University of Essex
School of Computer Science and Electronic Engineering
Department of Computer Systems Engineering

CE264 Digital Systems Design

Laboratory Design Report

Prepared by
Waleed Bin Asad
1805967
wa18382@essex.ac.uk
Computer Systems Engineering

Table of Contents

Table of Figures

Summary

There are two stages in this report, the first stage2 is to describe the final design of a sub-system to output raster drive signals on the VGA port (to drive a VGA computer monitor). The second stage3 is the main component that is to design a system to create a dynamic/programmable video test pattern.

The major points documented/covered in this are divided in main part for the stages 2 and 3. For stage 2, the VGA interface uses timing signals that can be generated digitally. Basys 3 board has digital outputs from the FPGA that allow you to output simple graphics to the VGA port, hence this gives a display. For stage 3, pattern required to create a designed for a 4:3 aspect ratio display (which is what you have with a 640 by 480 resolution as designed in Stage 2)

The major conclusions in this report are that when the timings of the four signals are accurate with the current frequency and time period. This would hence allow it to generate the twelve RGB outputs on the VGA computer monitor for stage 2.

1 Introduction

This report is about the VGA displays a little information about it [4], is that they were originally based on cathode-ray tubes more about CRT is talked in [8] (not the LCD flat panels that we use now). Although modern displays don't use the same technology, Also in [6] the VGA signals still have the same format. Even when using LCD displays, you have to imagine that electron beam scanning, because the signals that drive the VGA display are analogue signals based on that electron beam scan. (This is a classic case where modern technology works in a way that was originally developed for a different technology, but it has to maintain compatibility at the interface.)The Basys 3 board uses 14 FPGA signals to create a VGA port with four bits per color and two standard synch. There are 16 switches on it. The process it does is that the scan pattern known as the ***raster*** starts from the top left of the display and scan lines from left to right, working down the display. There are two signals called Vertical sync and Horizontal synch. Where ***VSYNC*** is a low going pulse and it calculates per frame whereas ***HSYNC*** indicated the start of each scan line. Based on this idea about how VGA works, we were given to design a sub circuit (basically creating hierarchical blocks in Multisim and using existing VHDL to provide the implementation on the FPGA board in [1].) that would use 640 x 480 raster, and a 25 MHz pixel clock. The inputs that needs to be focused on is the ***PIXEL CLOCK*** and the key outputs that's required are five different signals. ***VSYNC, HSYNC, BLANK, VCOUNT, HCOUNT***. These signals would then be connected to their specified pins on the VGA port and hence depending on each switch connected to the colored bits it would display a block of color on the VGA computer monitor. After getting stage 2 working there was another part stage 3 where by inputting a 12-bit value on the switches, allowing it to select the square to be accessed using the cursor keys, and the centre cursor key as the LOAD button. The currently accessed square can be shown by flashing, or by displaying its coordinate on the 7-segment display, or both (there are three rows and four columns, so to do this we only need one digit for each coordinate).

2 Design Outline

The task we were given was to design a hierarchical blocks that would display a 640 x 480 raster using 25 MHZ pixel clock and 60+/-Hz refresh. To get this working we had to figure out the two pulses VSYNC and HSYNC that scans pattern, starting at the top left of the display and scan lines from left to right. The pixel clock used defines the timings available to display one pixel of information. Once we have VYSNC and HSYNC correct, we could then drive the red/green/blue outputs from the switches on the board, and thus control the colour of the display. After that, we can create VCOUNT and HCOUNT and use them to generate simple patterns on the display this will really confirm that our circuit is working properly Further in the report the schematics, signal timings etc. will be shown and how I've implemented my design and came to a final conclusion with the expected output. There was another stage 3 where pattern was required to create a designed for a 4:3 aspect ratio display based on the stage 2. Which will be explained later in the report.

2.1 Top-down

The final testing that generated simple patterns on the VGA display and this confirms that my timings of the signals were accurate and the LCD raster 640 x 480 with 60Hz refresh was precise. Hence the task given was completed with no errors and was a success for Stage 2. Below are few screenshots that I took during my analysis.

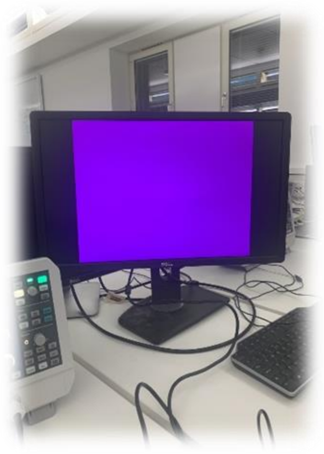


Figure 1. Purple display

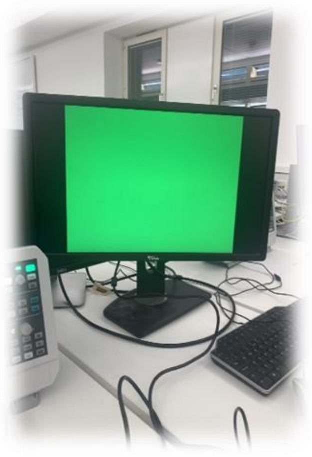


Figure 2. Green display

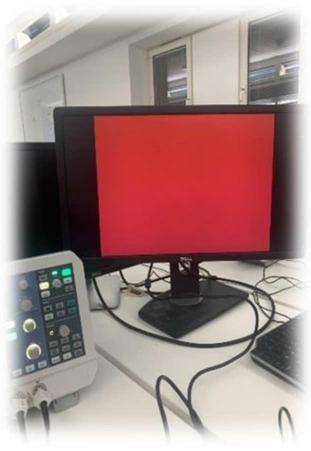


Figure 3. Red display

Here is a short video of the color changes based on the switches.

IMG_3259.MOV

2.2 Hierarchical Blocks

So giving an outline of my schematic for stage 2. As mentioned in [2], We used hierarchy blocks because this reduces the workload for identical blocks of circuitry. As seen from the *Figure 4. Schematic for RGB*, the HBLANK and VBLANK goes into a Logic OR gate and the output of it gets inverted which gives the **display** color. The idea behind it is that when the blank is high the twelve outputs must be low because the display is low and it won't copy the color bits. When the blank is low and the display is high the twelve RGB output bits are copied and hence this displays a color on the VGA monitor depending on switch.

