ECE 272 Lab #7

Displaying Images with VGA

Marshall Saltz

12/05/2022

1. Introduction

Lab 7 expands on Lab 6's goal of outputting color to a monitor and kicks it up to displaying a sprite, or image. The same concepts are at play with one addition: ROM, or Read Only Memory. ROM stores the sprite so that it can be projected onto the display. The materials used are: Quartus Prime 18, a monitor, a VGA cable, a USB to USB-B cable, my computer, and the DE10-Lite FPGA. The system will work to load the data as a MIF (memory initialization file) into memory and then displaying that data in an array of pixels.

2. Design

Figure 1: Sprite

This is the image I chose to create a MIF of. I followed the steps and used mifMaker to create it.

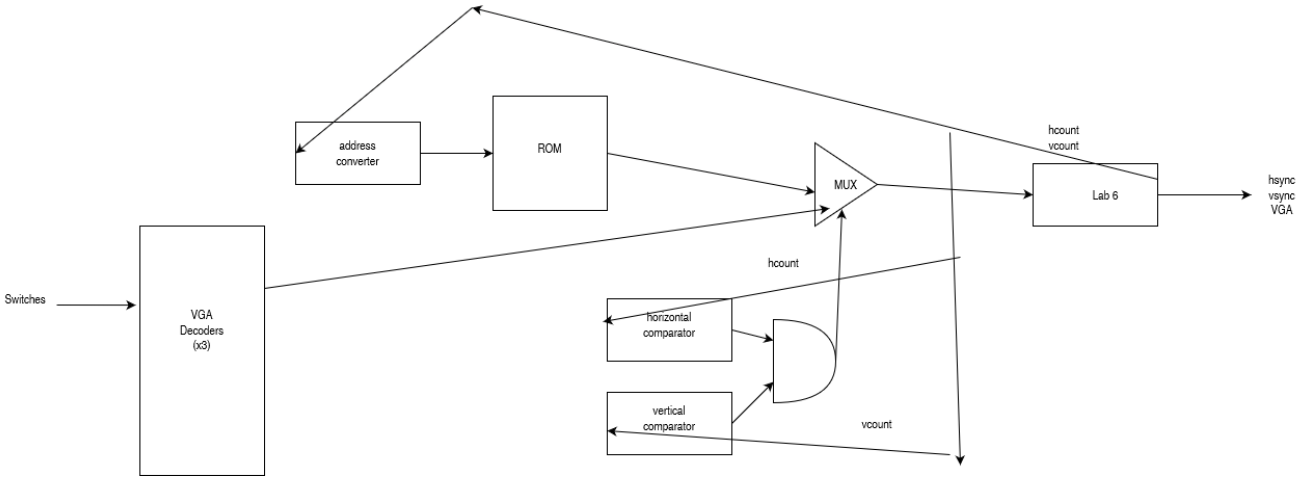

Figure 2: Block Diagram

This shows the combinational logic for the planned schematic. It sends a count of the horizontal and vertical clocks to the address converter, which communicates with the ROM, decoders, and comparators via a multiplexer to output the display.

