University of Essex - CESS

VGA Controller Design Report

CE264 Digital System Design

Yu Wenlu
Essex ID: 1909135
wy19403@essex.ac.uk

Word Count: 3803

2022-5-3

Table of Contents

Abstract

This paper aims to explore the application of PLD design in NI-Multisim 14, an excellent EDA tool that combines circuit diagram design, simulation, analysis, report generation and component modelling. This project serves as a design report for the "CE264 digital System Design-lab stage3". FPGA has long been a subject of great interest in a wide range of digital system design fields, and The Basys3 board is a complete, ready-to-use digital circuit development platform based on the latest Artix®-7 Field Programmable Gate Array (FPGA) from Xilinx® [1]. VGA is widely used as a standard display interface. This paper provides a method to design a VGA driver module by using the PLD provided by Multisim as a logical description, complete with programmable block scanning that can be done for the entire screen. This paper has been divided into four parts. The first part describes the whole project mainly in general terms, the second part details the inputs and outputs of each module and their specific roles, the third part proves the work of each module, and the fourth part contains some reflections and improvements regarding this project. All detail project document is available in author's GitHub.

1 Design Requirements

1. Outline

The purpose of this lab stage is designing a system to create a dynamic/programmable video test pattern [2]. The pattern is designed for a 4:3 aspect ratio display (A 640*480 resolution screen provided by Northwestern University Lab), and the actual colour is programmable displaying in the pattern is generating by Basys3 board 12-bit RGB switches programmable [3]. The currently access square can be selected by four pushbuttons (left, right, up and down), and shown by blinking on the screen and by displaying its coordinates on the 7-segment display in Basys3 board [2].

2. Terminology explanation

- **VGA controller**: VGA (Video Graphics Array) is widely used as a standard display interface. The common color display is generally composed of CRT (cathode ray tube), and the color of each pixel is composed of **R (red), G (green), and B (blue)** primary colors. The display is progressive scan. The **horizontal synchronization signal** and **vertical synchronization signal** generated by the VGA display module control the electron beam generated by the electron gun in the cathode ray tube to bombard the phosphor-coated screen to produce RGB trichromatic colors, which are combined into a color pixel on the display. [4]

- **Raster (Scan)**: When a raster scan monitor displays graphics, the electron beam is scanned according to a fixed scan line and a defined scan sequence. The electron beam starts from the upper left corner of the fluorescent screen, sweeps a horizontal row to the right, then quickly sweeps back to the left side of the lower position, then sweeps a second horizontal row. [5]Then sweep the second horizontal row, according to this fixed path and the order of scanning, until the last horizontal synchronization signal, that is, the completion of the entire screen scanning.
- **Synchronisation signals**: There are two signals, horizontal synchronization signal (HSYNC) and vertical synchronization signal (VSYNC), control the raster [3]. The HSYNC signal generates a low pulse of fixed width at the beginning of each row, and the data is valid at some fixed intersection of rows and columns Defines two timing intervals are -**The front porch** between the end of displayed video and the beginning of the sync pulse. -**The back porch** after the sync pulse and before displayed video. (Fig.1 Synchronisation of HSYNC signals with VSYNC signals) The VSYNC signal generates a low pulse of fixed width at the beginning of each data frame Clock.

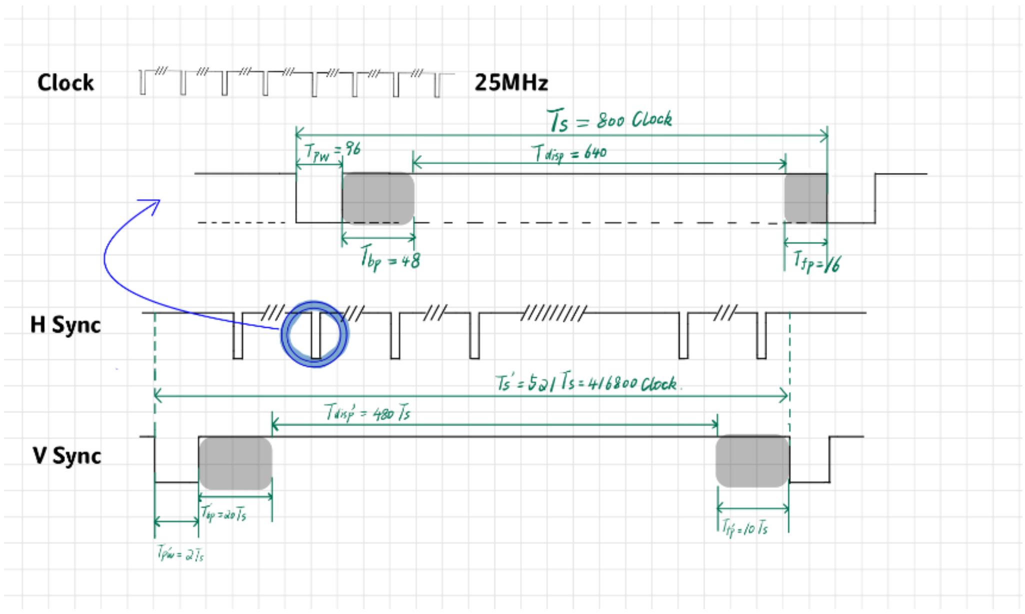


Figure 1 Synchronisation of HSYNC signals with VSYNC signals

- **Blank Signal**: Create an image by scanning the electron beam from left to right on the screen to produce a trace of a scanning line (**horizontal blank signal, Hblank**), reduce the brightness of the electron beam to zero (LOW level), then move it back to the position as low as possible on the left side of the screen as quickly as possible, restore the brightness (HIGH level), and continue scanning until all lines are displayed, and the beam is located in the lower right corner of the screen. [6]Then its intensity decreases to zero again (**vertical blank signal, Vblank**), and then quickly moves to the upper left corner to start over, creating the next frame and starting from horizontal blanking again.

3. The challenges

For this project, it was necessary to first understand the Basys3 control principles and to master the learning of PLDs. understand some basic concepts such as: cathode ray tubes, CTR displays, VGA signals. Define the waveform required for the module before designing it, and experiment with various combinations of basic components and with and without gates in order to output the desired waveform. Also dealing with human factors such as sudden computer crashes and files forgotten to be saved.

2 Design Description

1. Overview

The top module of this experiment consists of **seven** sub-modules, and their relationship is shown in Fig.2 Outline of this project. In this flow chart, the purple block represents hardware, the blue block represents the native integrated system on Basys3 board, the yellow block represents the self-designed sub-module, and the red line represents physical connection.

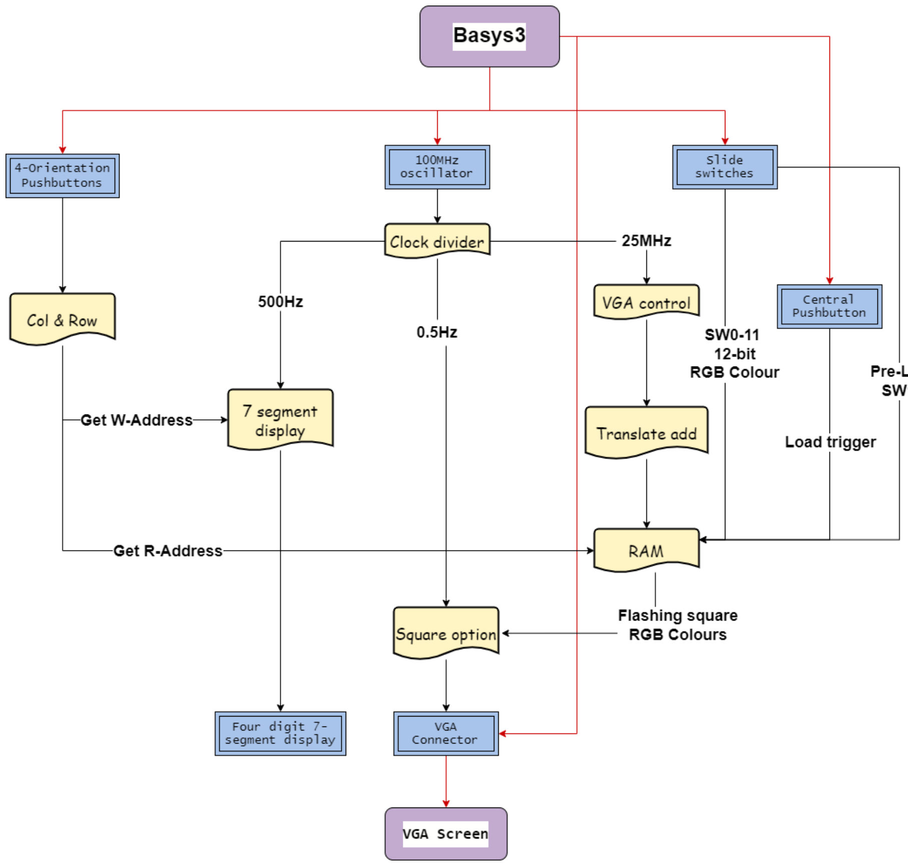


Figure 2 Outline of this project

The following Table1 brief information of the individual hierarchical blocks

Name	***Role***	***Input***	***Output***
Clock divide	Provide different clock frequencies for other sub-modules.	Basys3 chip Crystals	250MHz 500Hz 0.5Hz
7-segment display	Display currently selected square coordination	500Hz time signal Column and row information	Common cathode digital display signal of currently row and column
VGA control	Provide signals for VGA screen	250MHz time signal	HSYNC VSYNC Hblank Vblank Count state
Translate add	Translate HSYNC and VSYNC signal into square coordination	HSYNC VSYNC Hblank Vblank	square row square column
Col Row	Translate user input into square position	4 pushbuttons	row and column information of user's choice
RAM	Store colour data	Read and write coordination Hblank Vblank SW0-11 signal	12*12 colour information of current screen state.
Square option	Display colour	0.5Hz time signal RGB signals from RAM Hblank Vblank	RGB signals Highlight flashing time

Table 1 Brief information of individual hierarchical blocks

2. Stage-1 Clock Divide

According to the development documentation of Basys3, this development board contains a 100MHz crystal oscillator (clock) [1]. To simulate VGA signal, you need 25MHz clock, to drive 7-segment digital display, you need 500Hz clock, and to flash the current square, you need 0.5Hz clock. Therefore, this sub-module "Clock divider", as shown in Fig.3 Clock divide hierarchical block. The input section is the **100MHz** signal of Basys3 board, and the output has three ports, which correspond to the required **25MHz**, **500Hz** and **0.5Hz** respectively (Fig.4 Clock divide flow chart).