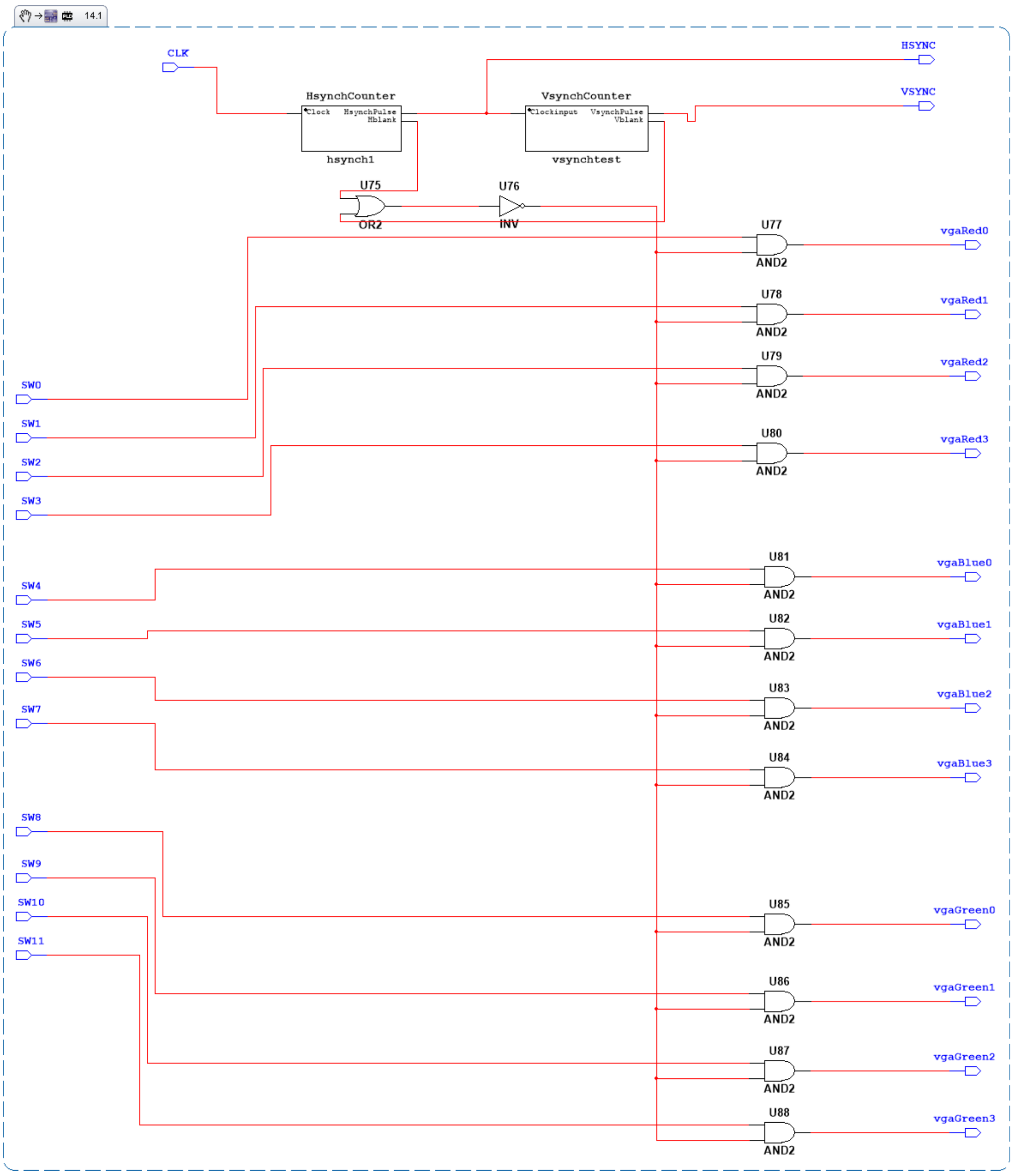
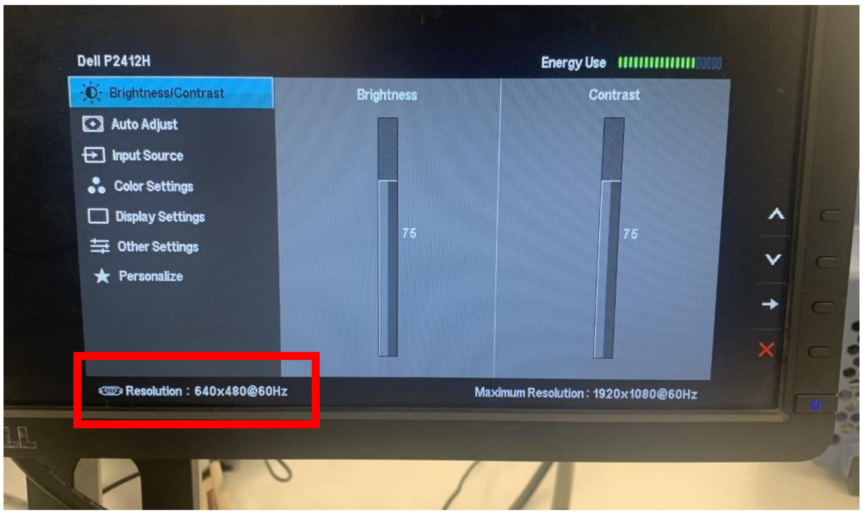


Figure 4. Schematic for RGB

This is the evidence in *Figure 5. Resolution 640 x 480* for my VSNCH and HSYNC pulses which gives the exact resolution 640 x 480 @ refresh frequency of 60 Hz. This tells us that the timings of the two signals are accurate.



Resolution : 640x480@60Hz

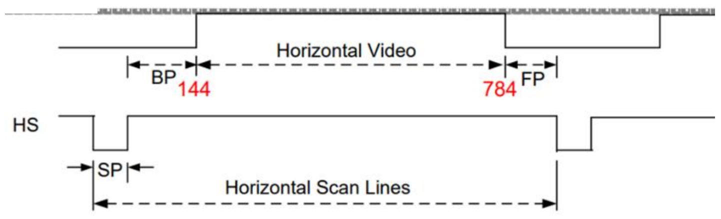
Maximum Resolution : 1920x1080@60Hz

Figure 5. Resolution 640 x 480

2.3 Horizontal Synchronization Counter

Inside the Horizontal synch counter there are two signals that are being output based on one input pixel clock. The HSYNC pulse and the HBLANK. The HSYNC is low going pulse and indicates the start of each scan line. Based on the research in [7] a video frame is made with 800 pixels which is approximately 521 lines per frame. Here is how the Horizontal Timing is calculated.

Horizontal Timing



Pixel clock = 25 MHz   Pixel time = 0.04 µs
Horizontal video = 640 pixels x 0.04 µs = 25.60 µs
Back porch, BP = 16 pixels x 0.04 µs = 0.64 µs
Front porch, FP = 16 pixels x 0.04 µs = 0.64 µs
Sync pulse, SP = 128 pixels x 0.04 µs = 5.12.µs
Horizontal Scan Lines = SP + BP + HV + FP
= 128 + 16 + 640 + 16
= 800 pixels x 0.04 µs = 32 µs
1/60 Hz = 16.67 ms / 32 µs = 521 horizontal scan lines per frame

Figure 6. Horizontal Sycn Time

So starting of with here is a small schematic diagram for the horizontal counter.

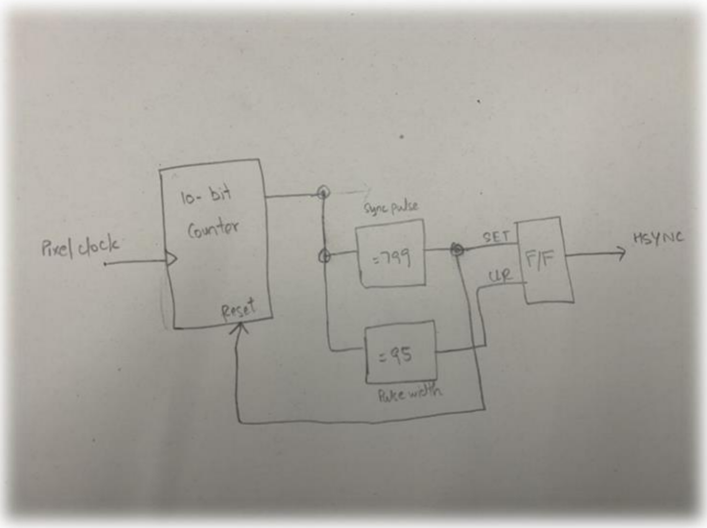


Figure 7. Horizonatl Schematic Counter

First thing is the design of the 10 bit counter which is key part to generate the clocks value (Sync pulse and the Pulse Width). We used a CNTR_4BIN_S multiplexer it's a Modulo 16 positive edge-triggered Synchronous counter as read in [3]. It is high only for one count state. In *Figure 8. 12 Bit Counter,* I used three of the multiplexer that gives twelve bits output but we only use ten of them.

