

# **VL505 System Design with FPGA**

## **Course Project**

Lakshmi Sai Niharika Vulchi MT2023513

Valipireddy Pranathi MT2023510

**GitHub link:** [https://github.com/V-Pranathi/FPGA\\_PROJECT](https://github.com/V-Pranathi/FPGA_PROJECT)

**Guidance:** Prof. Nanditha Rao

### **Project statement:**

Read in the image from a laptop-camera, store it in block RAM, display it on VGA.  
Transform the image and display it in real-time on the VGA monitor.

### **Project hierarchy:**

1. Implementation of VGA controller with FPGA
2. Interfacing the UART for transferring the image to the FPGA
3. Implementing a real time parallel interface for the image taken from laptop camera to be displayed on the screen using VGA controller
4. Transforming the image, applying image blurring using image convolution, and displaying it on FPGA

## **Section 1: Project Description and Implementation**

### **VGA controller**

VGA, which stands for Video Graphics Array, is a standard interface used for connecting computers to displays, such as monitors and projectors.

We implemented VGA for 1280x1024 display with 60Hz frequency. The standard horizontal and vertical porch values for this specific display resolution are as follows:

- a. Horizontal front porch, retrace, back porch = 48, 112, 248 respectively
- b. Vertical front porch, retrace, back porch = 1, 3, 38 respectively

Horizontal sync and vertical sync signals are the control signals which are given to the VGA controller. These timing signals are asserted after the completion of horizontal **row** of pixels and the entire **frame** respectively.

RGB pixel data each of 4 bit is sent to the VGA interface for the display.

**VGA clock frequency:** We are using a display resolution of 1280x1024, to match the display frequency of 60 frames per second, each pixel in the image should be transmitted at a frequency of **108 MHz**.

1 frame =  $((1280+48+112+248) \times (1024 + 1 + 3 + 38))$  pixels: 60 Hz

1 pixel: 107.964 MHz  $\rightarrow$  108 MHz

We are displaying the image in 320x240 dimension and 12 bit pixel data on the 1280x1024 screen, thus the remaining screen except that of the image is made black.

### **Storing the image on FPGA**

The image pixels each of length 12 bits, where each RGB component is of 4 bits is stored in a block RAM. Basys 3 board can be used to store only one image of 320x240 size where each pixel is 12 bit width or two images of 320x240 size where each pixel is 11 bit width. A coe file is created using python which converts the image pixel data to 12 bit.

### **UART interface**

A parallel UART interface is made which transfers the image taken from the laptop camera to FPGA. Python file reads the data from the image and writes it to the serial port which is connected to the laptop. This data is then sent it to the FPGA using USB, on the Basys 3 board FTDI FT2232 chip converts the USB signal to UART and this is sensed by FPGA using the specific pins.

A **two state FSM** is designed to handle the operation of the module. FSM states:

- LOW\_BYTE: for the lower 8 bits of data – 6 bits of pixel data and 2 safety bits
- HIGH\_BYTE: for the higher 8 bits of data

The **clock frequency** used is 50 MHz and **baud rate** of 1834200 which enables very fast transmission of image from laptop to the board.

Baud rate is chosen to be such that we can transfer 2 images of 320x240 of 11bit data width for pixel.

**Oversampling** of data is done to reduce the interference of noise and make sure that the right data is transferred over UART, here the oversampling factor is 15, which can be chosen as your wish

$\text{UART\_frequency} = \text{oversampling\_factor} * \text{baud rate} * \text{clock\_divider\_factor}$