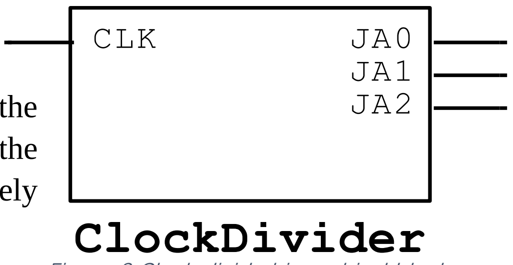


Figure 3 Clock divide hierarchical block

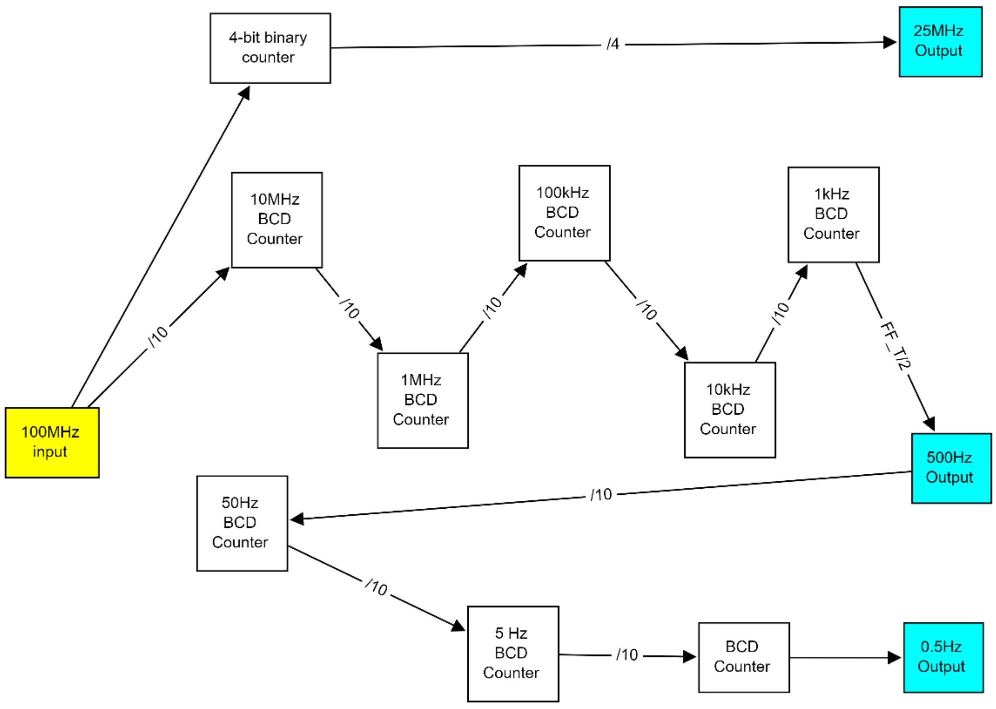


Figure 4 Clock divide flow chart

In the 'Clock divider' module, I used the **CNTR_4BIT_S** 4-bit binary counter to create a 25MHz divider, which outputs a high level when the counter goes high through four CLK signals (100MHz). For the 500Hz and 0.5Hz dividers, I used **CNTR_BCD_S**, the clock input signal of the high bit is used as the low bit feed signal, and the ENP and ENT pins of all counter chips are connected high, i.e., all counter chips are always in operation, and the counter state changes when there is a valid change in the CLK pin of the counter. Using 500Hz as an example, when the tick of the RCO of the low bit chip is changed from low to high, the high bit chip will add 1, and the counter will count from "**0000 0000 0000 0000 0000**" to "**0011 0000 1101 0100 0000**" to achieve 0~200,000 frequency division counting.

3. Stage-1 7 segment display

This sub-module, (Fig.5 Seven-segment display hierarchy block) is mainly composed of four multiplexers, two decoders and a 10-bit counter. The **500Hz clock signal** of the Clock divide is input into the counter, and the QA and QB pins of the timer are used as inputs to serve as S0 and S1 interfaces of the four multiplexers. Select one of the four digital tubes to light it up. When it is determined that one of the nixie tubes needs to be lit, the digital input through SW (a binary 4-digit number) is connected to the decoder DEC_BCD_S to generate the cathode signal, while the QA and QB of the timer generate the anode signal. Through the above operation, the numbers can be displayed on the 7-segment digital display.

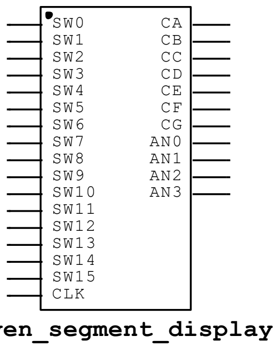


Figure 5 Seven-segment display hierarchy block

4. Stage-2 VGA Control

This sub-module receives a **25MHz** clock signal form "Clock Divider" sub-module, and outputs four signals: **HSYNC signal, VSYNC signal, Hblank signal** and **Vblank signal**. The status of the counter in this module, named **Hcount1-Hcount10 Vcount-Vcount10**, are also used as outputs to indicate the exact position of the current raster scan and to debug whether the module's output is in the ideal state (Fig. 6 VGA control hierarchical block).

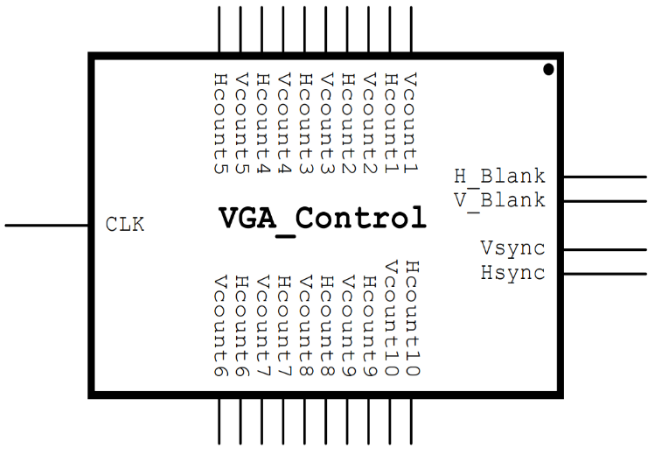


Figure 6 VGA control hierarchical block

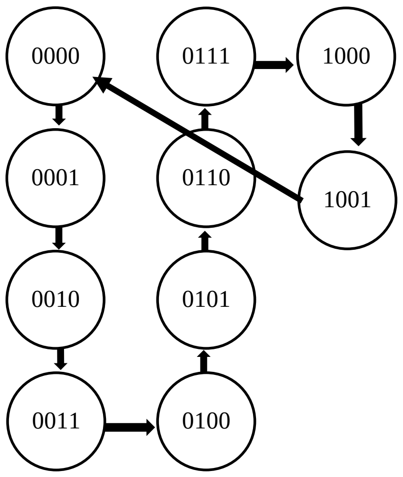


Figure 7 10_Counter flow chart

It is worth noting that in this module we need to design a 10-bit counter to keep track of whether the number of pixel clock ticks has reached the desired number. Unfortunately, Multisim PLD does not provide a 10-bit counter, so I used 3 CNTR_4BIN_S 4-decimal counters to build the hierarchical block I needed and named it **10counter**. 10counter accepts **an external clock as an input**. Every time the clock ticks, it can add one to the previous state like (Fig.7 10_Counter flow chart) and output **the current state** (Fig.8 10 counter hierarchical block). First the HSYNC signal is pulled down 96 pulses wide when the rising edge of the 25 MHz clock pixel, the data frame length required to scan a row is 800, and the VSYNC signal is scanned in a similar way to the above steps, being pulled down by 2 pulse widths at the arrival of a certain clock and then pulled up, and then delayed by 20 pulse widths after the pull-up, so that the pixels of each row are scanned. After completing this scan, the VSYNC signal is pulled down again and the process is repeated. This completes the scanning of each pixel of the 640*480 resolution display as shown in Fig.9 VGA signal scanning process. With the help of the 10counter, it is necessary to have the 10counter output the corresponding numbers separately when the above operation needs to be implemented. The Table.2 Horizontal and vertical synchronisation timings below shows the different signal passing through the pixel clock required in VGA control. It is also important to note that since the clock signal corresponding to the VSYNC signal is for the HSYNC signal, the clock input to the VSYNC signal should be the end signal of the HSYNC signal.