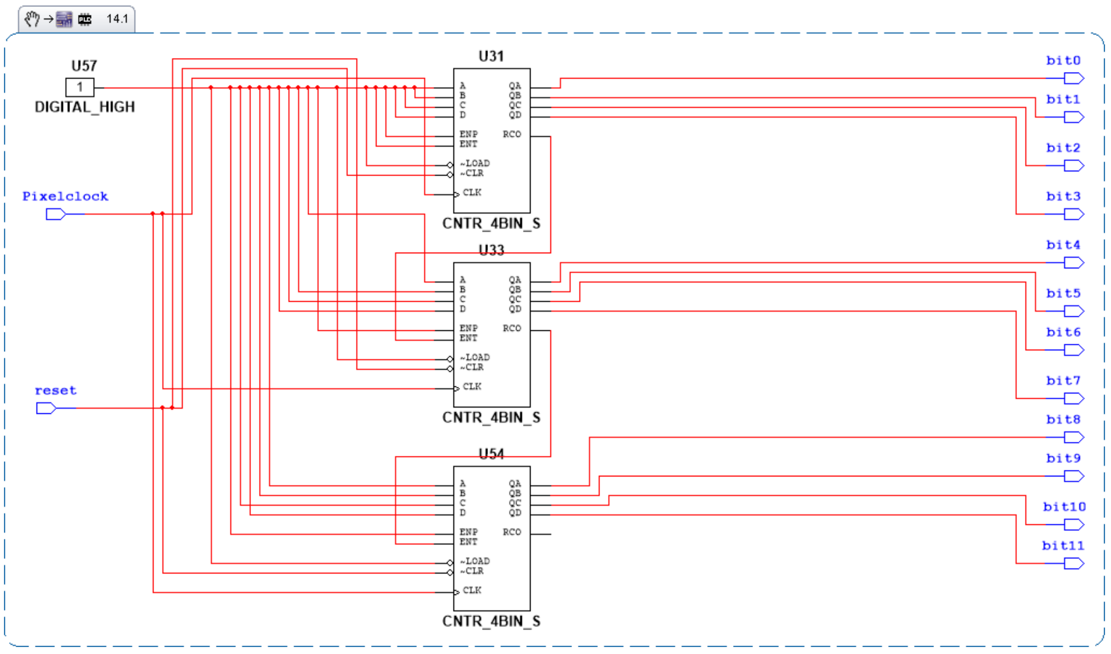


Figure 8. 12 Bit Counter

Once the 12 bit counter was working, I moved on with the Sync pulse and the pulse width. It is easy to implement using different Logic gates, and the synch pulse resets the counter when it is low. The pulse width counts 96 Clks and the Synch Pulse counts 800 Clks. I designed a clock divider which produces 25MHz output. This is the input to the 12 bit counter.

We used a FF_JK_NSCLR_CO Flip Flop after reading its truth table in [3], which is set at 799 at the start of the cycle and cleared after 95 clock cycles. It toggles when both J and K are at High state. This will switch the output only one time in every clock cycle from one state to another state.

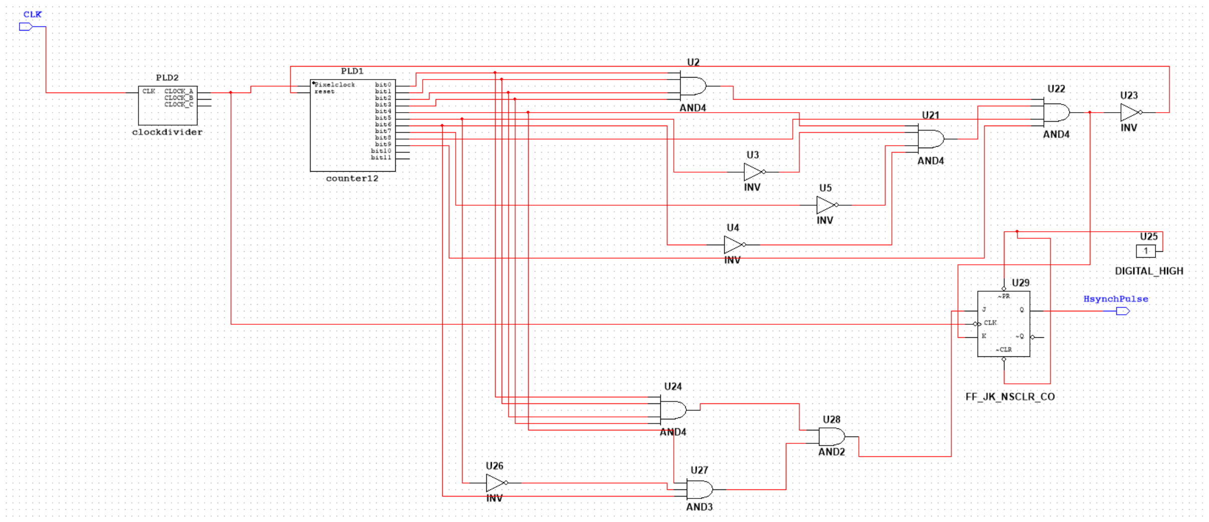
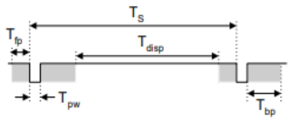
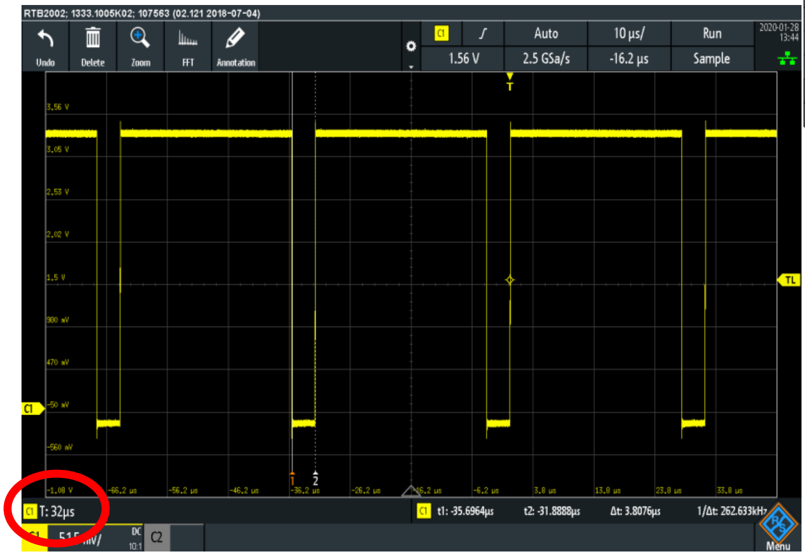


Figure 9. Hsync Pulse



Symbol	Parameter	Vertical Sync			Horiz. Sync	
		Time	Clocks	Lines	Time	Clks
T_S	Sync pulse	16.7ms	416,800	521	32 us	800
T_{disp}	Display time	15.36ms	384,000	480	25.6 us	640
T_{pw}	Pulse width	64 us	1,600	2	3.84 us	96
T_{fp}	Front porch	320 us	8,000	10	640 ns	16
T_{bp}	Back porch	928 us	23,200	29	1.92 us	48

Figure 10. Signal Timings



T: 32µs

t1: -35.6964µs
t2: -31.8888µs
Δt: 3.8076µs
1/Δt: 262.633kHz

Figure 11. Hsync pulse test

As seen from the figure 10 table the time period of the HSYNC pulse was accurate 32 microseconds.

Understanding the use of **blanking** it is a period where all RGB colour signals output LOW. During the time when old monitors were used they needed a time for a small element inside them to rotate back to the beginning of the line. Nowadays the latest monitors work on a altered principle so that period, but for the sake they added blanking in the VGA monitors, even if they don't need it and lose frame rate because of it. So I then moved on with the HBLANK, this has the same schematic as the HSYNC pulse but instead we use different values for the counter. We use the same *Fig 8. 12 Bit Counter* for both the pulses. To calculate HBLANK we acquire these values.