### Image transformation (blurring)

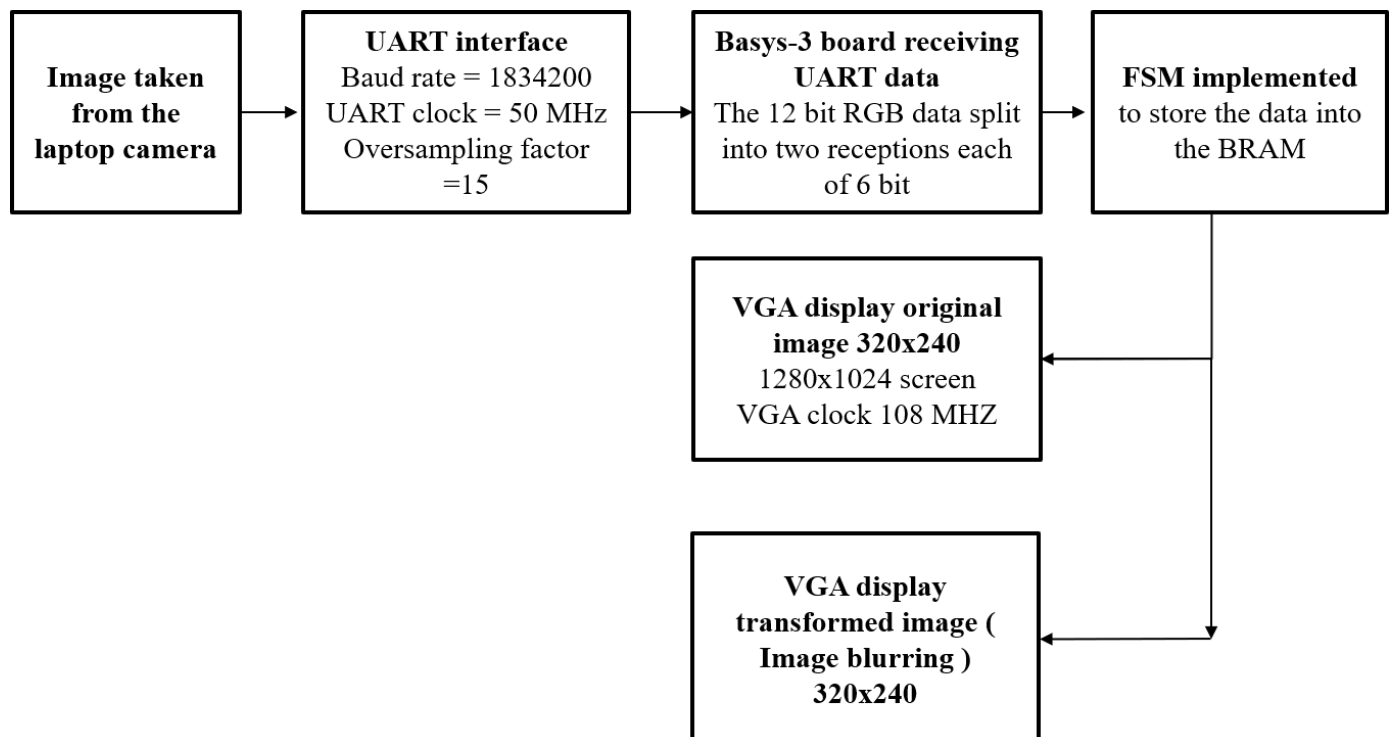
We are applying image transformation on the received image from UART, technique used is image blurring.

The image which is stored in the first block ram is read and then the convolution operation is operated on each pixel and its surrounding