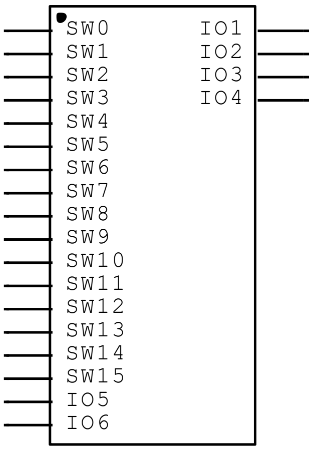


Figure 8 10_counter hierarchical block

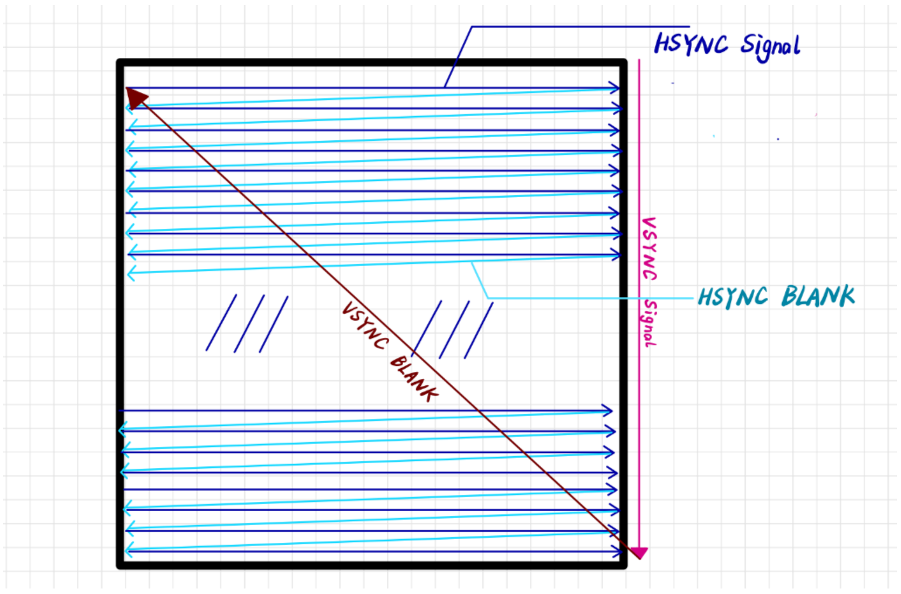


Figure 9 VGA signal scanning process

	Visible area	Front porch	Sync pulse	Back porch	Whole line
Horizontal *(μs)*	25.422	0.636	3.813	1.907	31.778
Vertical *(ms)*	15.253	0.317	0.063	1.048	16.683

Table 2 Horizontal and vertical synchronisation timings

5. Stage-3 Translate add

Input the counter states **HSYNC**, **VSYNC**, **Hblank** and **Vblank** from sub module "VGA Control", and output **square rows** and **columns** after passing through this sub-module system (Fig.10 Translate add hierarchical block)

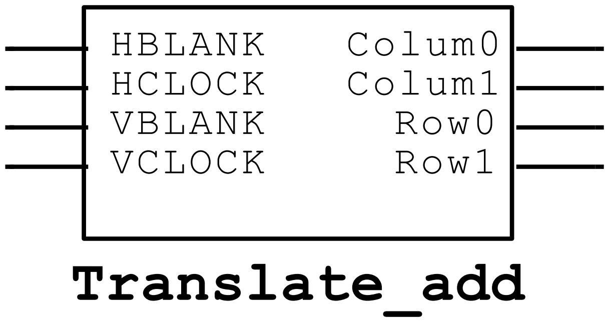


Figure 10 Translate add hierarchical block

This sub-module is internally made up of two identical 10-bit counters (detailed in the "VGA control" sub-module) and two-bit counters (implemented using a 4-bit counter), corresponding to the serial numbers of the output rows and columns respectively. For the 640*480 resolution provided by the lab the screen is displayed on the screen in 12 squares of 4*3, meaning that each square has a resolution of 160*160, so every time HSYNC/VSYNC (CLK in the case of this module) ticks the 10-bit counter is driven to increment, and when it counts up to 160 the 2-bit counter is incremented. And the status of the 2-bit counter shows the rows and columns that are currently swept by the VGA signal. When the 10-bit counter reaches 161, the 10-bit counter is to be cleared (indicating that the row and column have been counted). the Hblank and Vblank signals are used to clear the 10-bit counter and the 2-bit counter (meaning go to the next square), Fig.11 translate_add flow chart shows the above work.

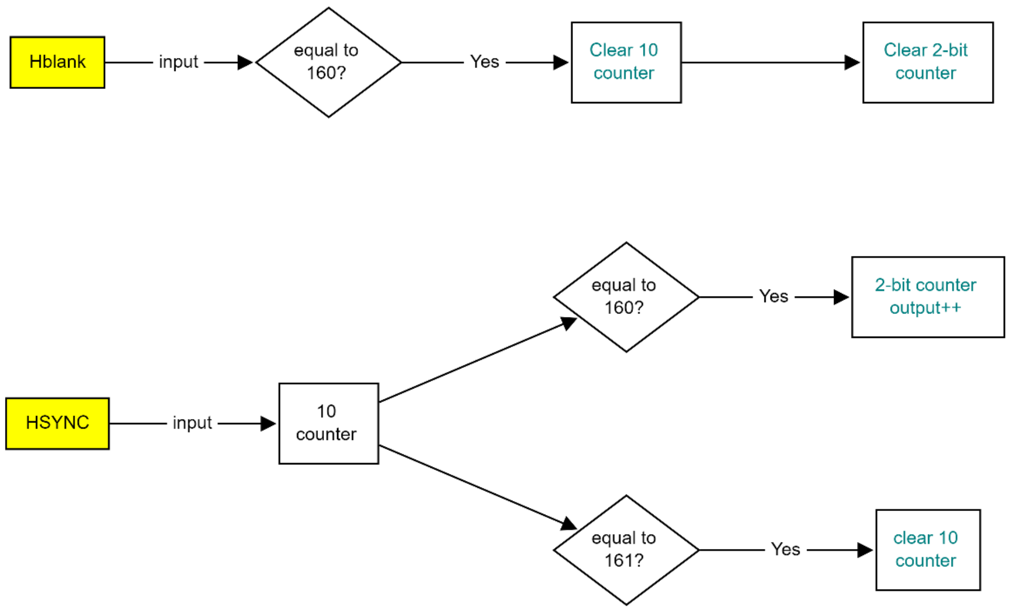


Figure 11 translate_add flow chart

6. Stage-3 Col Row

This sub-module (Fig.12 Col Row hierarchical block) **output the row and column's information** of user's choice through **4 pushbuttons (up, down, left**

and right). As we can see from the logic diagram of the module's behaviour (Fig.13 Col Row flow chart), its core control module is a counter that counts each pushbutton pressed by the user. It is worth mentioning that when the number of user presses exceeds the maximum number of rows or columns, we need to reset the current row information so that the excess is not displayed and some indescribable error is generated.

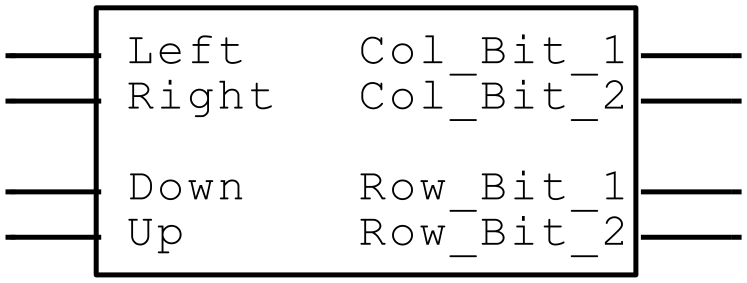


Figure 12 Col Row hierarchical block

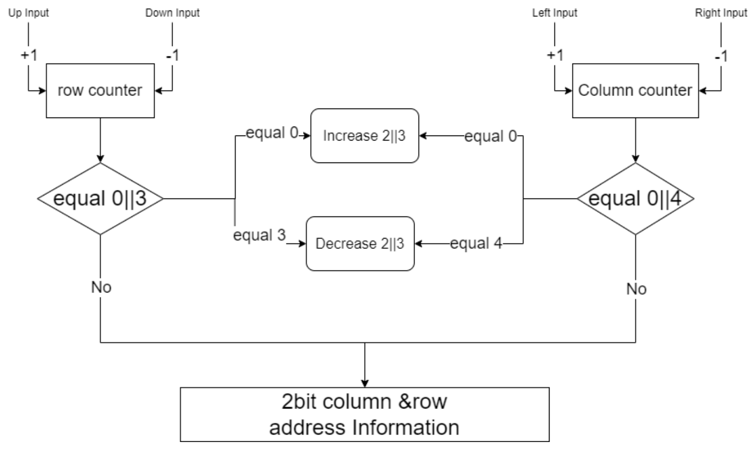


Figure 13 Col Row flow chart

7. Stage-3 RAM

RAM (read access memory) has always been one of the proudest cores of modern technology, and it has enabled the invention of a wide range of applications.
In fact, RAM is one of the most cumbersome sub-modules in the whole project, being a memory that provides direct access to bytes on switches 0-11 of the Basys3 board. (Fig.14 RAM hierarchical block) Of course, for this simple project there is no need to consider read and write speed.

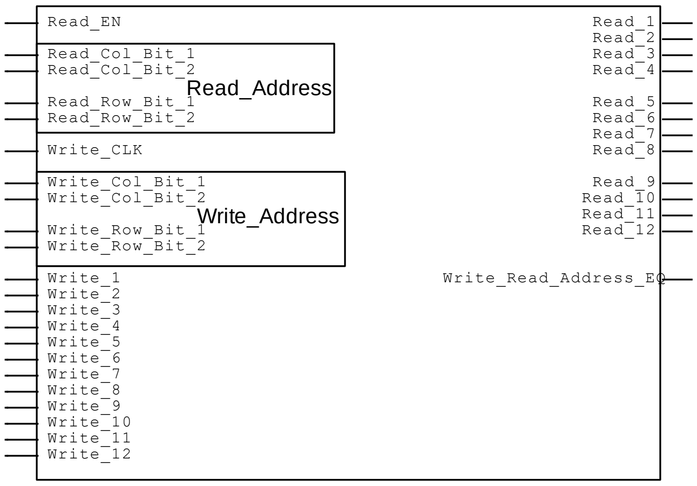


Figure 14 RAM hierarchical block

The RAM consists of the following connections

- It has separate read/write address inputs (input pins **read_c1, read_c2, read_r1, read_r2, write_r1, write_r2, write_c1, write_c2**) allowing the 12bit colour signals from **SW0-11** inputs to be written to each RAM unit.
- The read and write enable control inputs (**Read_EN, Write_EN**) are used to select the memory read/write operation.
- **Read_1-12** pins to output all colours currently stored in RAM

In this module we need a 4*3 RAM consisting of **12 RAM Units** (Fig.15 RAM Unit hierarchical block), each capable of storing 0 or 1, as well as an **address decoder** and **read/write enable**.

