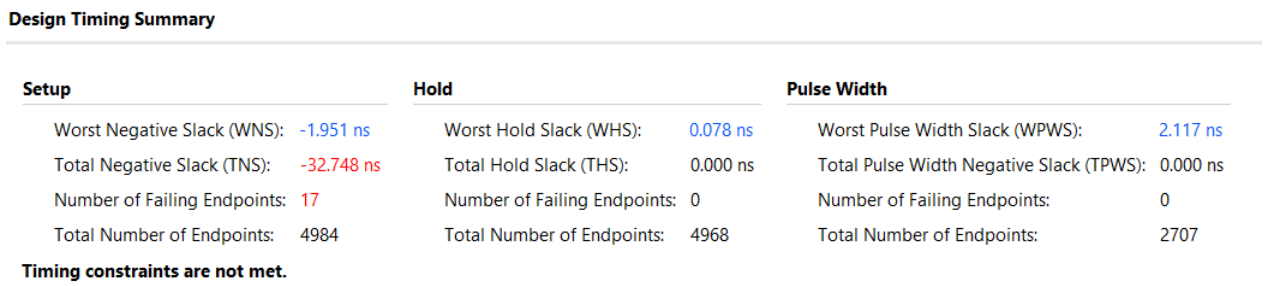


Figure 13: Timing summary

Hence, we can conclude that the maximum clock frequency achieved = **148 MHz**.