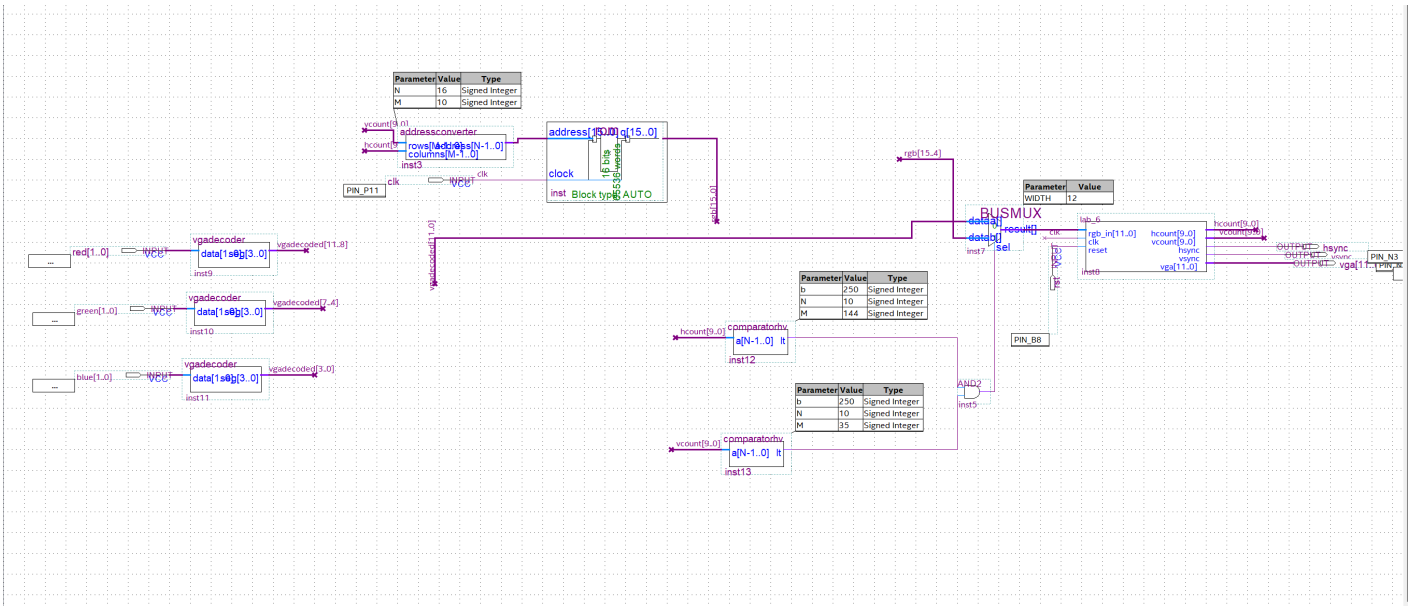

Figure 3: Top Level Schematic

This schematic incorporates lab 6, the decoders, and the comparators from lab 6 as well as a new address converter, ROM, and a multiplexer.

Interface Name	Interface Purpose	Input or Output	Number of Pins	Active High or Low	Board Label	FPGA Pins
Blue, Green, Red	Gate Inputs	Inputs	6	High	Slide Switch	B12 through C11
Clk	Gate Clock	Input	1	High	50 MHz Clock	P11
Rst	Gate Input	Input	1	Low	Push Button	B8
Hsync	Gate Output	Output	1	High	VGA	N3
Vsync	Gate Output	Output	12	High	VGA	N1
vga	Gate Output	Output	12	High	VGA	AA1 through N2

Figure 4: Interface Definition
This figure details which pins were aligned with which inputs and outputs.

3. Results

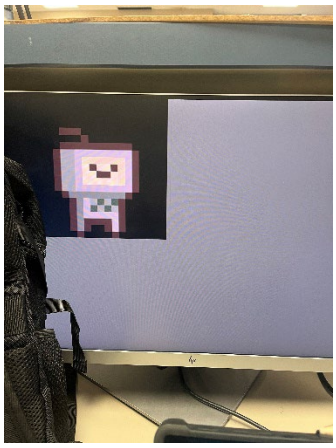

Figure 5: Results Photo 1
This photo shows that in addition to displaying a sprite, the output also displays a background color.

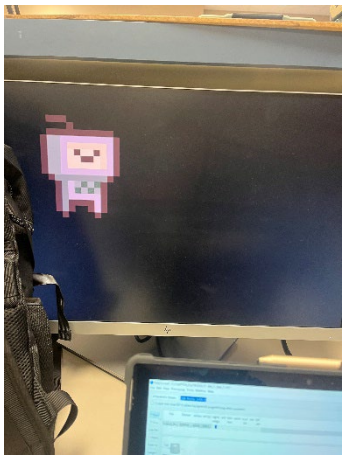

Figure 6: Results Photo 2
The output without color.

After tinkering with lab 6 and the top level diagram, the display clearly shows the sprite after the code and schematics were pushed to the board and a VGA cable was connected.

4. Experiment Notes

This has been the only lab that did not take an extraordinary amount of time. I felt like I understood the contents from Lab 6 going in, and with the instructional video, creating the MIF and ROM was easy. At first, the sprite was diced up due to an incorrect pixel count, but that was easily fixed.

5. Study Questions

1. The most difficult part of ECE 272 was understanding what the labs were asking and how to use the Quartus software. Maybe more clarification or photos of what the output should look like would be helpful.

2. I would like to try to physically break the FPGA by overclocking it. I looked into building a clock multiplier with a PLL, so I think that will be a fun project going forward—trying to break the FPGA with code and push it past its limits.

3. I disliked the seven segment decoder the most. It wasn't due to a lack of understanding, but rather the unnecessarily complicated and difficult to troubleshoot schematic that I had to create.

4. I liked section 7 the most because I felt at that point I understood everything to the point of where I needed minimal help. I wish I had time to try for the extra credit opportunity.

6. Appendix

```
module addressconverter#(parameter N = 16, M = 10)(input logic [M-1:0] rows, columns, output logic [N-1:0]address);

      assign address = ((rows-35) * 250) + (columns-144);


endmodule
```

```
module comparatorhv #(parameter b = 256, N = 10, M = 35)(input logic [N-1:0] a, output logic lt);


      assign lt = ((a-M) < b);

endmodule
```