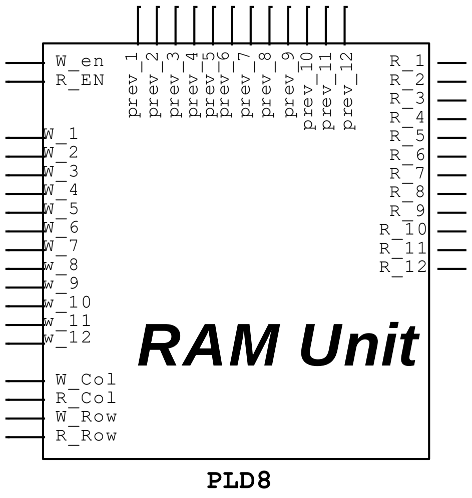


Figure 15 RAM Unit hierarchical block

When given a high level for Write_EN for RAM, the Read and write decoders for columns/ rows are received the address from Translate add, and it will point the position of RAM Unit. For the selected RAM Unit, Write_EN should be opened so that the information of SW0-11 should be stored in RAM Unit. The Unit consists of D latches, each of which stores a bit in its internal latch

When given a high level for Read_EN for RAM, The first RAM unit (1, 1) will output its own stored information to the second RAM unit (2, 1). For the second

RAM unit, it will attach its stored information to the first RAM unit and send it to the third RAM unit, and so on. After 11 RAM units, the twelfth unit will send the information from the previous 11 units with the information it has stored (4*3 12bit colours) together to the Square option as output.

8. Stage-3 Square option

After all this preparation, we've got the currently selected square via Col & Row and Translate add, and the colour information of all the pixels on the screen via RAM, so we just need to display all this information on the screen and the project is finished. This sub-module allows the input of **RGB signals** to be c**onnected directly to the VGA signal outlet** of the Basys3 board to light up the screen. (Fig.16 Square option hierarchical block)

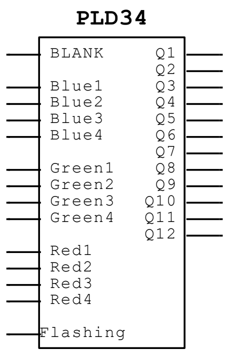


Figure 16 Square option hierarchical block

In the VGA control sub module, we mentioned that the VGA signal is not carrying any useful information when the Blank signal is at a low level, so we designed to use the Blank signal as the enable signal for the Square option, to control the work and rest of the module and avoid inverting the RGB signal during fading. It’s clock signal comes from the 0.5Hz generated by Clock divide and is used to flash the square selected by the user.

3 Performance

1. Complete demonstration

All the design requirements mentioned in labstage3 have been fulfilled. Specifically, there are four functions: 1) The whole screen is divided into 3*4 areas. 2) Different squares can be selected by means of four pushbuttons. 3) The colour of the selected square is overridden by means of 12 switches (SW0-SW12) and the central pushbutton. 4) The selected square is highlighting on the screen and its row and column are displayed on a 7-segment. (Fig.17 Full project display）

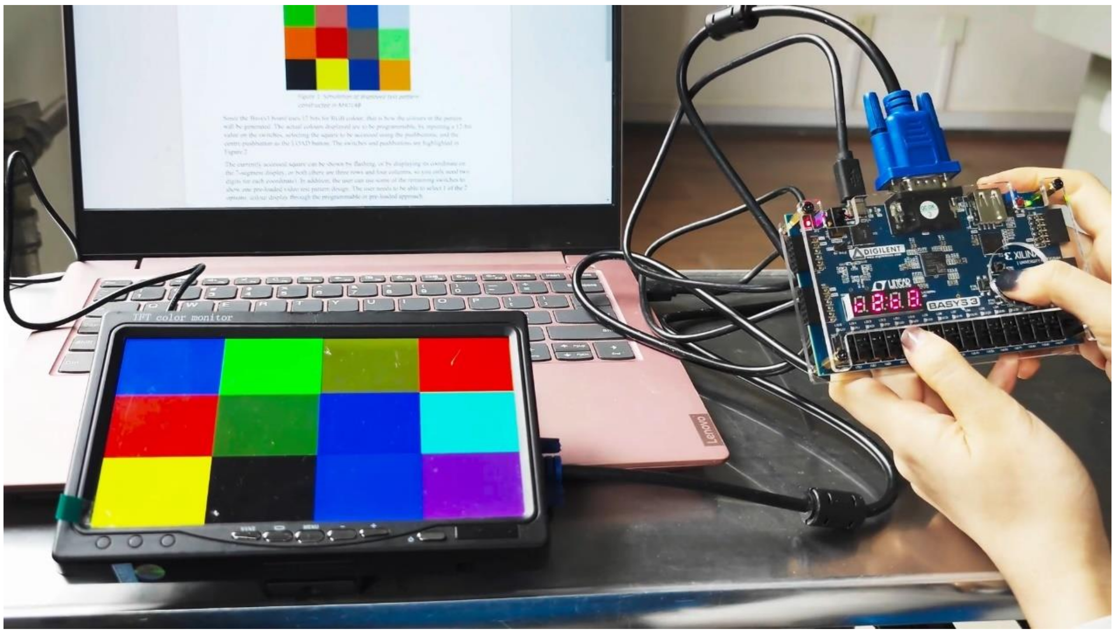


Figure 17 Full project display

The behaviour of the demonstration can be viewed online at Bilibili or Youtube.

2. Hierarchical block demonstration

VGA control hierarchical block: After connecting the circuit, verify with an oscilloscope, as shown below