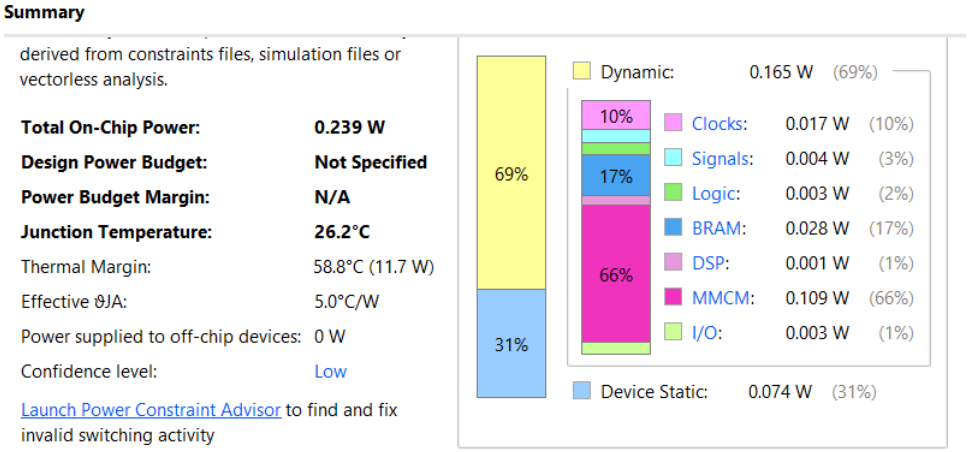


Figure 14: Power summary

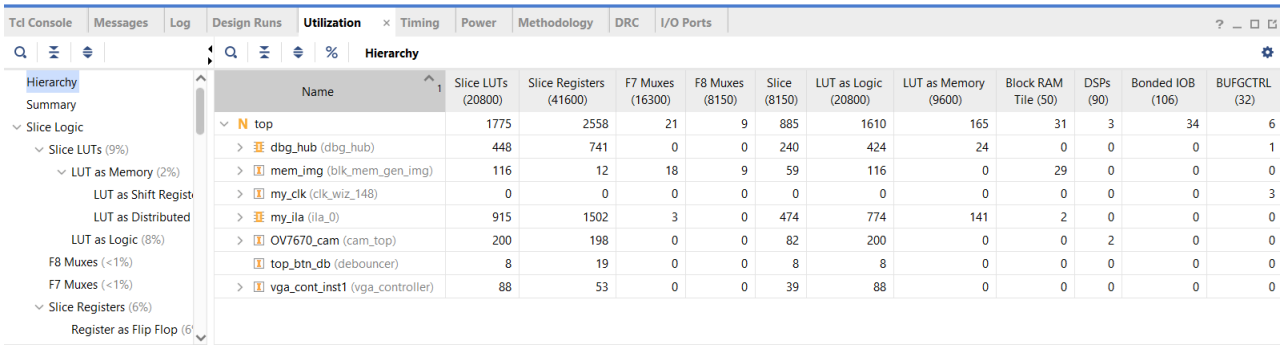


Figure 15: Resource Utilization

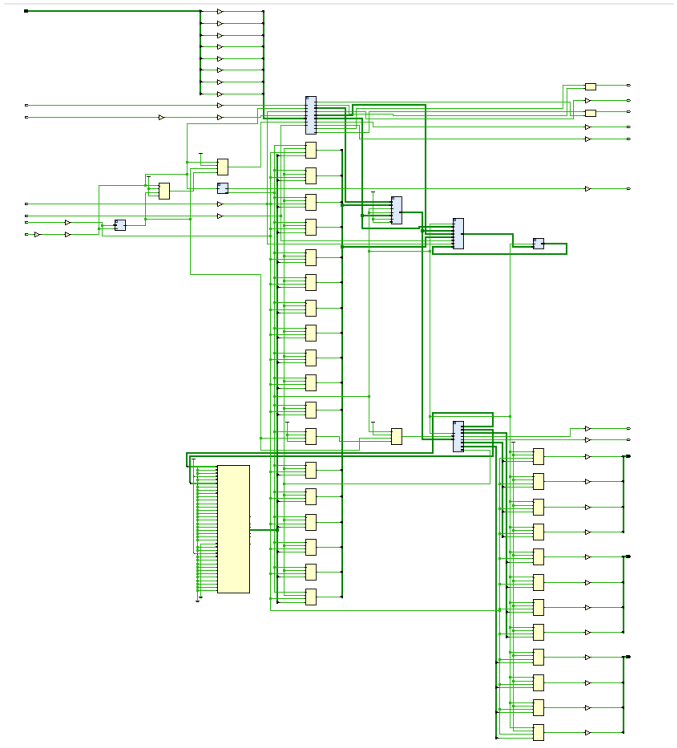


Figure 16: Schematic

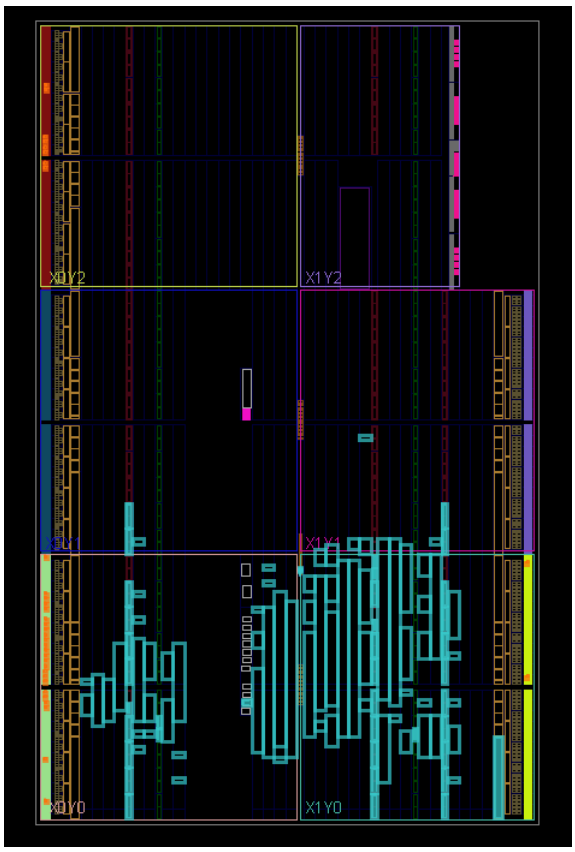


Figure 17: Layout

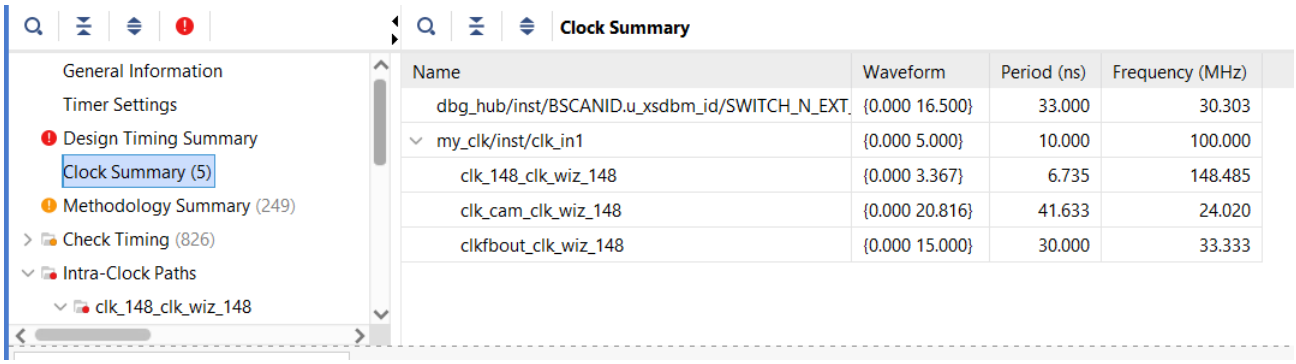


Figure 18: Clock Summary

Design Timing Summary

Setup		Hold		Pulse Width	
Worst Negative Slack (WNS):	-1.951 ns	Worst Hold Slack (WHS):	0.078 ns	Worst Pulse Width Slack (WPWS):	2.117 ns
Total Negative Slack (TNS):	-32.748 ns	Total Hold Slack (THS):	0.000 ns	Total Pulse Width Negative Slack (TPWS):	0.000 ns
Number of Failing Endpoints:	17	Number of Failing Endpoints:	0	Number of Failing Endpoints:	0
Total Number of Endpoints:	4984	Total Number of Endpoints:	4968	Total Number of Endpoints:	2707

Timing constraints are not met.

