ECE 272 Lab 6

Video Graphics Array

Emily Becher

June 1, 2022

Jacob Field

1. Introduction

The goal of this lab was to create a VGA display using a DE10-Lite FPGA. VGA or Video Graphics Array displays hexadecimal values to a monitor by changing each pixel starting in the top left. It goes across the screen to the far right before moving down. After completing the screen it loops back to the top and starts again. This lab displayed a solid color onto the screen. The extra credit challenge was to have the color fade down the screen. The lab included eight inputs and fourteen outputs. The inputs were an active LOW button for an asynchronous reset, a clock signal provided by the board, and six active HIGH switches. For the extra credit the number of switches was reduced to three for a total of five inputs. The outputs were sent to the VGA. Twelve of the outputs made up the hexadecimal RGB information. The remaining two outputs were to synchronize the output with the FPGA. The project could display 64 different colors and the extra credit portion could display fades for 7 different colors. Unlike many of the other labs, this lab didn't build off a previous lab. It did use some of the same System Verilog blocks as lab 5.

2. Design

The logic in this lab can be split into two main parts. One part controls the timing and determines when data should be displayed. The second part takes in user inputs and converts the inputs into a form that can be read by VGA. The timing of the lab has to match timing specifications dictated by VGA protocol and shown in Figure 1.

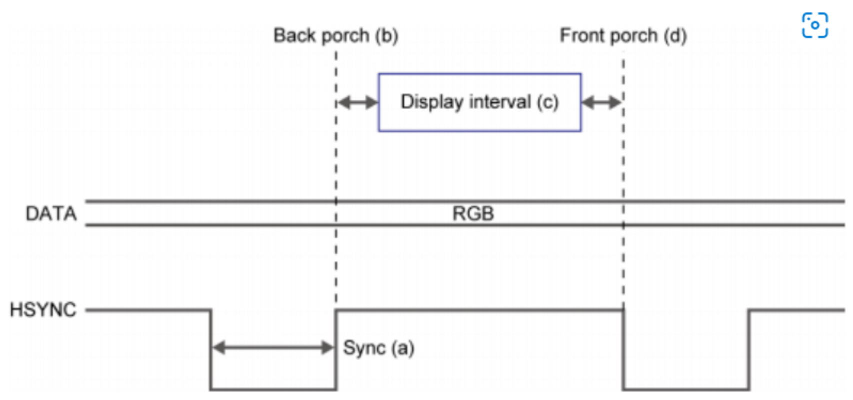
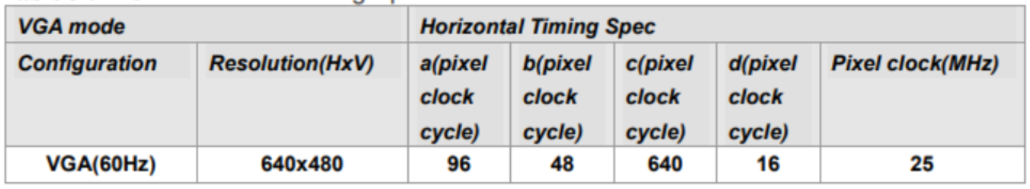


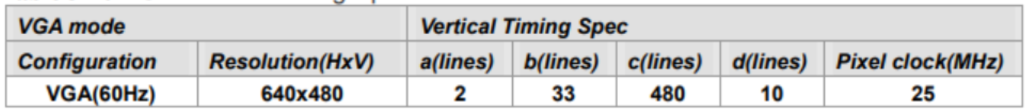
Figure 6.3 Horizontal Timing Specifications

Table 3-9 VGA Horizontal Timing Specification



VGA mode		***Horizontal Timing Spec***				
Configuration	***Resolution(HxV)***	***a(pixel clock cycle)***	***b(pixel clock cycle)***	***c(pixel clock cycle)***	***d(pixel clock cycle)***	***Pixel clock(MHz)***
VGA(60Hz)	**640x480**	**96**	**48**	**640**	**16**	**25**

Table 3-10 VGA Vertical Timing Specification



VGA mode		***Vertical Timing Spec***				
Configuration	***Resolution(HxV)***	***a(lines)***	***b(lines)***	***c(lines)***	***d(lines)***	***Pixel clock(MHz)***
VGA(60Hz)	**640x480**	**2**	**33**	**480**	**10**	**25**

Figure 1: The VGA timing specifications. The horizontal component has an 800 clock cycle period and displays for 640 cycles. The vertical component has a 525 clock cycle period and displays for 480 cycles.

To create the necessary timing a counter is needed for the horizontal and vertical components. The total number of clock cycles they count for is the sum of time periods a, b, c, and d in Figure 1. The horizontal and vertical sync (h_sync and v_sync) times are given by a in Figure 1. Figure 2 shows the schematic diagram for implementing the timing for the horizontal component. The vertical component will be the same with the values changed to match the timing in Figure 1. The clock signal for the vertical component would be the output of the second comparator of the horizontal component instead of the 25 MHz clock so that the display only moves down after the row is complete.