pw: Pulse Width

bp: Back Porch

disp: Display

Tpw + Tbp + Tdisp = 96 + 48 + 60 => **784**

Tpw + Tbp = 96 + 48 =>**144**

This is the final schematic of the Horizontal synchronisation counter

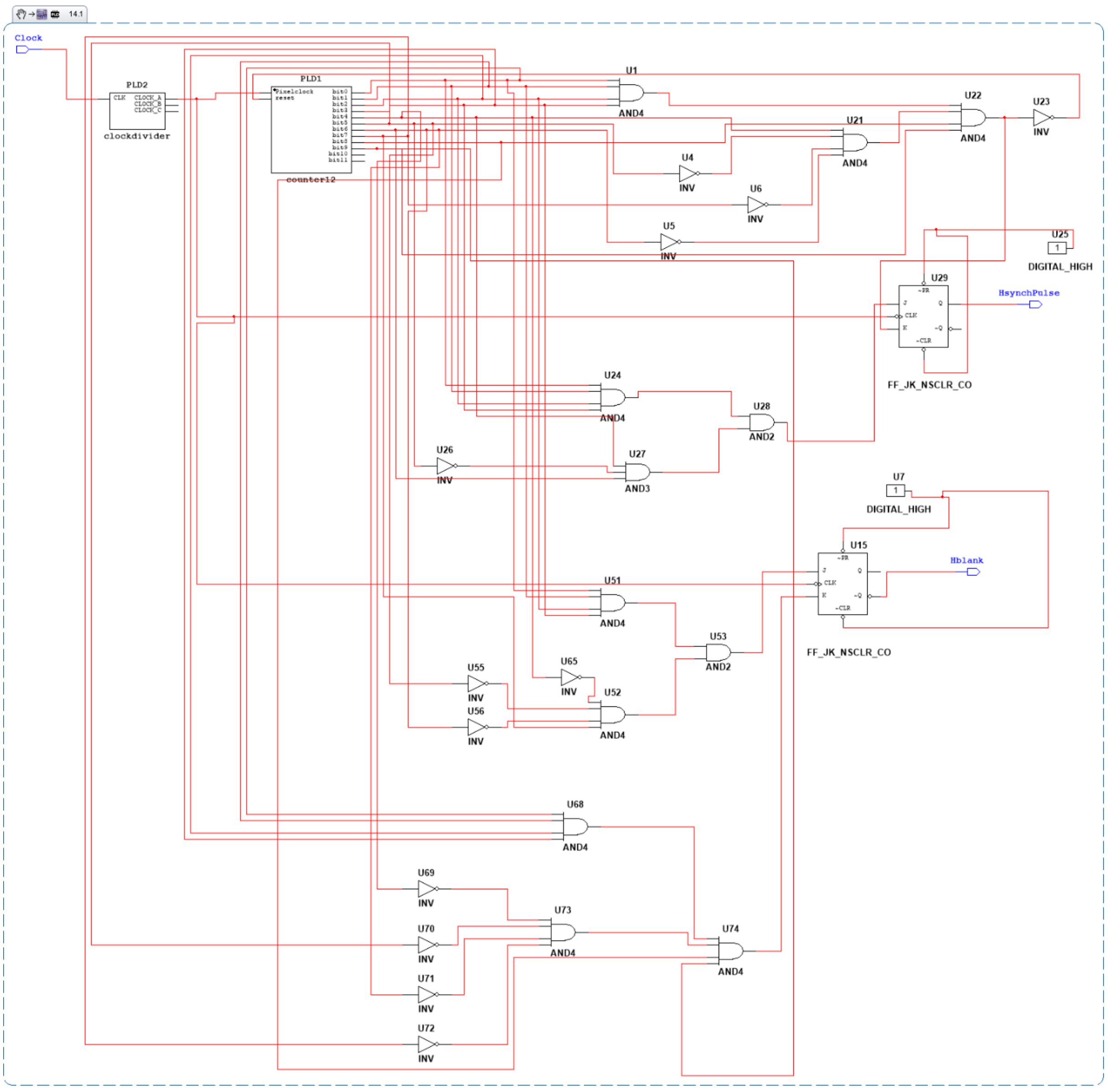
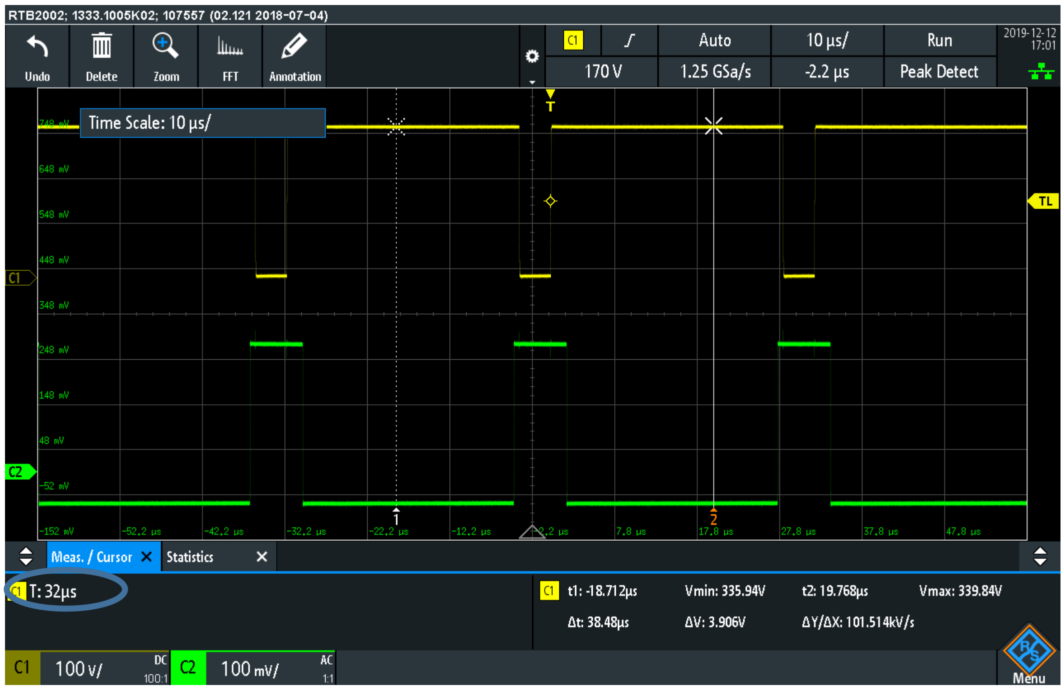


Figure 12. Final Horizontal counter Schematic

Finally after getting the Hblank and Hsynch pulse I tested them on the oscilliscope. Checking that the time period is accurate which will display a resolution of 640 px.

Key:		
	🟨	Hsync pulse
	🟩	Hblank



RTB2002; 1333.1005K02; 107557 (02.121 2018-07-04)

C1 Auto 10 µs/ Run 2019-12-12 17:01
170 V 1.25 GSa/s -2.2 µs Peak Detect

Time Scale: 10 µs/

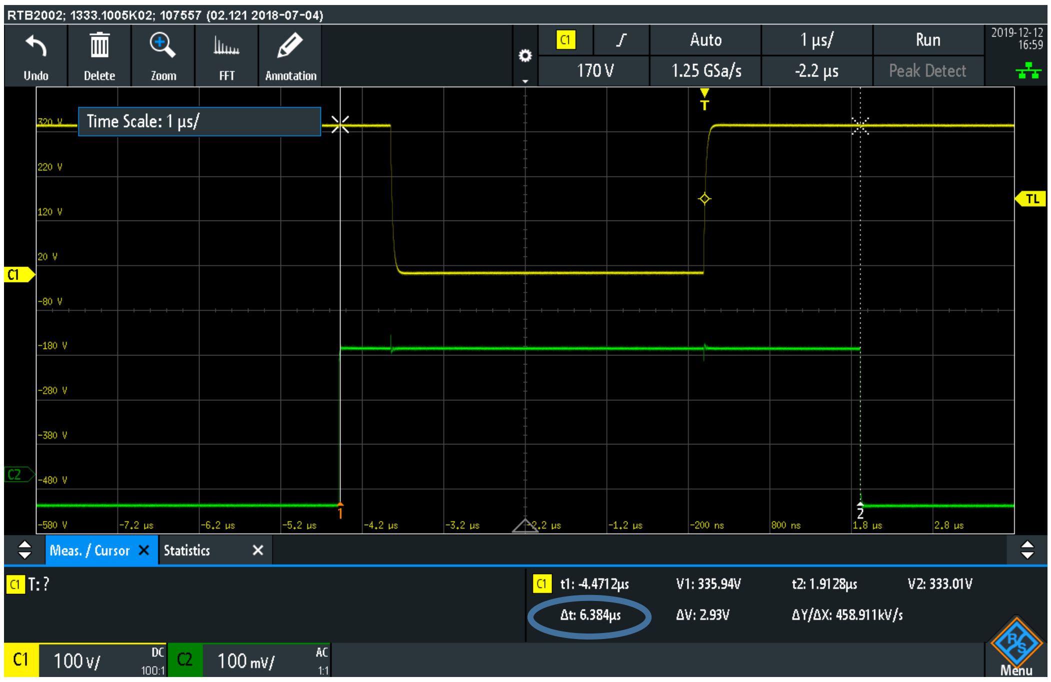
Meas. / Cursor Statistics

C1 T: 32µs

C1 t1: -18.712µs Vmin: 335.94V t2: 19.768µs Vmax: 339.84V
Δt: 38.48µs ΔV: 3.906V ΔY/ΔX: 101.514kV/s

C1 100 V/ DC 100:1 C2 100 mV/ AC 1:1

Figure 13. Hcounter result (sync pulse)



RTB2002; 1333.1005K02; 107557 (02.121 2018-07-04)