Figure 19: Timing summary

5 Final Output

The integration of the OV7670 camera with the FPGA is functional; however, the color scheme is not accurately represented. The camera struggles to detect dark objects, capturing only bright light sources effectively. This issue likely stems from improper configuration of the camera's registers, which number 75 in total. Unfortunately, the documentation for these registers lacks detail, making it challenging to accurately configure the camera settings for optimal performance.

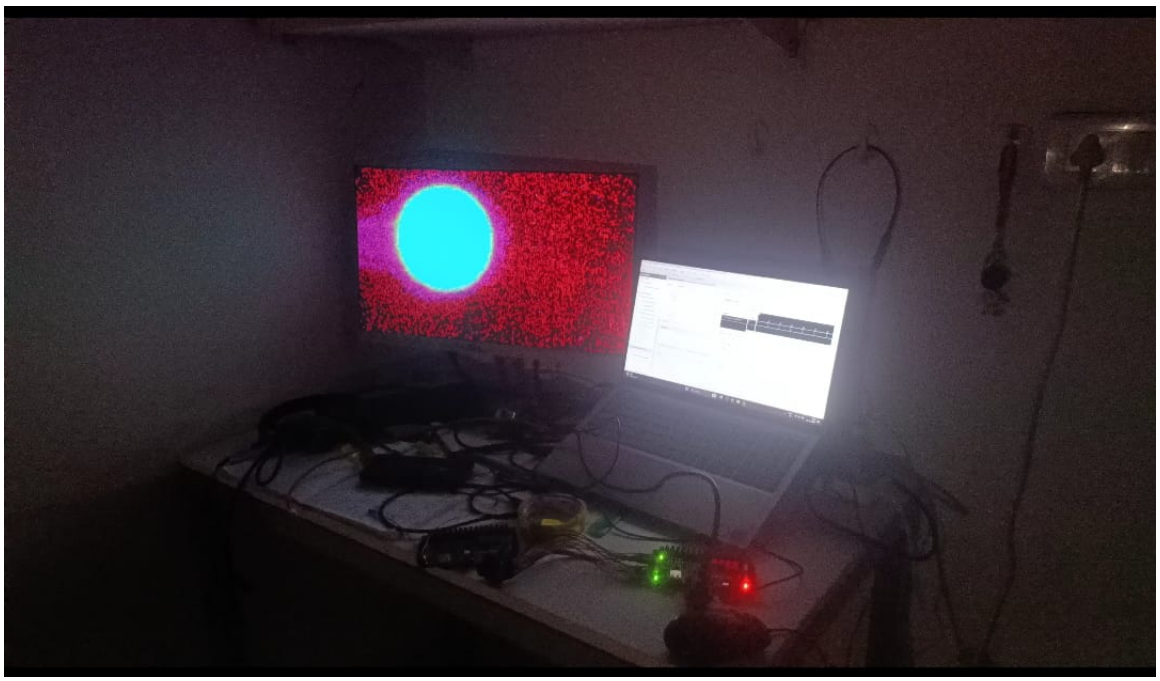


Figure 20: Light Source Visible prominently

To validate the functionality of the camera, a mobile flashlight was used to track the light. The demonstration video can be accessed **here**.

6 GitHub Link

The git hub link for the codes can be found **here**.

References

U. of Toronto, "Vga monitors," 2006, accessed: 2024-11-08. [Online]. Available: https://www.eecg.utoronto.ca/~jayar/ece241_06F/vga/vga-monitors.html

Elprocus, "Vga connector: Working, types and its applications," 2024, accessed: 2024-11-08. [Online]. Available: https://www.elprocus.com/vga-connector/

D.-K. Forum, "Vga controller vhdl," 2024, accessed: 2024-11-08. [Online]. Available: https://forum.digikey.com/t/vga-controller-vhdl/12794

YouTube, "Vga signal timing and controller," 2024, accessed: 2024-11-08. [Online]. Available: https://www.youtube.com/watch?v=l7rce6IQDWs

D. Inc., “Basys 3 reference manual,” 2024, accessed: 2024-11-08. [Online]. Available: https://digilent.com/reference/_media/reference/programmable-logic/basys-3/basys3_rm.pdf

C. University, “Sccb specification and application note,” 2021, accessed: 2024-11-08. [Online]. Available: https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/f2021/jfw225_aei23_dsb298/jfw225_aei23_dsb298/SCCBSpec_AN.pdf

MIT, “Ov7670 camera module datasheet,” 2006, accessed: 2024-11-08. [Online]. Available: https://web.mit.edu/6.111/www/f2016/tools/OV7670_2006.pdf

FPGA4Student, “Basys 3 fpga with ov7670 camera,” 2018, accessed: 2024-11-08. [Online]. Available: https://www.fpga4student.com/2018/08/basys-3-fpga-ov7670-camera.html

A. Sacks, “Ov7670-camera,” 2021, gitHub repository. [Online]. Available: https://github.com/amsacks/OV7670-camera