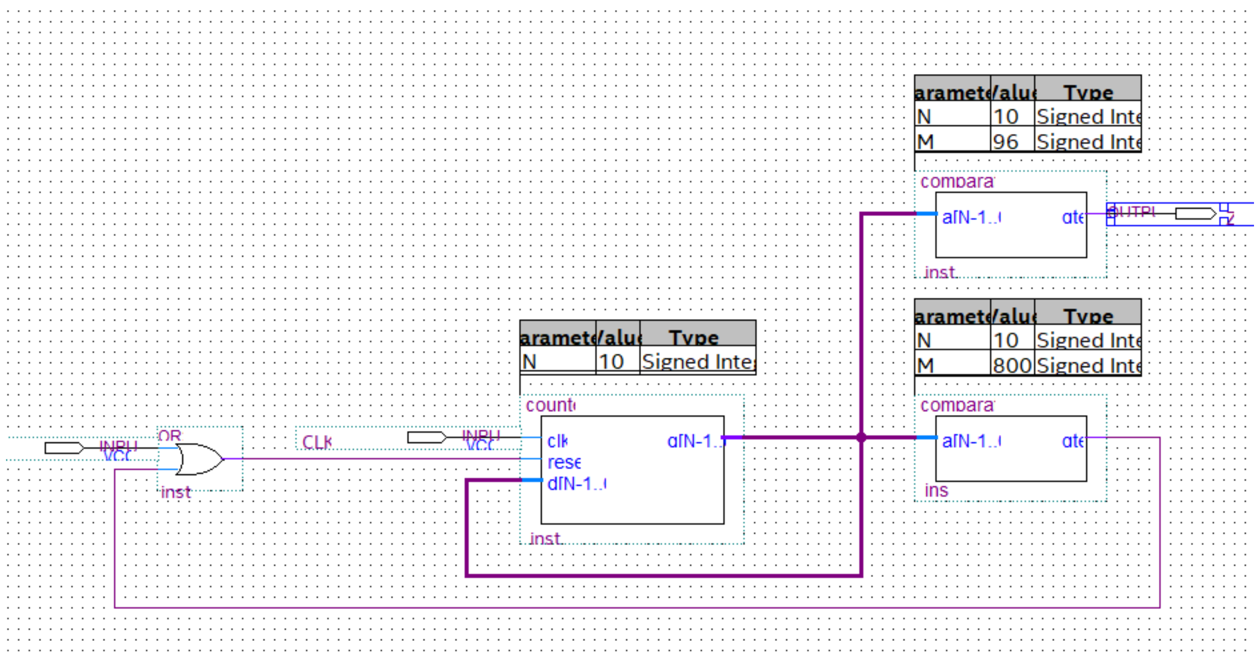


Figure 2: The schematic for the timing of the horizontal component of the VGA display. For the vertical component the M value for the first comparator would be 2 and the M value for the second comparator would be 525. The HDL for the counter is in Appendix 1 and the HDL for the comparator is in Appendix 2.

In addition to creating the h_sync and v_sync signals the timing component of the lab has to generate an enable signal that controls when the VGA displays the color. The enable needs to be HIGH when the horizontal and vertical counters are between their corresponding a + b and a + b + c. This ends up being when the horizontal counter is between 144 and 784 and when the vertical counter is between 35 and 515. The schematic for the enable is shown in Figure 3.

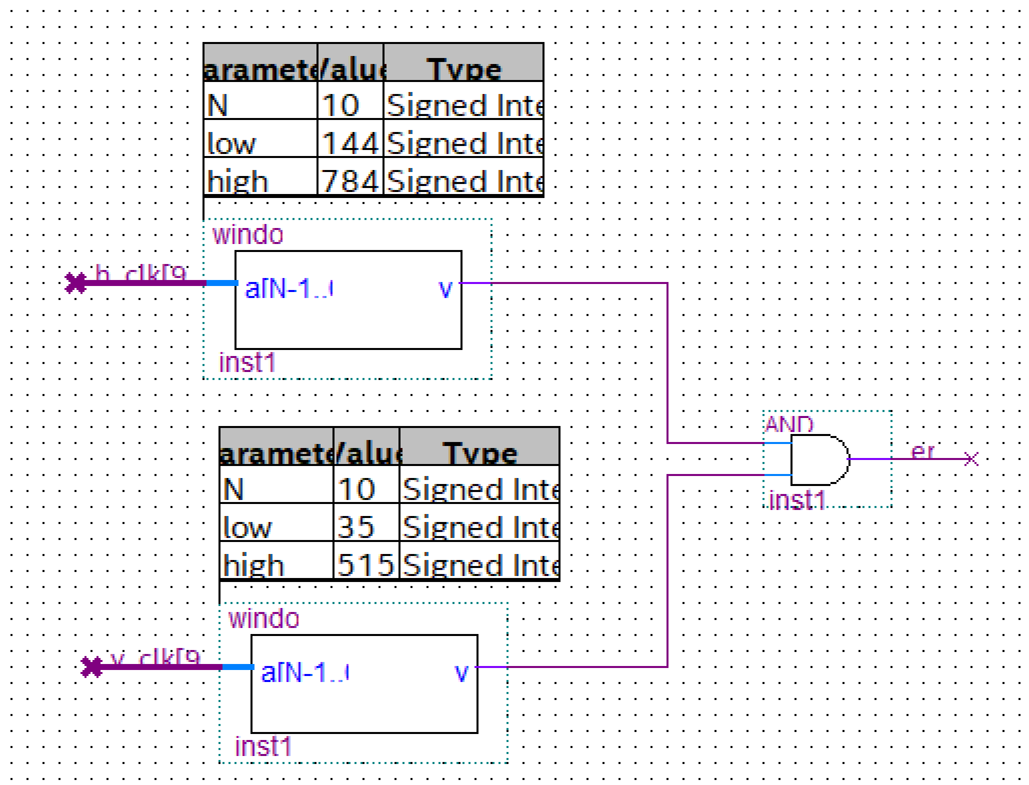


Figure 3: The schematic for how the enable signal is generated. The input signals are the outputs of the horizontal and vertical counters. The HDL for the window comparators is in Appendix 3. The low value is inclusive while the high value is not.

The complete timing section of the lab is shown in Figure 4. The final schematic includes counters for the horizontal and vertical timing and to divide the 50 MHz clock signal. It includes six comparators to check the sync, display, and reset conditions for both the horizontal and vertical counters.