C1 Auto 1 µs/ Run 2019-12-12 16:59
170 V 1.25 GSa/s -2.2 µs Peak Detect

Time Scale: 1 µs/

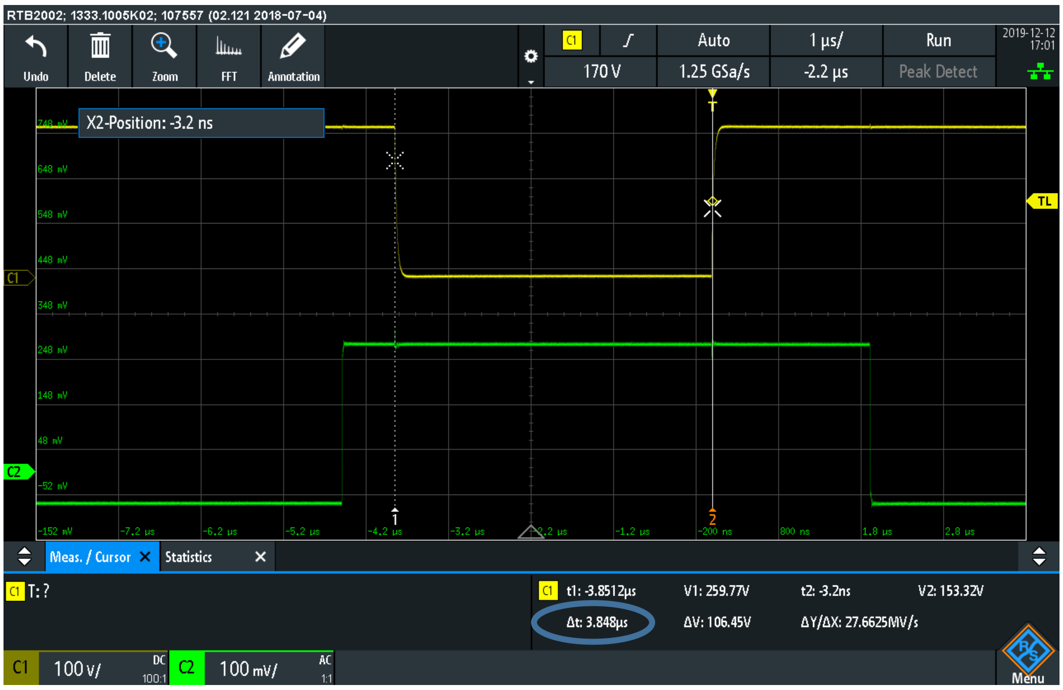
Meas. / Cursor Statistics

C1 T: ?

C1 t1: -4.4712µs V1: 335.94V t2: 1.9128µs V2: 333.01V
Δt: 6.384µs ΔV: 2.93V ΔY/ΔX: 458.911kV/s

C1 100 V/ DC 100:1 C2 100 mV/ AC 1:1

Figure 14. Testing Time period for Hblank(pulsewidth + back porch)



C1 T: ?

t1: -3.8512µs V1: 259.77V t2: -3.2ns V2: 153.32V
Δt: 3.848µs ΔV: 106.45V ΔY/ΔX: 27.6625MV/s

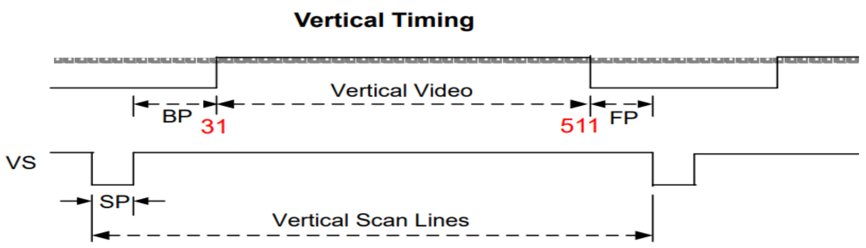
C1 100 V/ DC 100:1 C2 100 mV/ AC 1:1

Figure 15. Testing Time period for hsync pulse width

2.4 Vertical Synchronization Counter

Inside the Vertical Sync Counter there are two signal [5] VSYNC pulse and the VBLANK. The VSYNC pulse indicates the start of the raster (scan pattern: Starts at top left of the display and scan lines from left to right). It also defines the refresh frequency (60Hz) of the display. The refresh frequency is generated by the Hsync pulse which is the input to the Vscync counter. So in simples terms it refreshes the frame of the VGA display after every 60hz.

The schematic of the VSYNC pulse is the same as the HSYNC in *Fig 7. Horizontal Schematic Counter*. So keeping in mind about the diagram. Here is how the Vertical timing is calculated.



Pixel clock = 25 MHz   Horizontal scan time = 32 µs
Vertical video = 480 lines x 32 µs = 15.360 ms
Back porch, BP = 29 lines x 32 µs = 0.928 ms
Front porch, FP = 10 lines x 32 µs = 0.320 ms
Sync pulse, SP = 2 lines x 32 µs = 0.064 ms
Vertical Scan Lines = SP + BP + VV + FP
= 2 + 29 + 480 + 10
= 521 lines x 32 µs = 16.672 ms
1/60 Hz = 16.67 ms

Figure 16. Vertical Sync Time

So moving on with this we designed my Vsycn pulse using the 12 bit counter in *Fig 8. 12 Bit Counter* and the two counter values that were needed was 521 line for the sync pulse and 2 line for the pulse width. As read in [3], It uses the same FF_JK_NSCLR_CO Flip Flop which is set at 521 at the start of the cycle and cleared after 2 clock cycles. Below is the schematic:

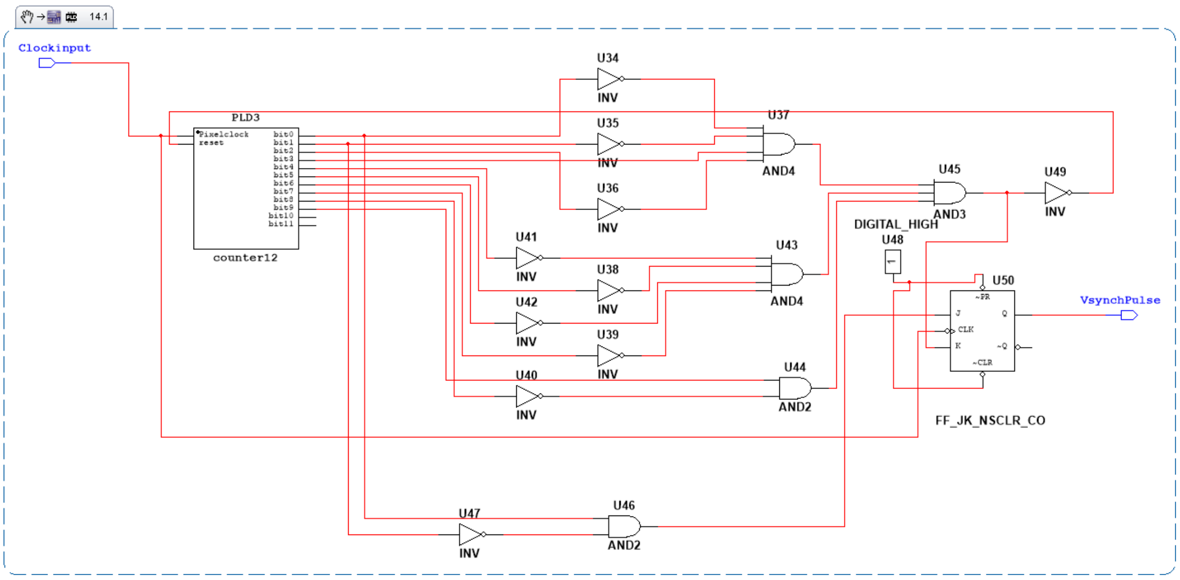
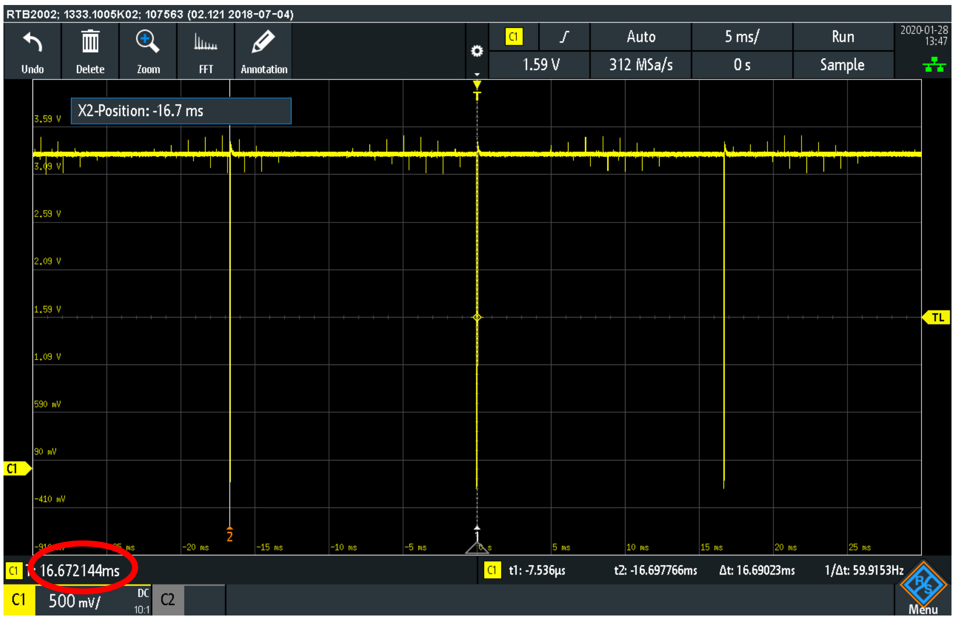


Figure 17. Vertical Sync Pulse schematic

Testing the timing period of my Vsync Pulse after the clockinput is connected with Hsync pulse.