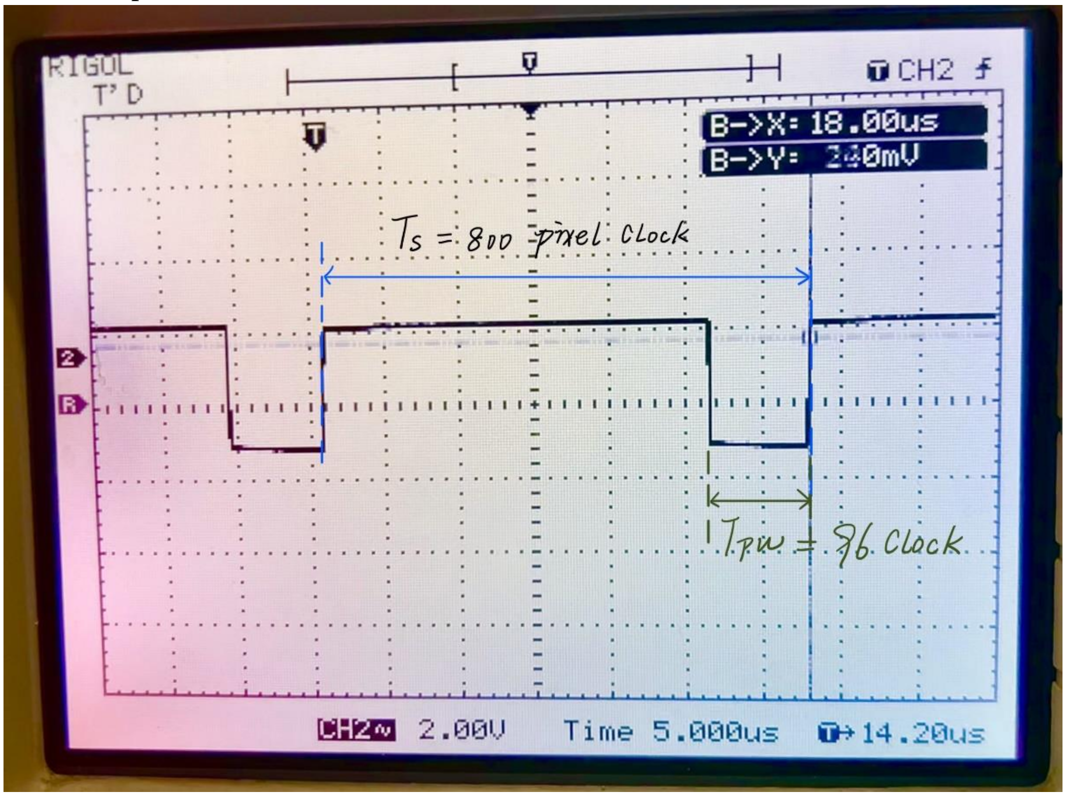


Figure 18 HBlank signal

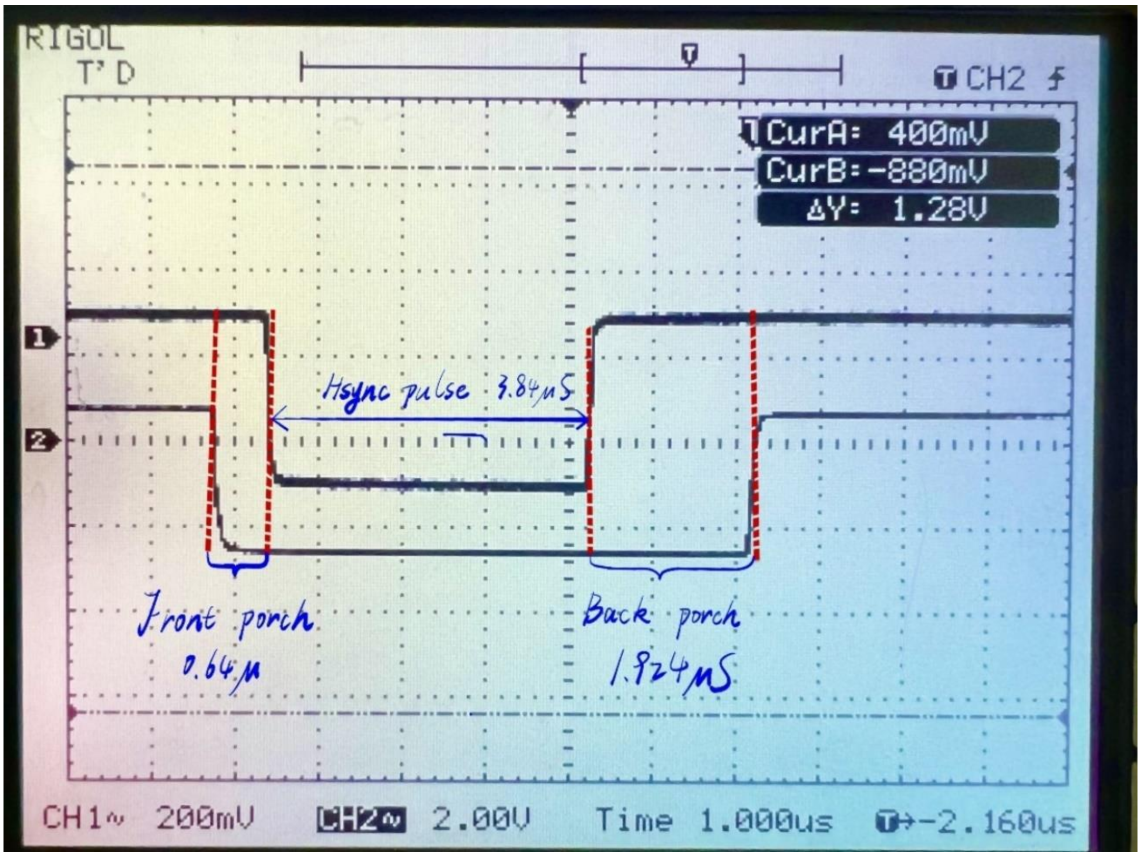


Figure 19 HSYNC and HBlank signal

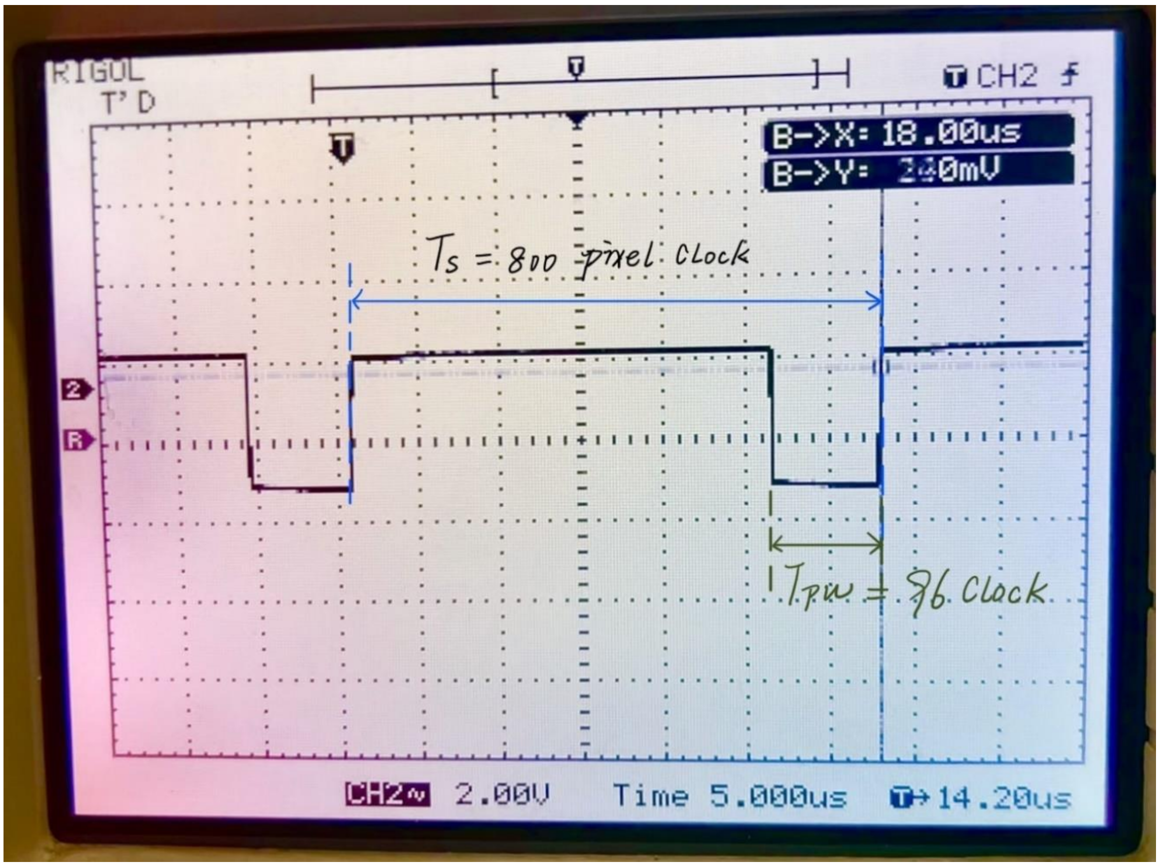


Figure 20 VBlank signal

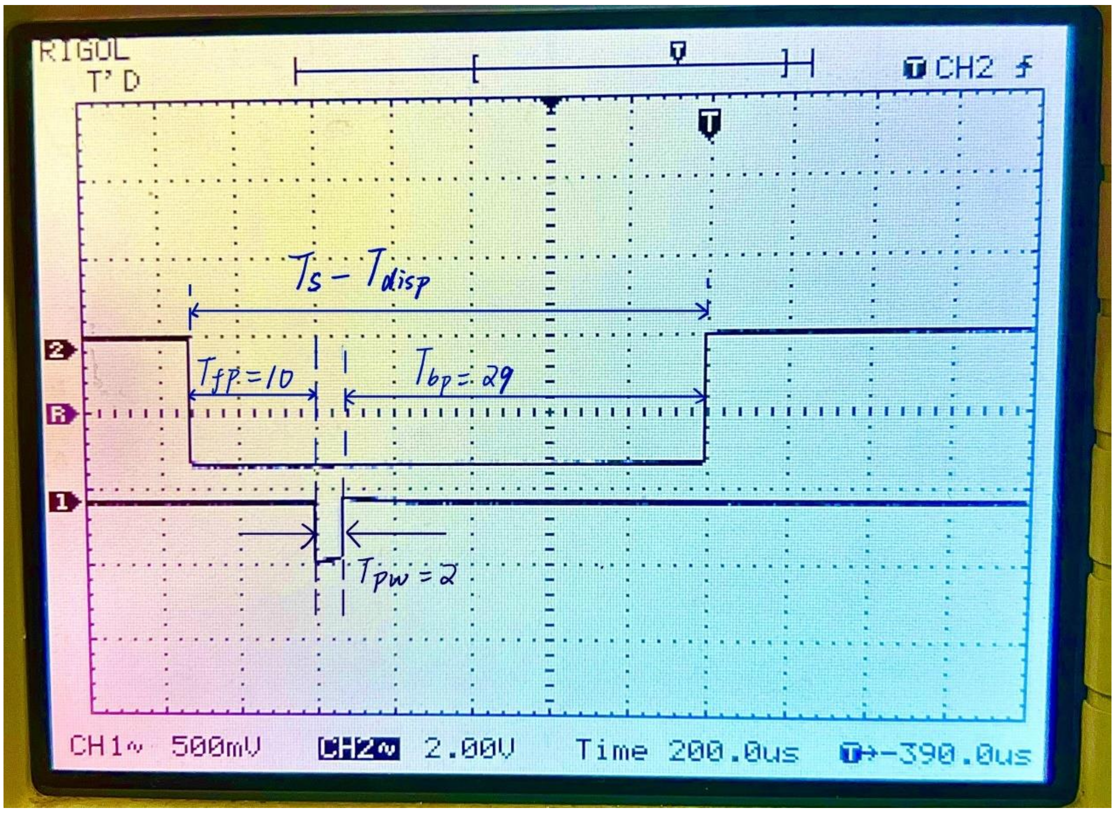


Figure 21 VSYNC and VBlank signal

The frequency readings collected via the oscilloscope are slower than the official data listed in Table 2. This may be because: the horizontal timing counter was reset at 801 instead of 800. The vertical timer counter is reset at 526 instead of 525. The reason why it is set to clear at 801 and 526 is that when set at 800, the horizontal timer will scan the next line 1 pixel clock faster than expected, causing the vertical timer to make some unpredictable errors that can result in no signal across the display.

4 Reflection and Improvement

Programmable logic devices PLDs are based on logic devices such as PALs and GALs. Compared to PALs and GALs, FPGAs/PLDs are larger in size and are suitable for applications such as timing and combinational logic circuits. It can replace dozens or even hundreds of general-purpose IC chips. The cores are programmable and the implementation is easy to change. Most importantly, I learnt to encapsulate functional modules in this project. Using different levels of reused functional blocks and debugging each layer of logic layer by layer greatly reduces debugging time and improves the systemic nature of debugging. At the same time, this design approach allowed me to break down the overall goal into smaller goals, which made my design more logical.
At the same time, I used the internet to look up various communication standards and

other people's experiences while executing the project, including looking up standards for the VGA protocol on Wikipedia and other people's register designs on GitHub, which made me more skilled at using information from the internet in the design of the project.
Debugging a small functional module every day gives me a great sense of pleasure and achievement. I am also very happy to help students when they ask me questions, not only because it gives me pleasure to help them, but also because I know that I will learn new ways of solving problems through interaction with them. I also chose to ask my teachers when I couldn't solve a problem and they always answered my questions in detail, in the process I learnt how to describe my problems more clearly when asking for help, thanks to my lab teacher.
If I were to do this project again, I would make the circuit clear at the beginning to make it more readable and make it easier to revise later and reduce repetition. I would also have documented my ideas for subdividing the functional modules in a framework diagram at the beginning to make the work more organised later on. Finally, I hope that I will save and back up my designs in time to avoid data loss in the event of equipment problems.

5 Reference

[1] Digilent, "Basys 3 reference manual," [Online]. Available: https://digilent.com/reference/programmable-logic/basys-3/reference-manual. [Accessed 13 4 2022].

[2] D. L. Hu, *CE264 Digital System Design Laboratory Design Work – Stage 3,* University of Essex, 2022.

[3] D. L. Hu, *CE264 Digital System Design Laboratory Design Work – Stage 2,* University of Essex, 2022.

[4] Z. Yaping and H. Zhanzhuang, "Design of VGA Display Module Based on FPGA," *COMPUTER TECHNOLOGY AND DEVELOPMENT,* vol. 17, no. 6, pp. 242-245, 2007.

[5] palaksinghal9903, "Raster-Scan Display," geeksforgeeks, [Online]. Available: https://www.geeksforgeeks.org/raster-scan-displays/. [Accessed 15 4 2022].

[6] Wikipedia, "Blanking (video)," [Online]. Available: https://en.wikipedia.org/wiki/Blanking_(video)#cite_note-1. [Accessed 21 4 2022].

[7] D. L. Hu, *CE264 Digital System Design Laboratory Design Work – Stage 1,* University of Essex, 2022.