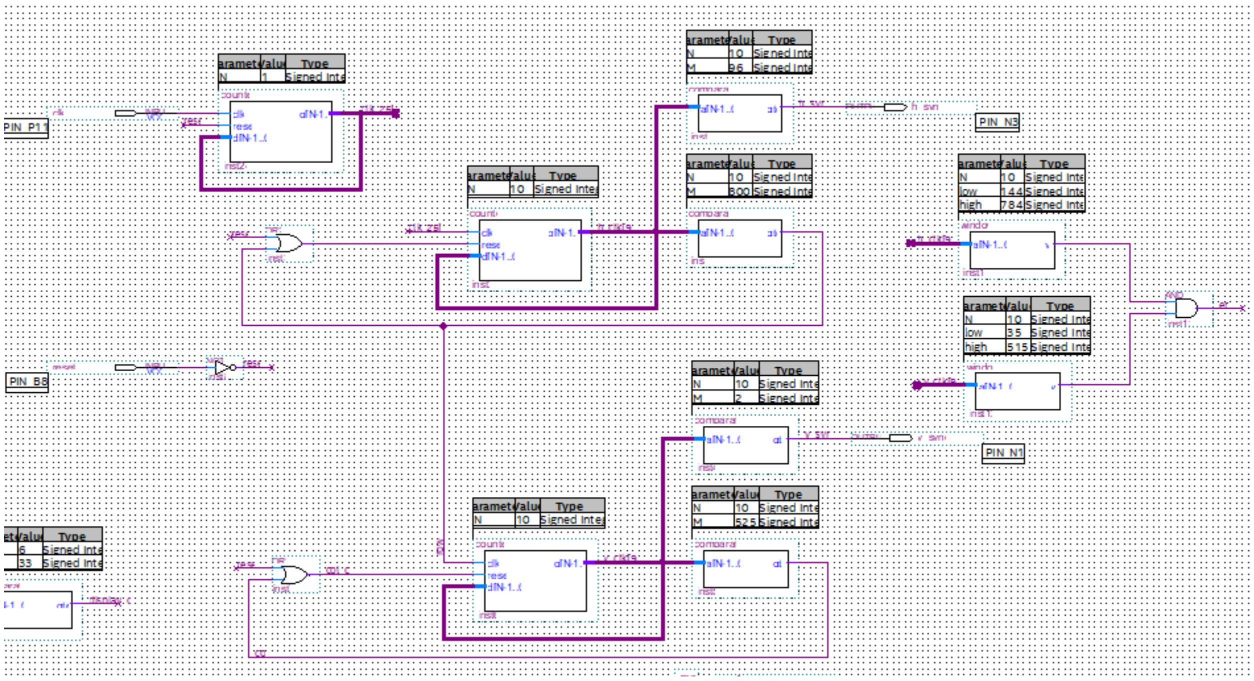


Figure 4: The schematic for the timing section. There is an additional counter to slow the 50 MHz clock to the required 25 Mhz.

The timing section of the lab interfaces with the display section as shown in the block diagram in Figure 5.

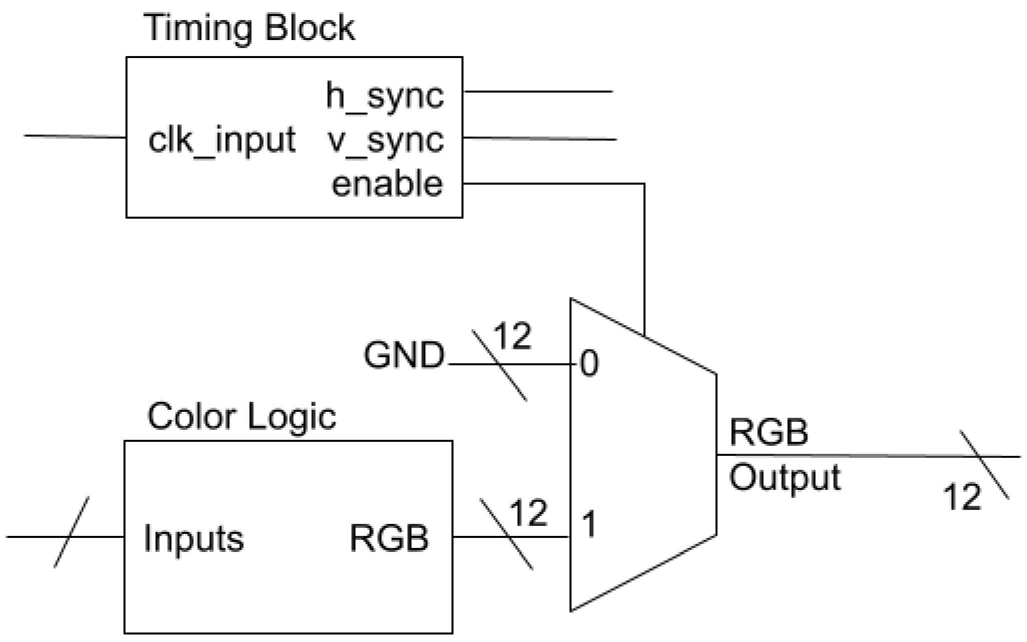


Figure 5: Block diagram showing the two major components of the lab and how they fit together. The input bus isn't labeled with a number because it is six for the lab and three for the extra credit.

The color logic for the lab and extra credit is different. For the lab the logic included a decoder to convert the two bit input for each color into a four bit signal. The decoder simply repeated the input so 10 became 1010. The schematic showing this logic is in Figure 6.

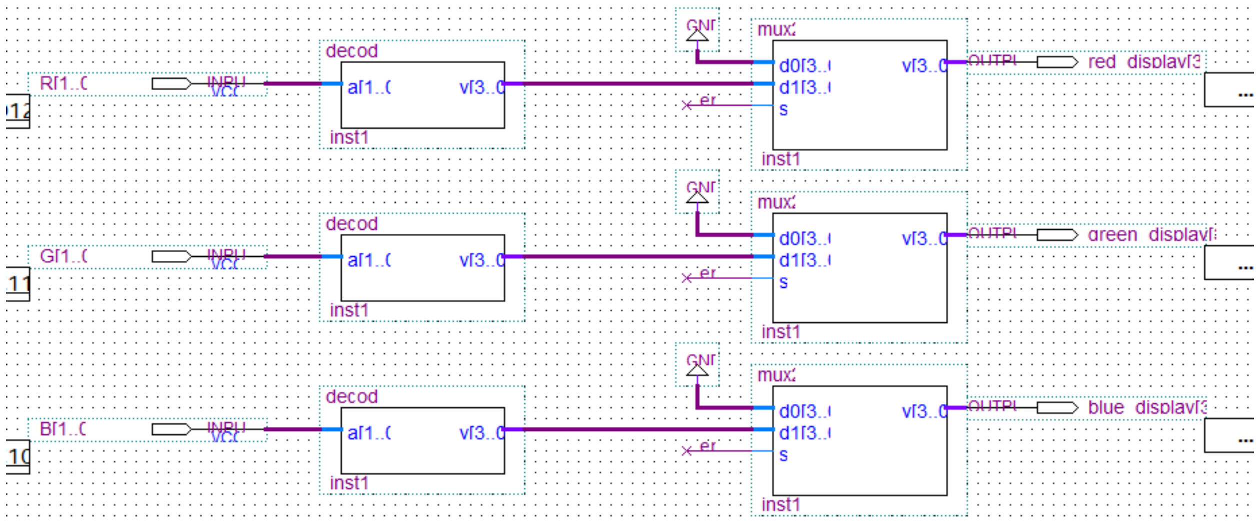


Figure 6: The schematic for the color and display logic. The HDL for the decoder and mux2 is shown in Appendices 4 and 5 respectively.