RTB2002; 1333.1005K02; 107563 (02.121 2018-07-04)

X2-Position: -16.7 ms

Auto 1.59 V | 5 ms/ 312 MSa/s | 0 s | Run Sample | 2020-01-28 13:47

16.672144ms

t1: -7.536µs t2: -16.697766ms Δt: 16.69023ms 1/Δt: 59.9153Hz

C1 500 mV/ DC 10:1

Figure 18. Vertical Sync Pulse test

This shows that my Vertical sync pulse is accurate, after every 64 micorseconds it toggles and a new cycle starts.

Moving on with this I designed the other signal which is the VBLANK. To calculate VBLANK we acquire these values.

pw: Pulse Width

bp: Back Porch

disp: Display

Tpw + Tbp + Tdisp = 2 + 29 + 480 => 511

Tpw + Tbp = 2 + 29 =>**31**

It uses the same FF_JK_NSCLR_CO Flip Flop which is set at 511 at the start of the cycle and cleared after 31 clock cycles. Below is the schematic:

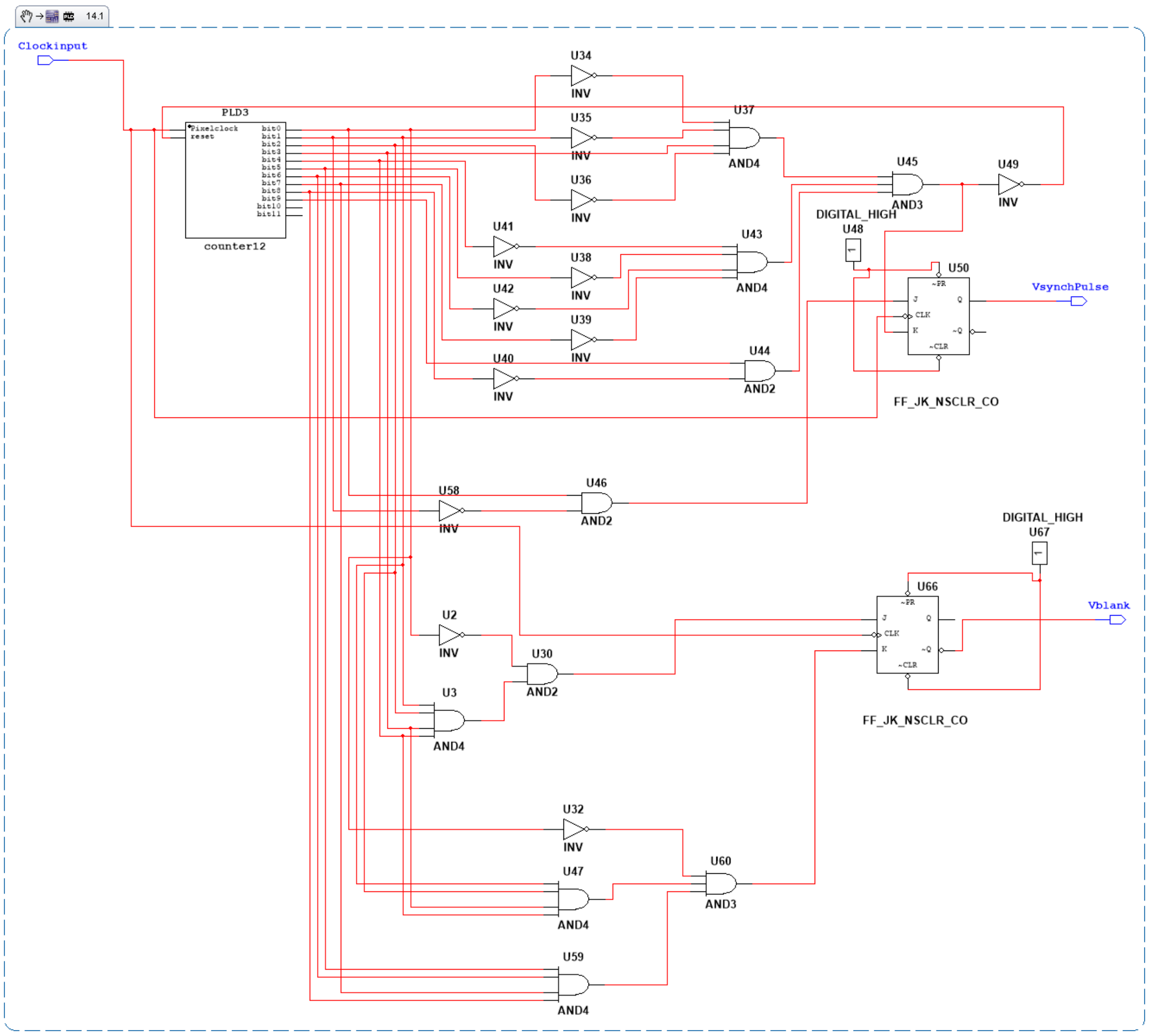
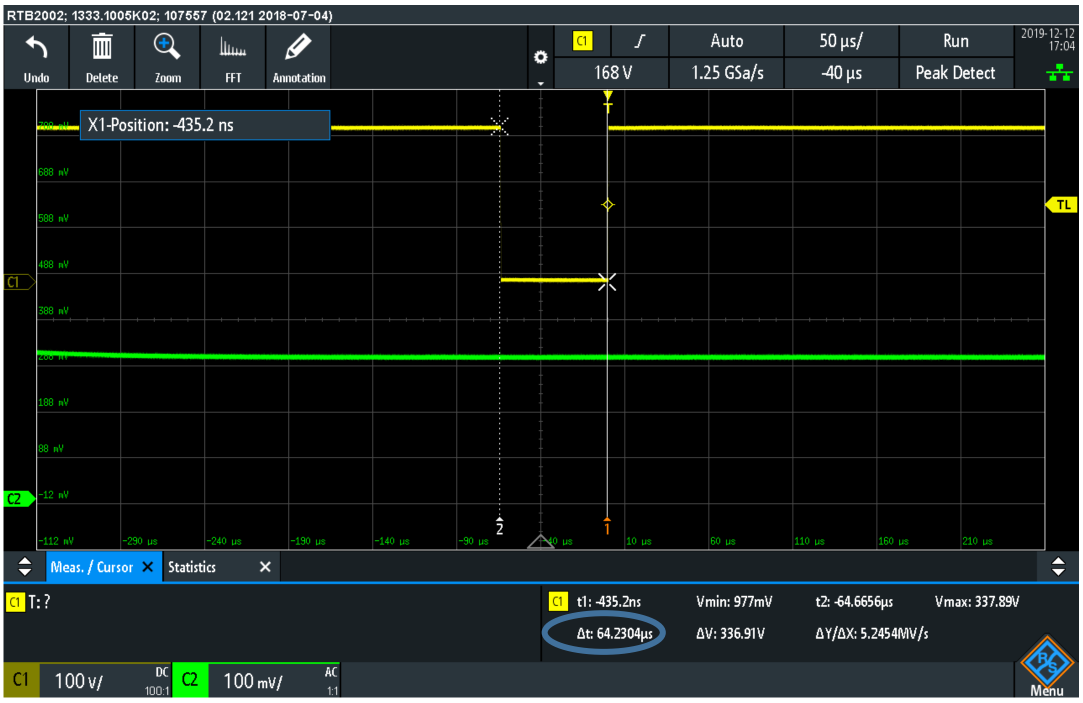


Figure 19. Final Vertical Sync Counter scehmatic

Finally after getting the Vblank and Vsynch pulse I tested them on the oscilliscope. Checking that the time period is accurate which will display a resolution of 480 px.