6 Appendix

1. Detail Schematic

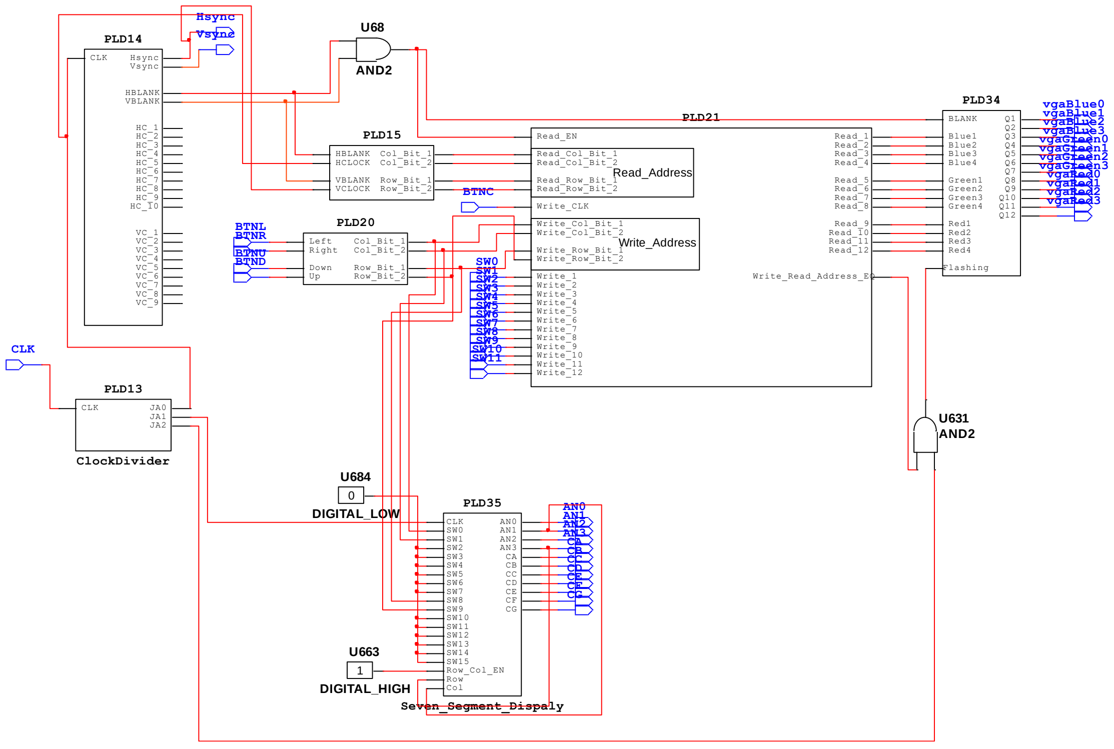


Figure 22 The Overview of Full project

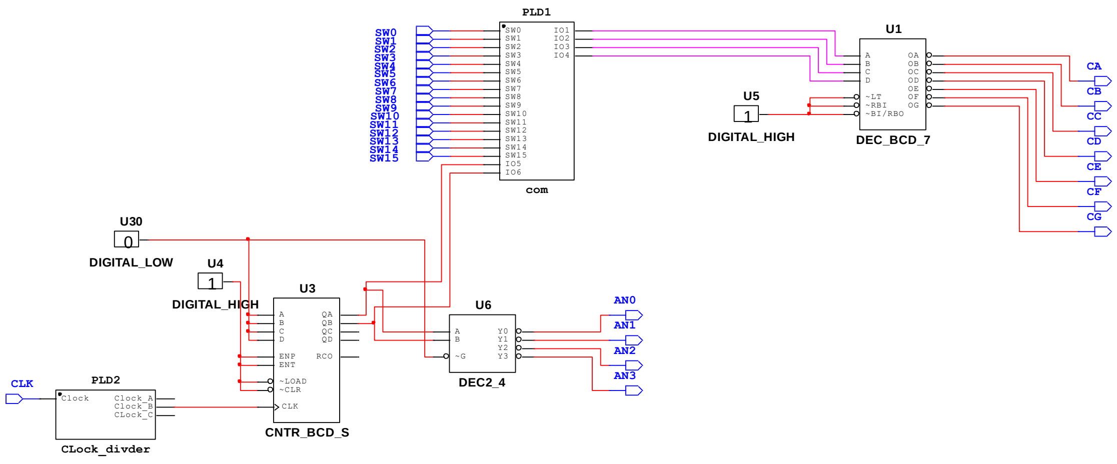


Figure 23 Stage-1 Overview

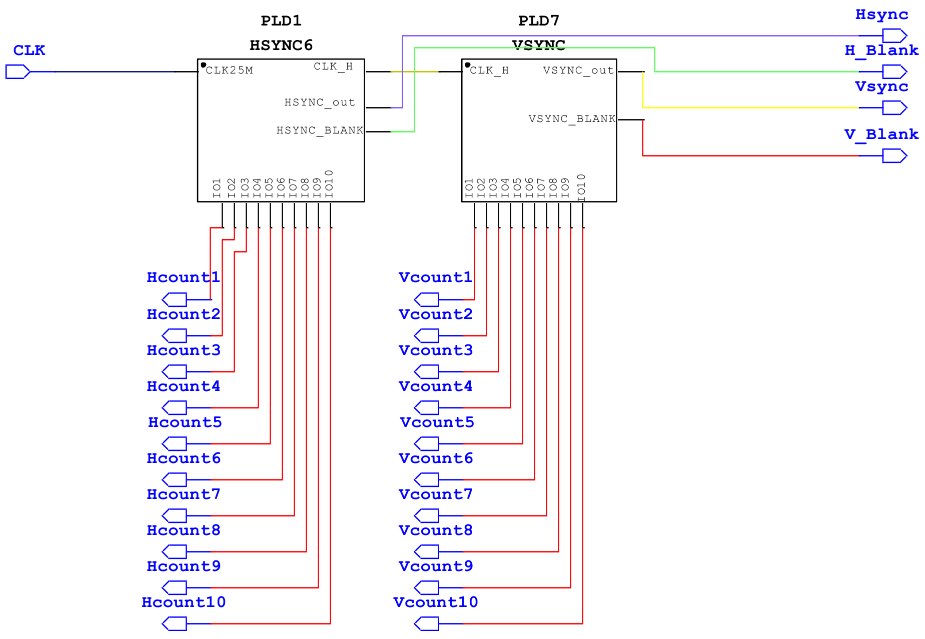


Figure 24 Stage-2 overview

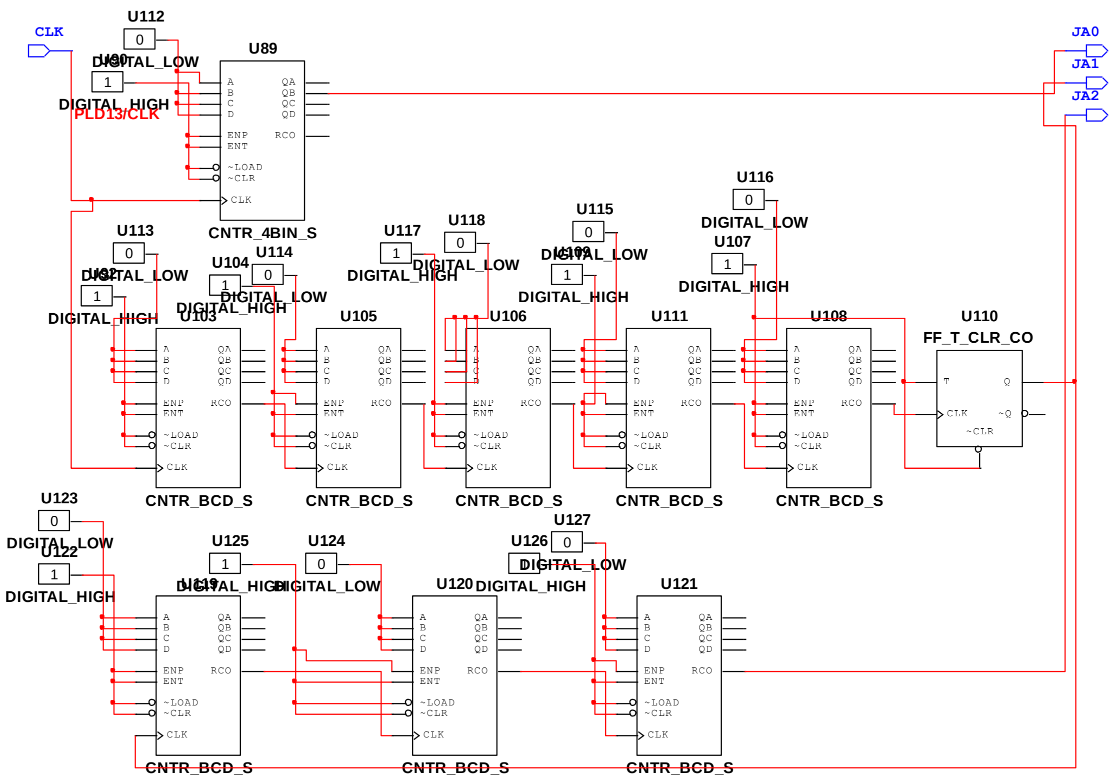