For the extra credit component the color logic only took in three inputs, one bit for each color. Rather than decode the input an incrementer added the input to 0 every 32 rows. The schematic for this logic is shown in Figure 7.

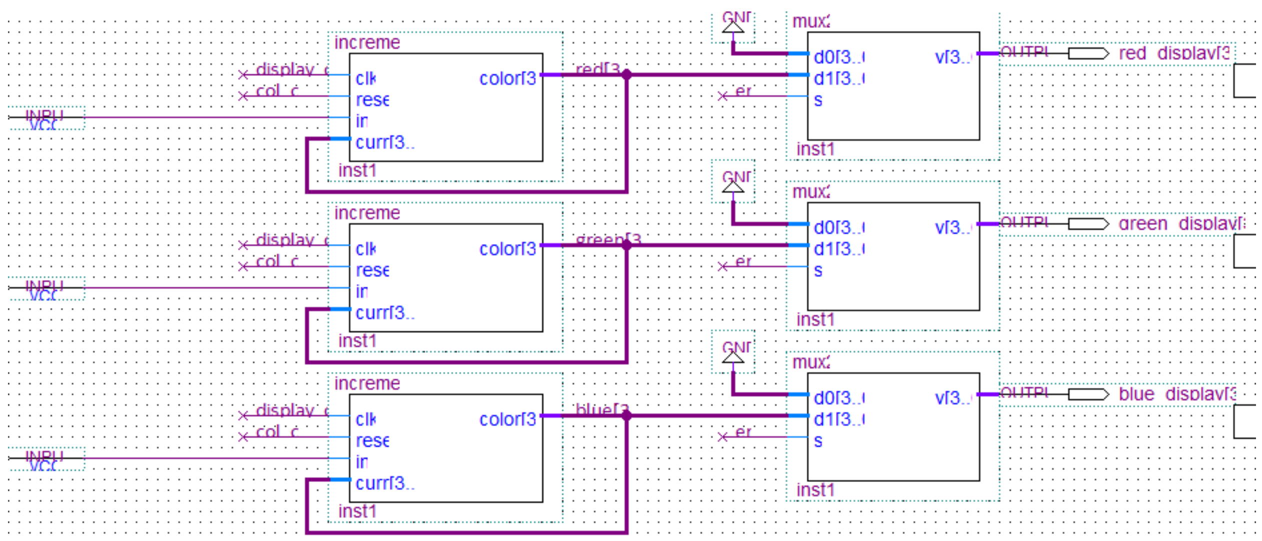


Figure 7: The schematic for the color and display logic. The clk signal to each increment block is provided by a counter that counts to 32 rows. The HDL for the increment block is shown in Appendix 6.

Interface Name	Interface Purpose	Input or Output	Number of Pins	Active High or Low	Board Label	FPGA Pin
CLK_50 MHz	Clock input	Input	1	HIGH	MAX10_ CLK1_50	Pin_P11
Reset	Gate input	Input	1	LOW	Key 0	Pin_B8
R[1..0]	Gate input	Input	2	HIGH	Switches 5 and 4	Pin_B12 Pin_A12
G[1..0]	Gate input	Input	2	HIGH	Switches 3 and 2	Pin_C12 Pin_D12
B[1..0]	Gate input	Input	2	HIGH	Switches 1 and 0	Pin_C11 Pin_C10
h_sync	Gate output	Output	1	HIGH	VGA_HS	Pin_N3
v_sync	Gate output	Output	1	HIGH	VGA_VS	Pin_N1
red_display[3..0]	Gate output	Output	4	HIGH	VGA_R[3..0]	Pin_Y1 Pin_Y2

						Pin_V1 Pin_AA2
green_display[3..0]	Gate output	Output	4	HIGH	VGA_G[3..0]	Pin_R1 Pin_R2 Pin_T2 Pin_W1
blue_display[3..0]	Gate output	Output	4	HIGH	VGA_B[3..0]	Pin_N2 Pin_P4 Pin_T1 Pin_P1

Figure 8: Interface table showing how the inputs and outputs interface with the FPGA. The pin corresponding to the most significant bit is listed at the top of each cell.

3. Results

All the results of tests on the FPGA and simulation matched the expected results. The first test that was conducted was after writing the code for the decoder. The simulation results are shown in Figure 9. The rest of the simulations show the timing. The simulations show that the enable signal is HIGH only during the periods when the VGA should be displaying color. Figure 10 shows that the sync signals operate correctly. Figures 12 and 13 show the enable signal going HIGH and going LOW at the appropriate times. Figure 14 shows the correct timings for the beginning of the screen.