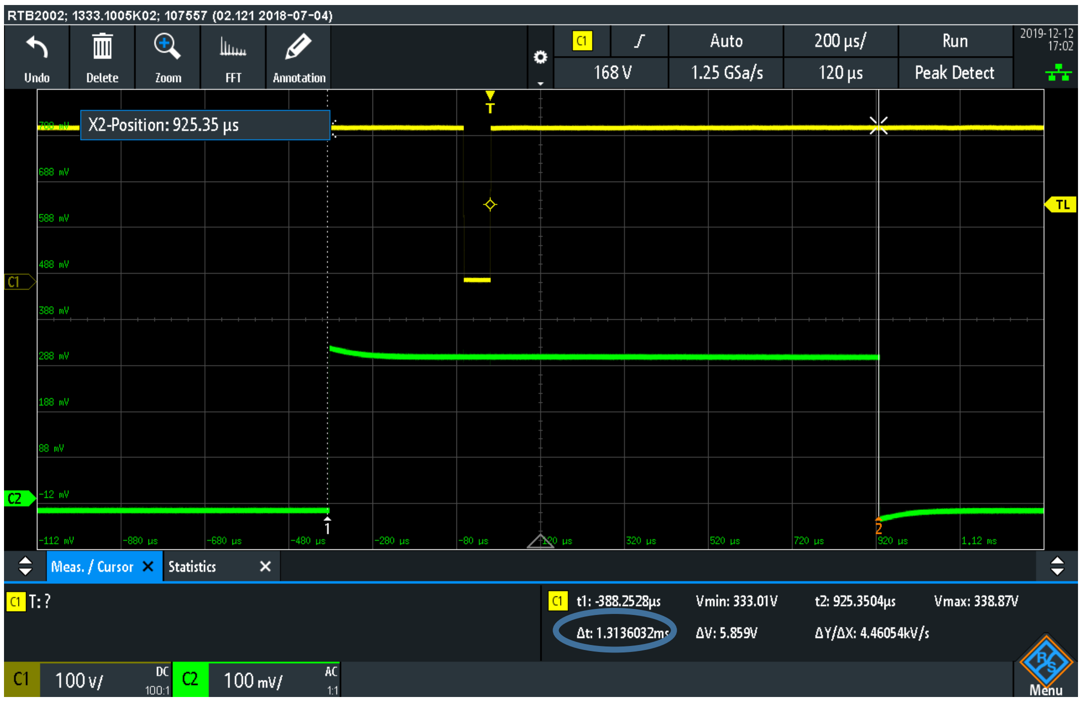
Key: 🟨 Vsync pulse

🟩 Vblank



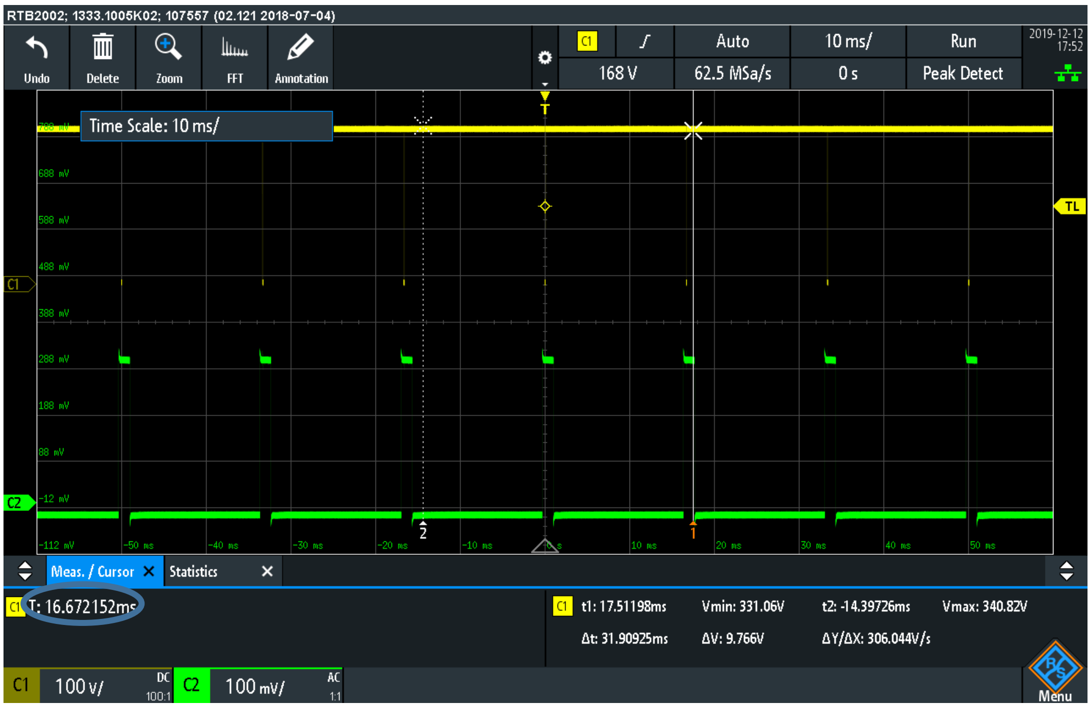
```
RTB2002; 1333.1005K02; 107557 (02.121 2018-07-04)
C1  168 V   Auto  1.25 GSa/s   50 µs/  -40 µs   Run  Peak Detect   2019-12-12 17:04
X1-Position: -435.2 ns
C1 T: ?
C1 t1: -435.2ns     Vmin: 977mV    t2: -64.6656µs    Vmax: 337.89V
   Δt: 64.2304µs    ΔV: 336.91V    ΔY/ΔX: 5.2454MV/s
C1 100 V/ DC 100:1   C2 100 mV/ AC 1:1
```

Figure 20. Tesing pulse width for Vsync



```
RTB2002; 1333.1005K02; 107557 (02.121 2018-07-04)
C1  168 V   Auto  1.25 GSa/s   200 µs/  120 µs   Run  Peak Detect   2019-12-12 17:02
X2-Position: 925.35 µs
C1 T: ?
C1 t1: -388.2528µs    Vmin: 333.01V    t2: 925.3504µs    Vmax: 338.87V
   Δt: 1.3136032ms    ΔV: 5.859V       ΔY/ΔX: 4.46054kV/s
C1 100 V/ DC 100:1   C2 100 mV/ AC 1:1
```

Figure 21. Testing time period of the VBlank (pulsewidth+Back Proch)


RTB2002; 1333.1005K02; 107557 (02.121 2018-07-04)

C1 ∫ Auto 10 ms/ Run 2019-12-12 17:52

168 V 62.5 MSa/s 0 s Peak Detect

Time Scale: 10 ms/

C1 T: 16.672152ms

C1 t1: 17.51198ms Vmin: 331.06V t2: -14.39726ms Vmax: 340.82V

Δt: 31.90925ms ΔV: 9.766V ΔY/ΔX: 306.044V/s

C1 100 V/ DC 100:1 C2 100 mV/ AC 1:1

Figure 22. Testing Vsync Pulse

2.5 Design Modification

As seen from my schematics above, it looks a bit messy and it is not properly organised. So to modify it I would have added a separate hierarchical block for the 12 switches and the 12 RGB output. This would give a clear understanding of how the comparators are being used in simple terms.