Figure 25 Clock divde hierarchical block

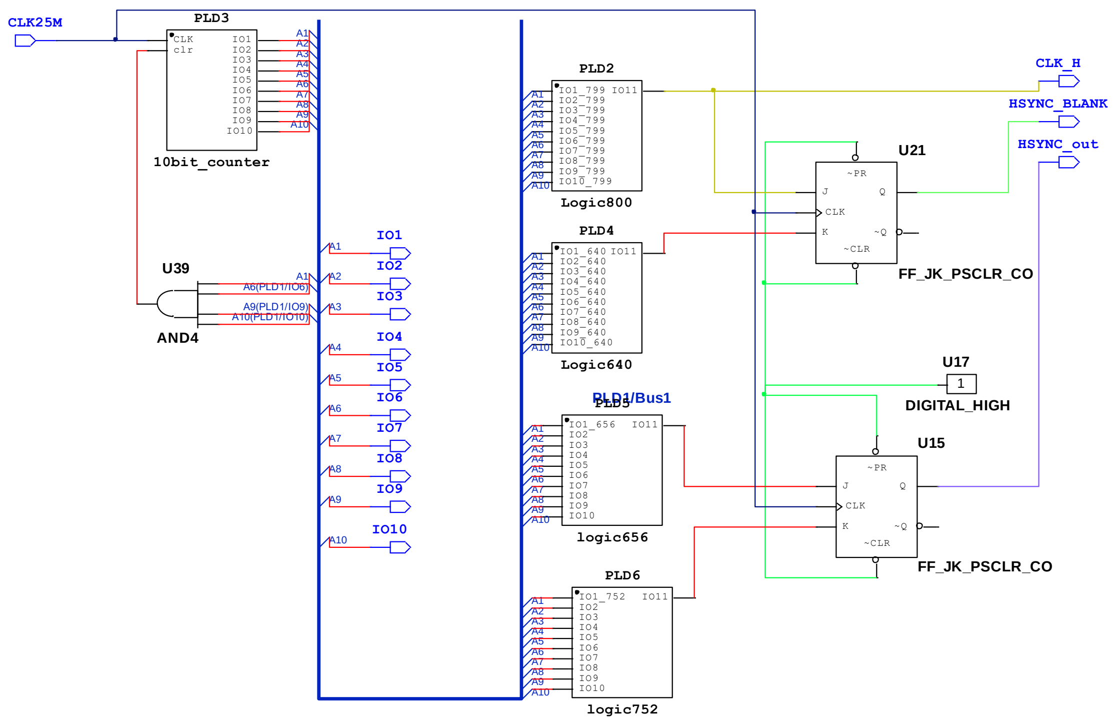


Figure 26 HSYNC hierarchical block

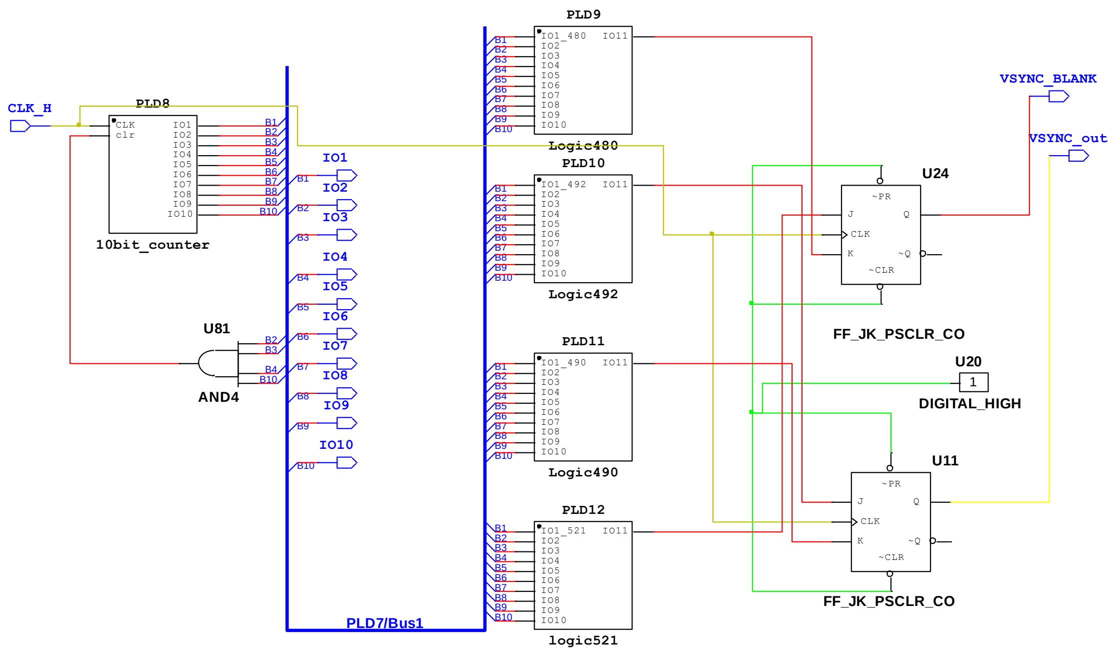


Figure 27 VSYNC hierarchical block

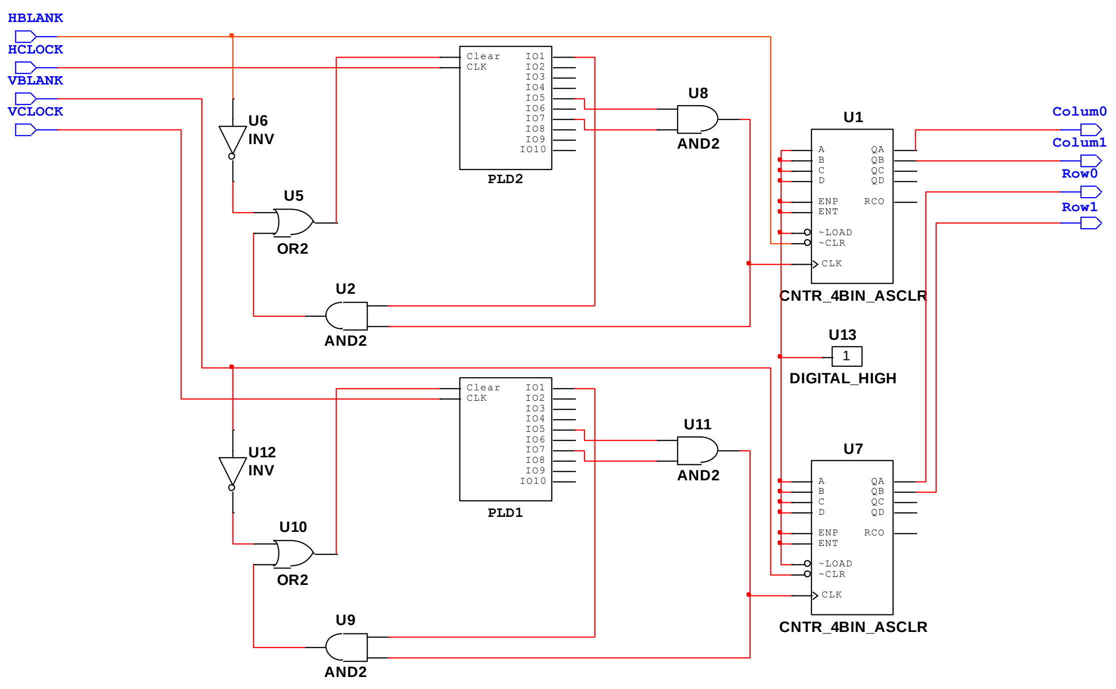
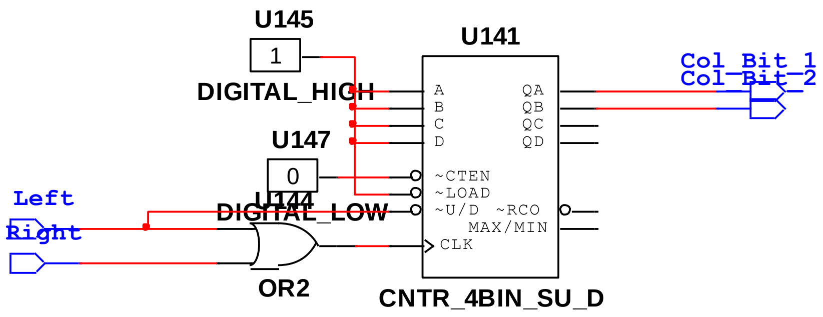