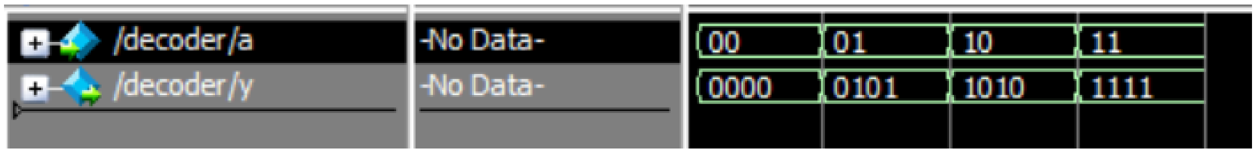
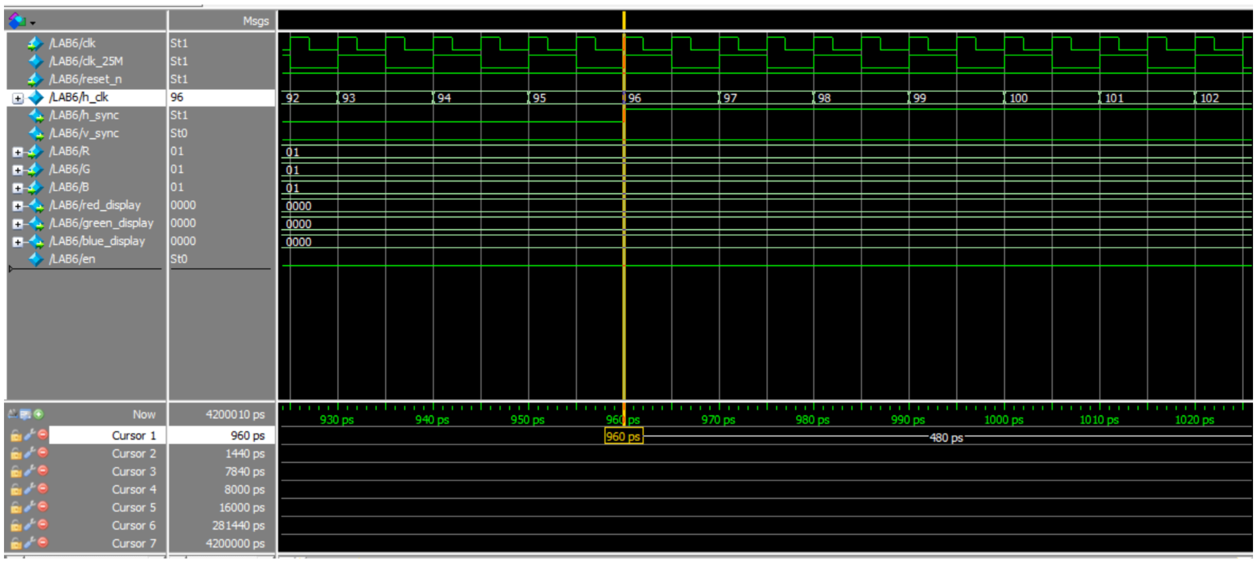


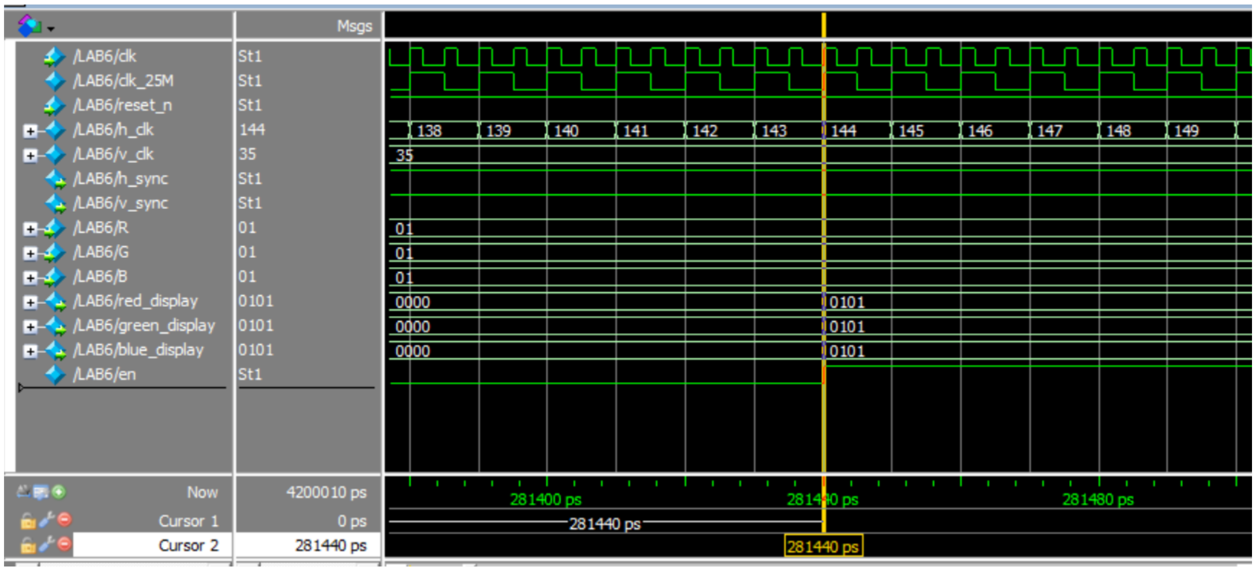
Figure 9: Simulation of the decoder module. Shows that the decoder correctly doubles the input.



```
/LAB6/clk            St1
/LAB6/clk_25M        St1
/LAB6/reset_n        St1
/LAB6/h_clk          96
/LAB6/h_sync         St1
/LAB6/v_sync         St0
/LAB6/R              01
/LAB6/G              01
/LAB6/B              01
/LAB6/red_display    0000
/LAB6/green_display  0000
/LAB6/blue_display   0000
/LAB6/en             St0

Now       4200010 ps
Cursor 1  960 ps
Cursor 2  1440 ps
Cursor 3  7840 ps
Cursor 4  8000 ps
Cursor 5  16000 ps
Cursor 6  281440 ps
Cursor 7  4200000 ps
```

Figure 10: Simulation showing the change of h_sync from LOW to HIGH when h_clk hits 96.



```
/LAB6/clk            St1
/LAB6/clk_25M        St1
/LAB6/reset_n        St1
/LAB6/h_clk          144
/LAB6/v_clk          35
/LAB6/h_sync         St1
/LAB6/v_sync         St1
/LAB6/R              01
/LAB6/G              01
/LAB6/B              01
/LAB6/red_display    0101
/LAB6/green_display  0101
/LAB6/blue_display   0101
/LAB6/en             St1

Now       4200010 ps
Cursor 1  0 ps
Cursor 2  281440 ps
```

Figure 11: Simulation showing the enable signal switching from LOW to HIGH when h_clk is 144 and v_clk is 35.