pixel by the weights of the kernel and then the data is stored in the second block ram.

The kernel we used is a 3x3 kernel with all the values as 1/9

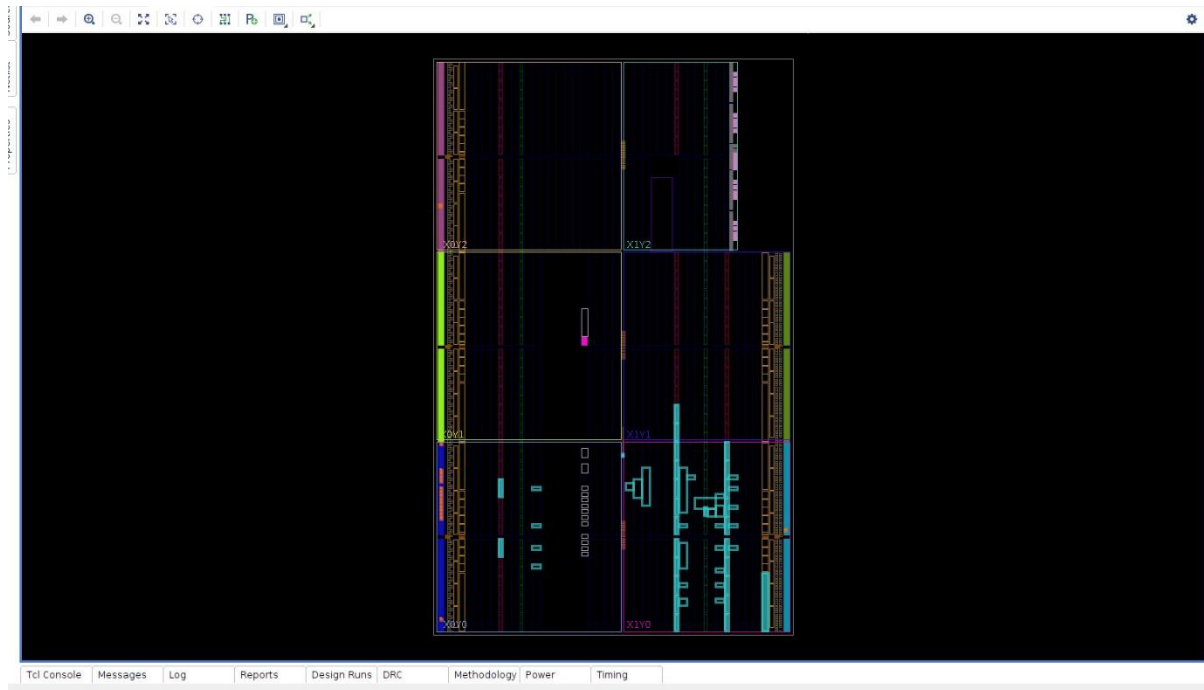
The data which is being written into the second block ram is read and sent to the VGA for the image display on the screen.