Figure 28 Translate_add hierarchical block



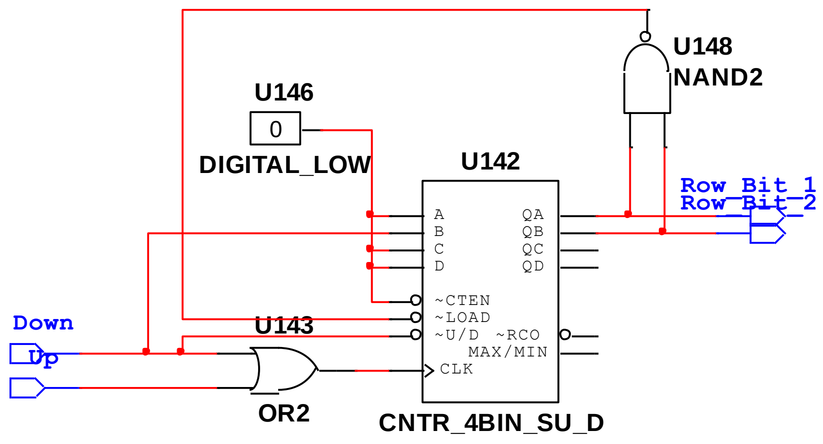


Figure 29 Col Row hierarchical block

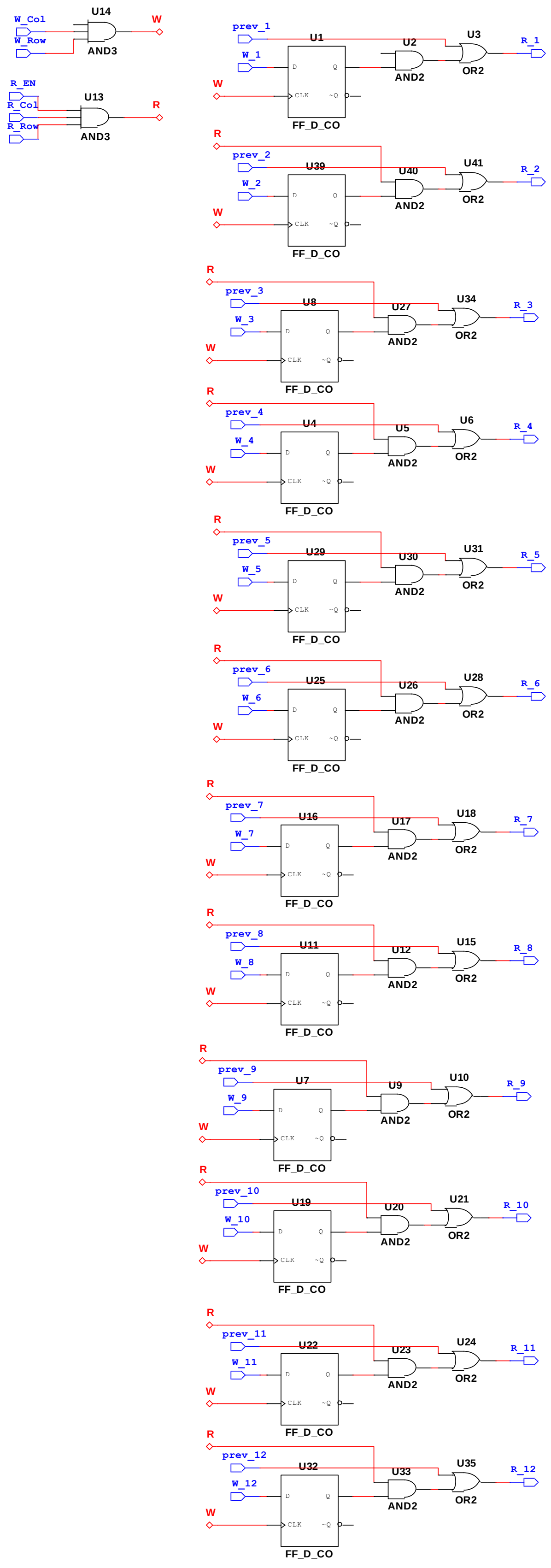


Figure 30 RAM Unit hierarchical block

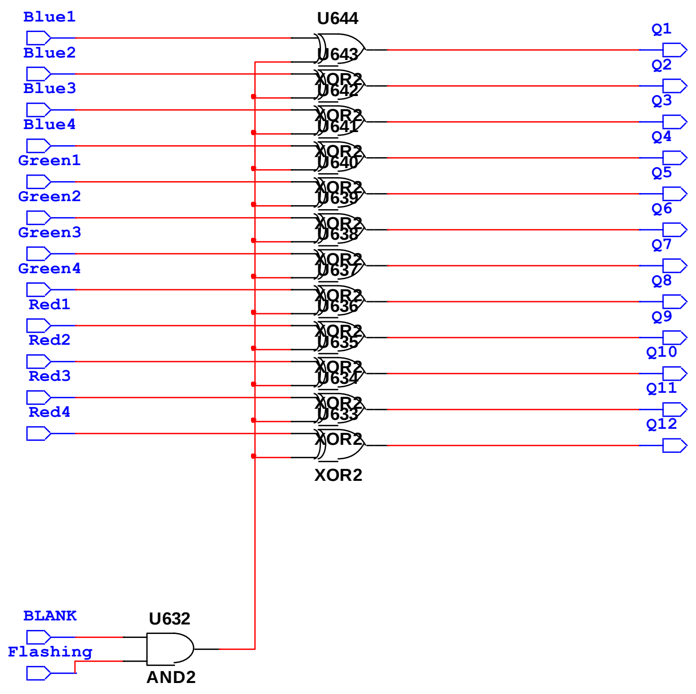


Figure 31 Square option hierarchical block

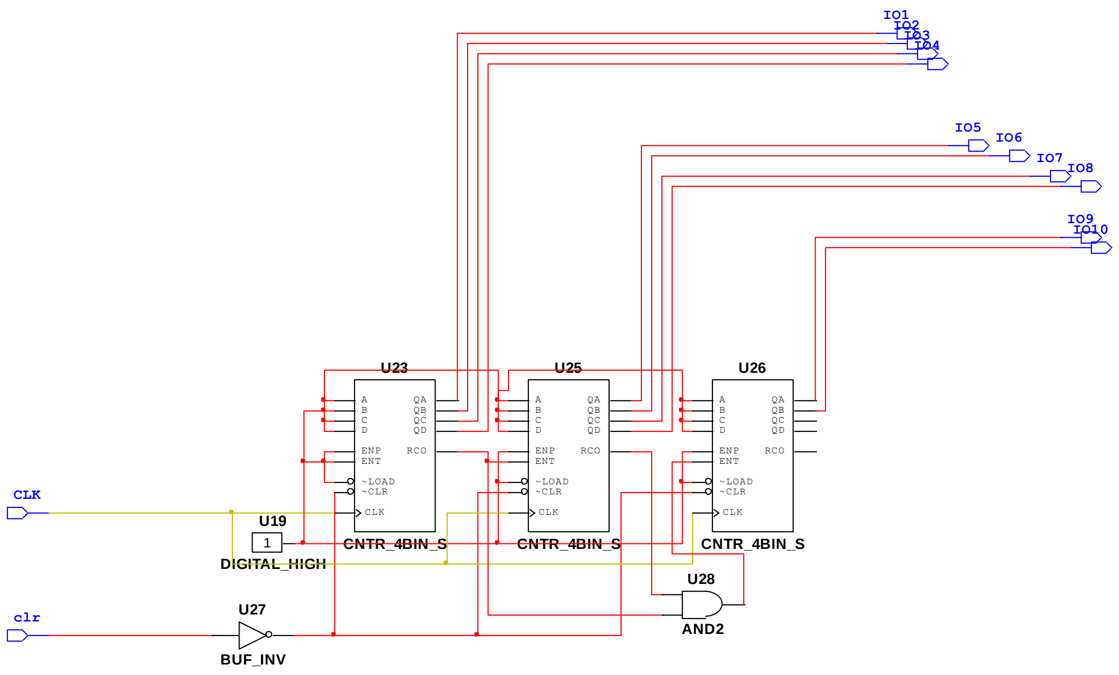
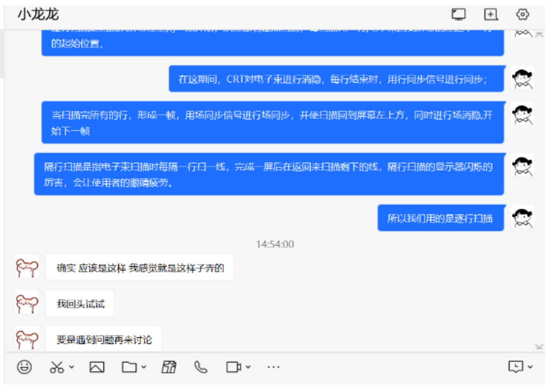


Figure 32 10 counter hierarchical block

2. Group Discussions


小龙龙

的起始位置，

在这期间，CRT对电子束进行消隐，每行结束时，用行同步信号进行同步；

当扫描完所有的行，形成一帧，用场同步信号进行场同步，并使扫描回到屏幕左上方，同时进行场消隐,开始下一帧

隔行扫描是指电子束扫描时每隔一行扫一线，完成一屏后在返回来扫描剩下的线，隔行扫描的显示器闪烁的厉害，会让使用者的眼睛疲劳。

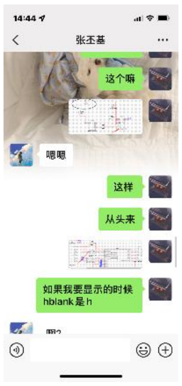
所以我们用的是逐行扫描

14:54:00

确实 应该是这样 我感觉就是这样子弄的

我回头试试

要是遇到问题再来讨论


14:44

张丕基

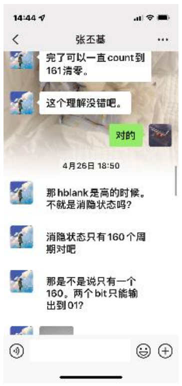
这个嘛

嗯嗯

这样

从头来

如果我要显示的时候hblank是h


14:44

张丕基

完了可以一直count到161清零。

这个理解没错吧。

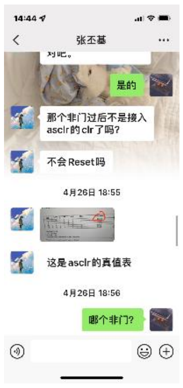
对的

4月26日 18:50

那hblank是高的时候。不就是消隐状态吗?

消隐状态只有160个周期对吧

那是不是说只有一个160。两个bit只能输出到01?


14:44

张丕基

对吧。

是的

那个非门过后不是接入asclr的clr了吗?

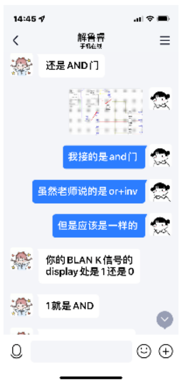
不会Reset吗

4月26日 18:55

这是asclr的真值表

4月26日 18:56

哪个非门?


14:45

解鲁青

还是AND门

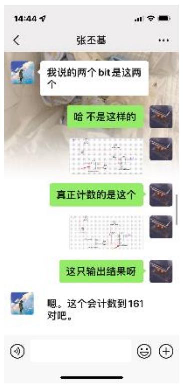
我接的是and门

虽然老师说的是or+inv

但是应该是一样的

你的BLAN K信号的display处是1还是0

1就是AND


14:44

张丕基

我说的两个bit是这两个

哈 不是这样的

真正计数的是这个

这只输出结果呀

嗯。这个会计数到161对吧。

14:44

张丕基

我们一直以为是低电位

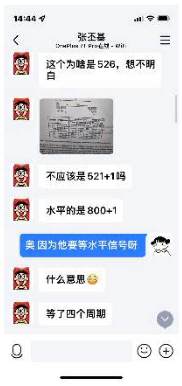
不好意思！！！我错了

等等

4月26日 19:01

我确认一下

4月26日 19:02


14:44

张丕基

这个为啥是526，想不明白

不应该是521+1吗

水平的是800+1

奥 因为他要等水平信号呀

什么意思😳

等了四个周期

14:44

张丕基

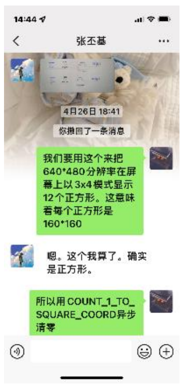
是他!

这里的hblank是高的时候。

4月26日 18:49

这里面会count对吧?

完了可以一直count到


14:44

张丕基

4月26日 18:41

你撤回了一条消息

我们要用这个来把640*480分辨率在屏幕上以3x4模式显示12个正方形。这意味着每个正方形是160*160

嗯。这个我算了。确实是正方形。

所以用COUNT_1_TO_SQUARE_COORD异步清零