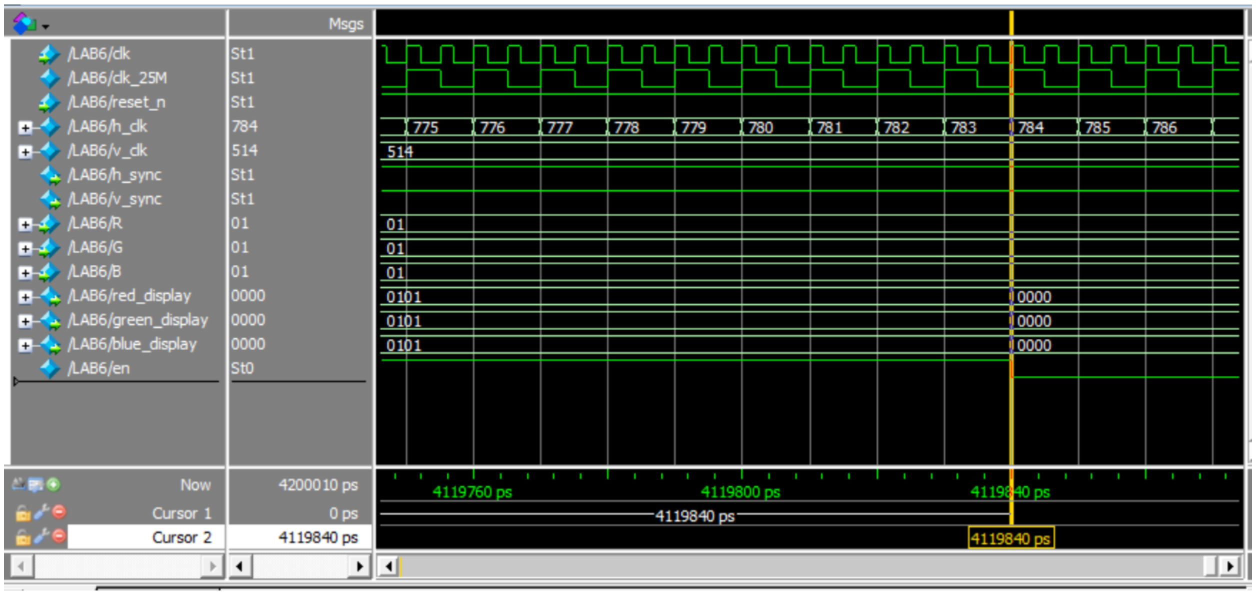


Figure 12: Simulation showing enable switching from HIGH to LOW when h_clk was 784 and v_clk was 514.

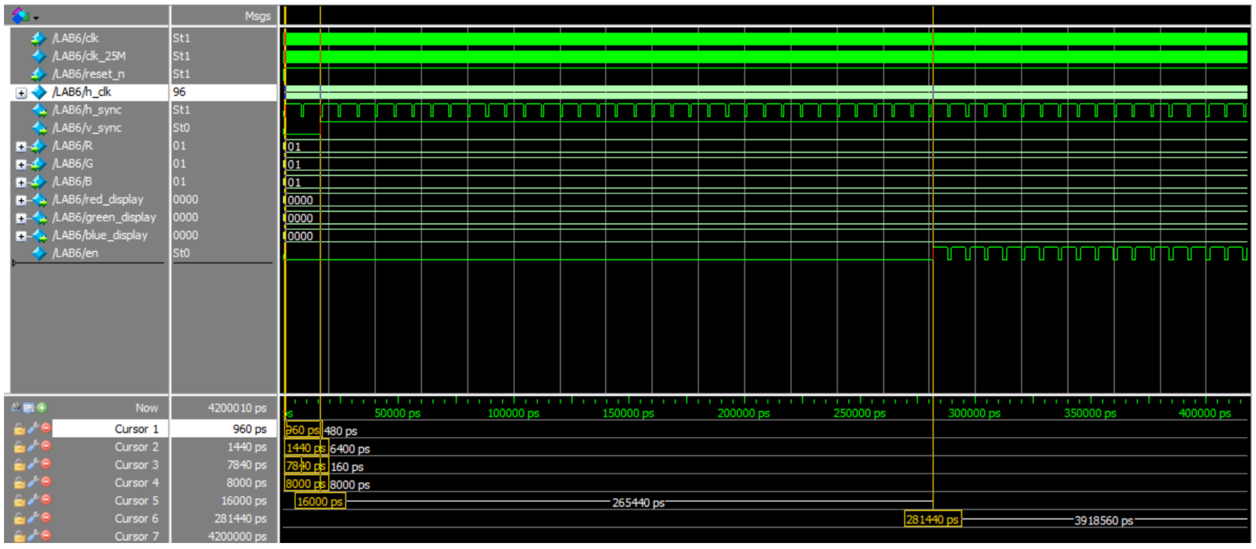


Figure 13: Simulation showing the timing before the VGA begins displaying. The cursors show the timing between each section. These timings match the specifications given in Figure 1.