Below figure is the block diagram detailing our project work.

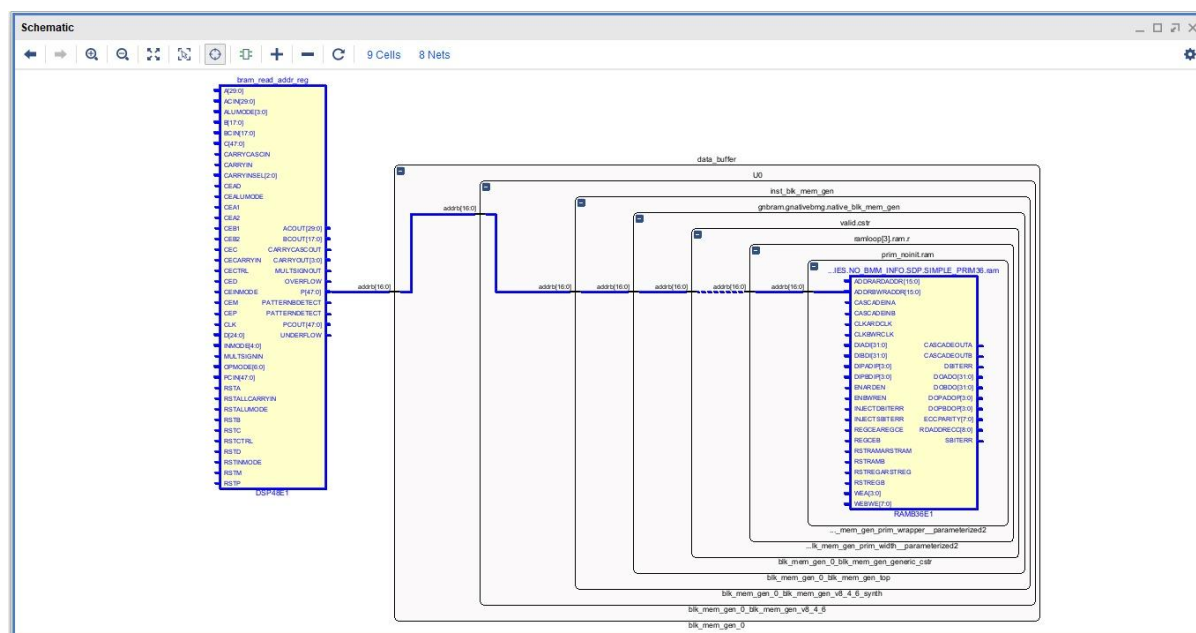


## Section 2: Vivado Implementation results

### Post implementation floorplan:



### Post implementation schematic with critical path highlighted



## Post implementation critical path analysis with source and destination

**Path 2 - controller\_timing\_summary\_routed**

Property	Value
Name	Path 2
Slack	2.893ns
Source	bram_read_addr_reg/CLK (rising edge-triggered cell DSP48E1 clocked by clk_vga_clk_wiz_0 (rise@0.000ns fall@4.630ns period=9.259ns))
Destination	data_buffer/U0Inst/blk_mem_gen/gnram.gnativebmg.native_blk_mem_gen/valid.cstr/ramloop[3].ram.r/prim_noinitram/DEVICE_7SERIES.NO_BMM_INFO.SDP
Path Group	clk_vga_clk_wiz_0
Path Type	Setup (Max at Slow Process Corner)
Requirement	9.259ns (clk_vga_clk_wiz_0 rise@9.259ns - clk_vga_clk_wiz_0 rise@0.000ns)
Data Path Delay	5.579ns (logic 0.434ns (7.779%) route 5.145ns (92.221%))
Logic Levels	0
Clock Path Skew	-0.148ns
Clock Uncertainty	0.073ns

**Intra-Clock Paths - clk\_vga\_clk\_wiz\_0 - Setup**

Name	Slack	Levels	High Fanout	From	To	Total Delay	Logic Delay
Path 2	2.893	0	26	bram_read_addr_reg/CLK	data_buffer/_DRBWRADDR[8]	5.579	0.434
Path 3	3.075	1	28	bram_read_addr_reg/CLK	data_buffer/U...ram/ENBWREN	5.514	0.558
Path 4	3.118	0	26	bram_read_addr_reg/CLK	data_buffer/U...RBWRADDR[12]	5.354	0.434
Path 5	3.188	0	26	bram_read_addr_reg/CLK	data_buffer/_DRBWRADDR[3]	5.282	0.434
Path 6	3.202	0	26	bram_read_addr_reg/CLK	data_buffer/_DRBWRADDR[5]	5.270	0.434

## Post implementation timing report:

**Design Timing Summary**

Category	Constraint	Value
Setup	Worst Negative Slack (WNS)	2.852 ns
	Total Negative Slack (TNS)	0.000 ns
	Number of Failing Endpoints	0
	Total Number of Endpoints	494
Hold	Worst Hold Slack (WHS)	0.153 ns
	Total Hold Slack (THS)	0.000 ns
	Number of Failing Endpoints	0
	Total Number of Endpoints	494
Pulse Width	Worst Pulse Width Slack (WPWS)	3.000 ns
	Total Pulse Width Negative Slack (TPWS)	0.000 ns
	Number of Failing Endpoints	0
	Total Number of Endpoints	81