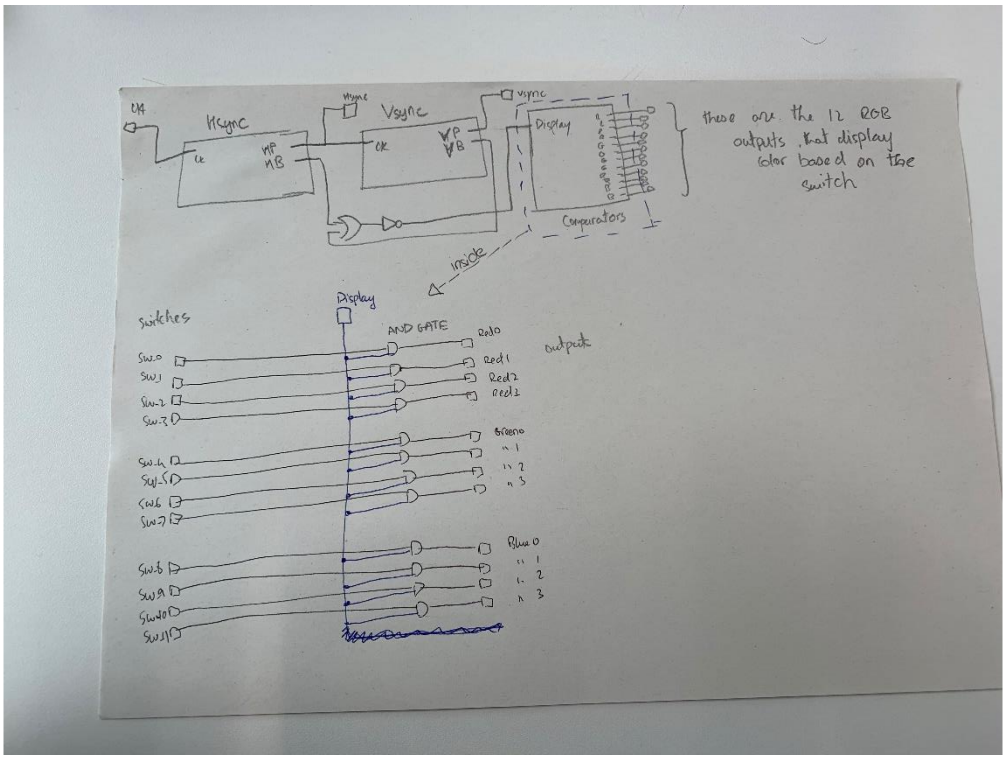


Figure 23. Modified Design

3 Stage 3

After getting stage 2 which is outlined in Part 2. This was another part stage 3 where by inputting a 12-bit value on the switches, allowing it to select the square to be accessed using the cursor keys, and the centre cursor key as the LOAD button. The currently accessed square can be shown by flashing, or by displaying its coordinate on the 7-segment display, or both (there are three rows and four columns, so to do this we only need one digit for each coordinate). However because there was no time, I was unsuccessful in completing this stage.

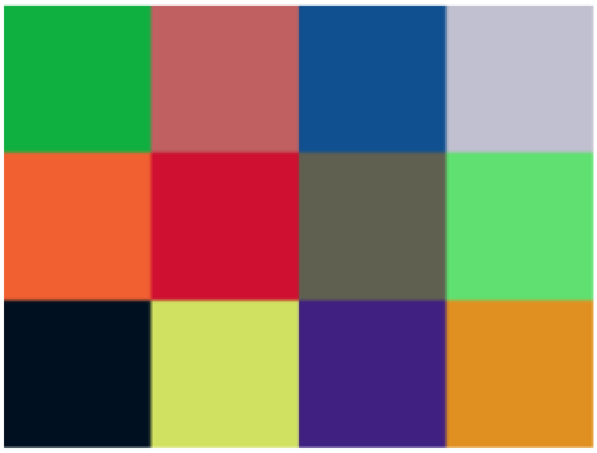


Figure 24. Test pattern for 4:3 ratio grid

Although it was incomplete, I know the way it could have been implemented. The first thing is I would have designed MEMORY which contains 12 of 12-bit Memory cells. Each 12-bit memory cell contains 12 flip flops.

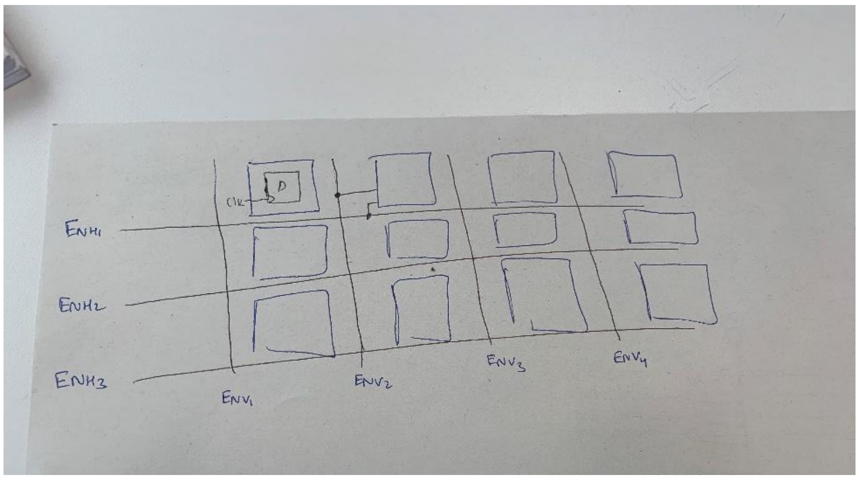


Figure 25. Sample design for stage 3

- So inside the blocks memory there are 12 flip flops connected for each of them.
- If I wanted to call the second block in row 1. It would have been the coordinates (EnH1,EnV2), Then I would have AND them to get output high.
- The En are the enablers, and these can be tested with the 7 switches, and derived to display a color.
- Also I would have used de multiplexers for the Vertical Blocks
- By following this method I would have got the grid ready.

4 Reflective Statement

Normally I set out a date to start with my assignments. When I was assigned to design a VGA port in the Basys 3 board, I had no clue about it. It was a bit tricky to understand and the concepts behind it were also difficult. Before starting of with the project, I researched in [6] about the VGA first, learning the different signal pulses and it's timings etc. Although it took me a while to figure out the first step I should proceed with for my design in Multisim. After getting help from the GLA, I understood the concept behind and moved on with my first step. Reading Reference Manuals made it a bit easier to understand the VGA System Timing and the schematics taught in our lectures gave a slight hint about the VSYNC and HSYNC pulses. How they are going to be designed and the CLk values used in generating them. So I carried on with the designing and step by step I was able to understand the uses of digital components and it's libraries in Multisim. It took me almost one a half month to complete stage 2, which resulted me in not completing stage 3. This was just the beginning of it, as it was my first time working with the VGA. It was interesting to learn when I once got my hand on it. The thing I would have changed it would probably be my time management. Which is a key criteria for these kind of projects. I must have thought about reading the [4] reference manual first and also the Components in [3] Multisim. There are many different flip/flops, multiplexers and each of them work differently. So it would have been beneficial if I did that first, learning the basics. Thinking out of the box is also a major aspect, which I should be doing often if I had to redesign the VGA port. Moreover I got to learn a lot from the laboratory design and it made me realize how the real world monitors screens display videos/colors with the VGA. I'm planning to research more about VGA as it is going to widen my knowledge. Finally my key criteria that I would be improving on is the time management, this will help me solve projects before the deadlines.

5 References

[1] Xilinx.com (2020), Published October 2019, [online] Available at: https://www.xilinx.com/support/documentation/sw_manuals/xilinx2019_2/ug835-vivado-tcl-commands.pdf [Accessed 30 January 2020].

[2] National Instrument, Edited February 2017, [online] Available at: http://zone.ni.com/reference/en-XX/help/375482B-01/multisim/addingahierarchicalblock/ [Accessed 30 January 2020]

[3] Multisim PLD datasheet, Available at: file:///M:/CE%20264/PLD%20datasheets.pdf, [Accessed 30 January 2020]

[4] Digilent, Basys 3 FPGA Board Reference Manual, Revised July 10 2019, Available at: https://reference.digilentinc.com/_media/reference/programmable-logic/basys-3/basys3_rm.pdf [Accessed 30 January 2020]

[5] ScienceDirect, Vertical Blanking interval, PDF document [online] Available at: https://www.sciencedirect.com/topics/computer-science/vertical-blanking-interval [Accessed 30 January 2020]

[6] Video Graphics Array (VGA) , PDF document [online], Available at: https://www.eecs.umich.edu/courses/eecs373/Lec/StudentF18/VGA%20Student%20Presentation.pdf [Accessed 30 January 2020]

[7] ScienceDirect, Horizontal Blanking interval, PDF document [online] Available at: https://www.sciencedirect.com/topics/computer-science/horizontal-blanking [Accessed 30 January 2020]

[8] ScienceDirect, Cathode Ray Tube, Book and PDF document [online] Available at: https://www.sciencedirect.com/topics/computer-science/cathode-ray-tube [Accessed 30 January 2020]