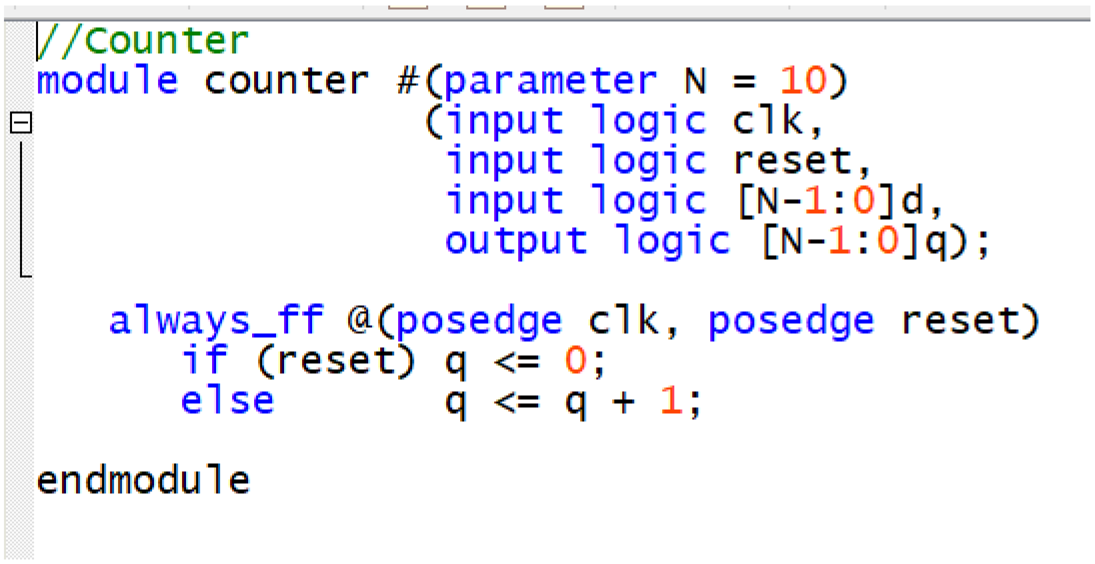
4. Experiment Notes

This lab went fairly smoothly. I didn't run into any major issues during this lab. I enjoyed completing the extra credit. I think I should've started counting the rows at the point they should be displayed so that I didn't begin incrementing before displaying the color. I tried to include more inputs for the extra credit, but ran out of the space on the FPGA which I thought was interesting.

5. Study Questions

1. How many pixels are displayed in a given frame of this VGA screen?
   a. 640 * 480 = 307200 pixels in the VGA screen.
2. What is the frame rate of this video driver?
   a. 800 * 525 = 420,000 clock cycles at 25 MHz.
   b. 25,000,000 / 420,000 = 59.52 frames per second.
3. How many possible colors can be displayed by this video driver?
   a. For the lab design with the decoder 64 colors could be displayed.
   b. For the extra credit 7 different colors of fade could be displayed.
   c. The VGA takes in a 12 bit input, so it could display 4096 different colors.

6. Appendix

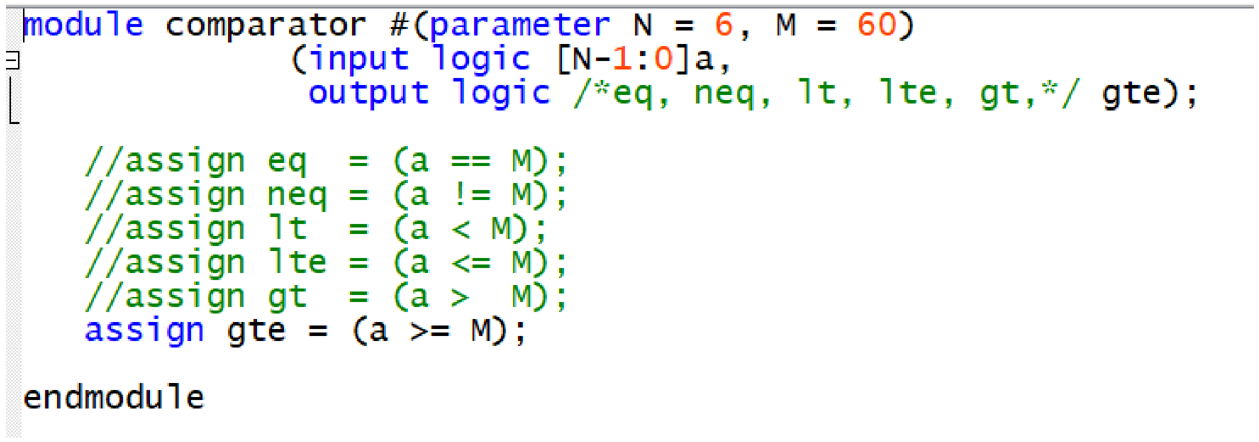


```
//Counter
module counter #(parameter N = 10)
                (input logic clk,
                 input logic reset,
                 input logic [N-1:0]d,
                 output logic [N-1:0]q);

   always_ff @(posedge clk, posedge reset)
      if (reset) q <= 0;
      else       q <= q + 1;

endmodule
```

Appendix 1: System Verilog code for the counter.

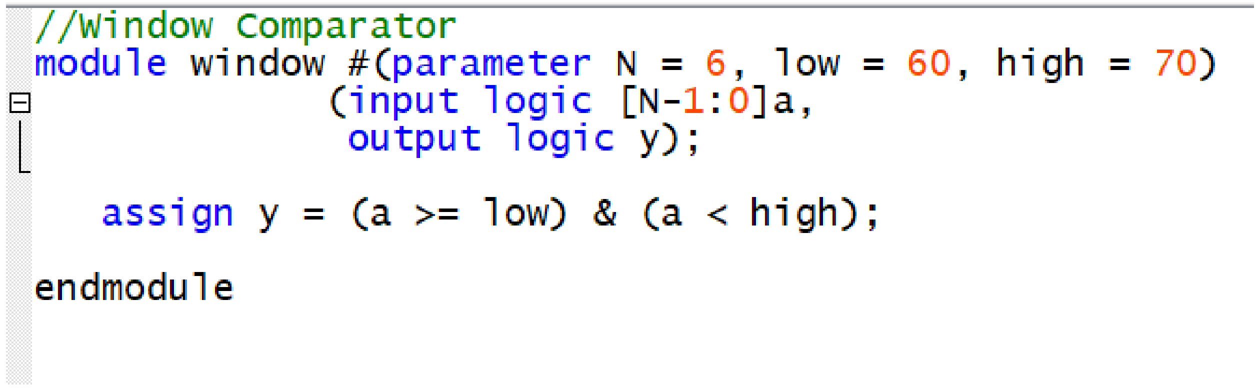


```
module comparator #(parameter N = 6, M = 60)
                (input logic [N-1:0]a,
                 output logic /*eq, neq, lt, lte, gt,*/ gte);

   //assign eq  = (a == M);
   //assign neq = (a != M);
   //assign lt  = (a < M);
   //assign lte = (a <= M);
   //assign gt  = (a >  M);
   assign gte = (a >= M);

endmodule
```

Appendix 2: The System Verilog code for the comparator. The only function needed for this lab was greater than or equal to.