**All user specified timing constraints are met.**

## Post implementation utilization report:

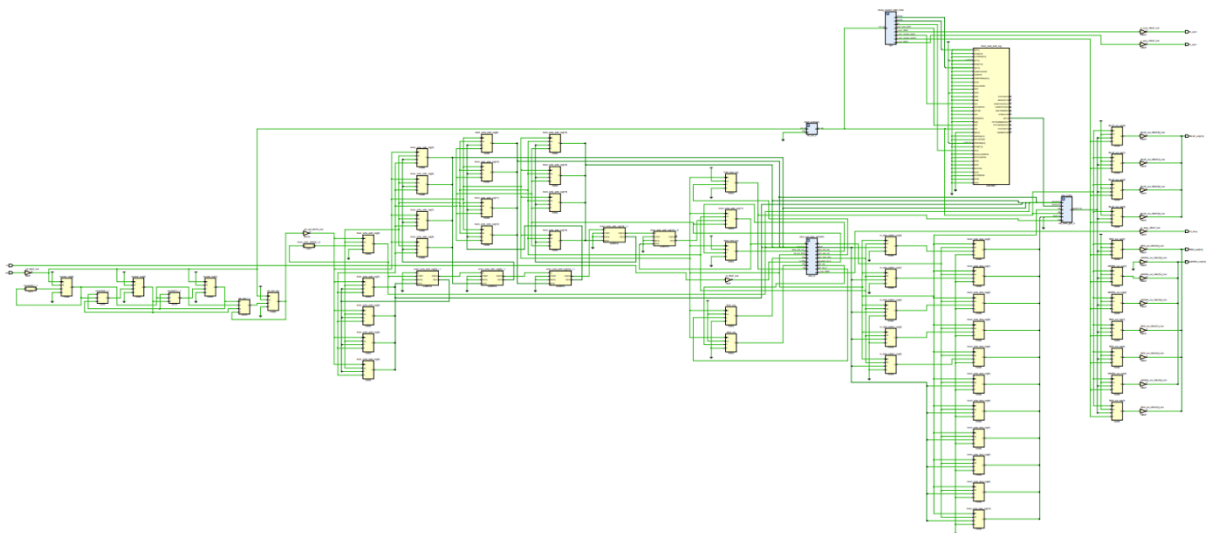
**Utilization**

Component	Value
Slice LUTs (20800)	202
Slice Registers (41600)	117
F7 Muxes (16300)	27
F8 Muxes (8150)	9
Slice LUTs (8150)	90
LUT as Logic (20800)	202
Block RAM Tile (50)	25
DSPs (90)	1
Bonded IOB (106)	17
BUFCTRL (32)	3
MMCM2_ADV (5)	1

### Timing analysis:

- Worst slack: 2.852 ns for 100MHz clock frequency
- Maximum operating frequency obtained: 139.9 MHz

### Post implementation schematic:



### **Section 3: VGA display**

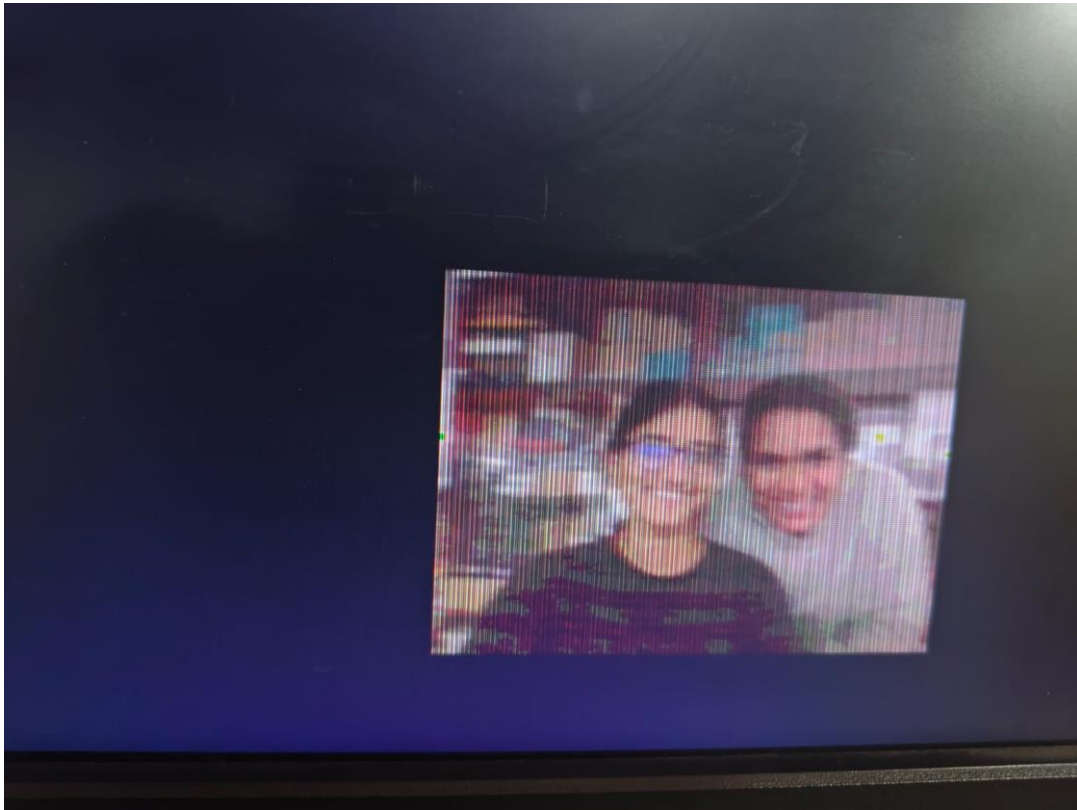
**Real time image capture from laptop camera to FPGA to display on monitor through VGA**



**Video link :**

[https://github.com/V-Pranathi/FPGA\\_PROJECT/blob/main/fpga\\_demo.mp4](https://github.com/V-Pranathi/FPGA_PROJECT/blob/main/fpga_demo.mp4)

**Image display after blurring :**



#### **Section 4: References**

1. <https://github.com/Shubhayu-Das/VL504-project/>
2. <https://www.youtube.com/watch?v=MVu5OAFZhKA&t=375s>

#### **Acknowledgments**

Jay Shah, MS, IIIT- Bangalore