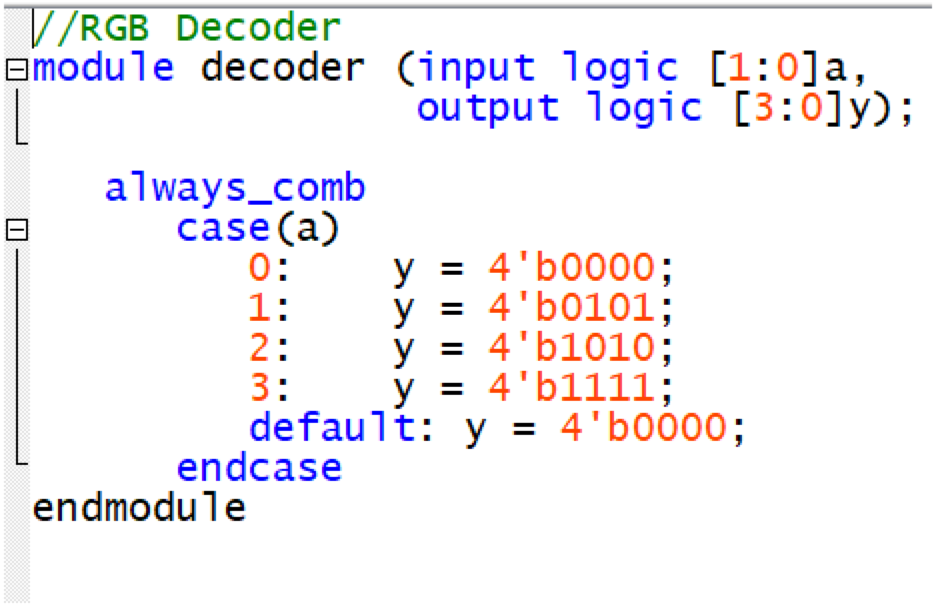


```
//Window Comparator
module window #(parameter N = 6, low = 60, high = 70)
                (input logic [N-1:0]a,
                 output logic y);

   assign y = (a >= low) & (a < high);

endmodule
```

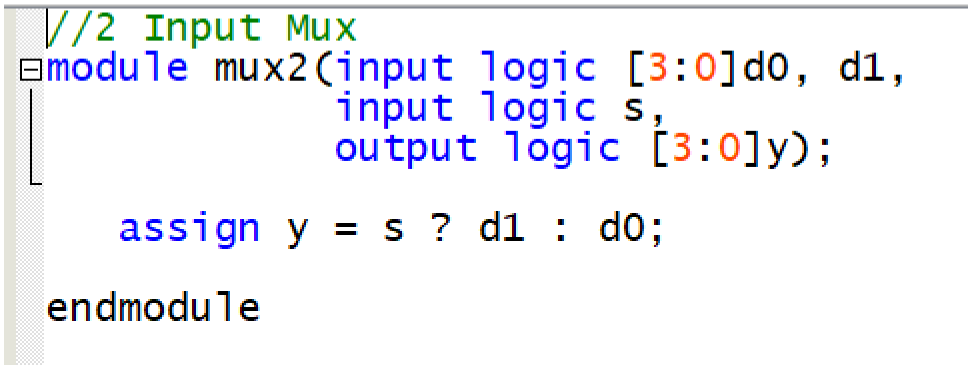
Appendix 3: The System Verilog code for the window comparator. It outputs high when the input is above or equal to the low parameter and lower than the high parameter.



```
//RGB Decoder
module decoder (input logic [1:0]a,
                output logic [3:0]y);

   always_comb
      case(a)
         0:     y = 4'b0000;
         1:     y = 4'b0101;
         2:     y = 4'b1010;
         3:     y = 4'b1111;
         default: y = 4'b0000;
      endcase
endmodule
```

Appendix 4: The System Verilog code for the decoder. This decoder converts a two bit input into a four bit signal such that the two bit input is repeated. For example, 01 becomes 0101.

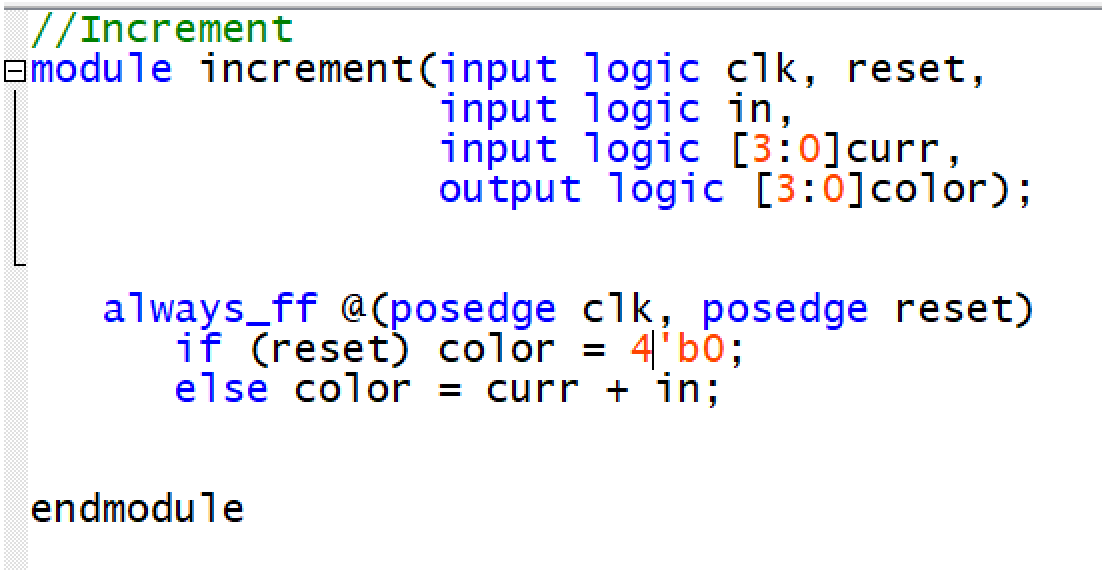


```
//2 Input Mux
module mux2(input logic [3:0]d0, d1,
            input logic s,
            output logic [3:0]y);

   assign y = s ? d1 : d0;

endmodule
```

Appendix 5: The System Verilog code for the two input multiplexer.



```
//Increment
module increment(input logic clk, reset,
                 input logic in,
                 input logic [3:0]curr,
                 output logic [3:0]color);


   always_ff @(posedge clk, posedge reset)
      if (reset) color = 4'b0;
      else color = curr + in;


endmodule
```

Appendix 6: The System Verilog for the increment block.