Lab11: VGA

107061112 王昊文

Experiment

1 VGA displaying functions.

1.1 Inputs of the VGA controller are **clk, reset, en** and outputs of the VGA controller are hsync, vsync, vga_red[3:0], vga_green[3:0], vga_blue[3:0].

1.2 At the beginning or when **reset** (button) is pressed, the VGA display shows the image (e.g. amumu.jpg). The VGA image stay still until **en** (button) is pressed.

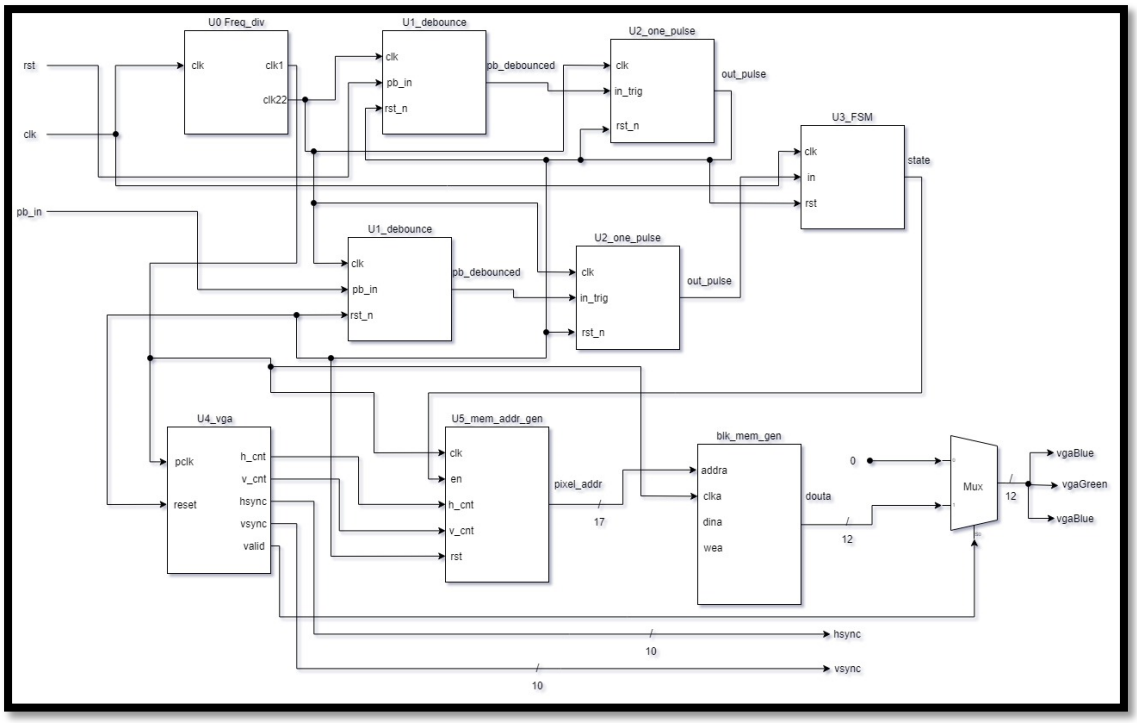
1.3 Pressing odd times **en** button to start/resume scrolling. Pressing even times **en** button to pause scrolling. Counter for **en** press is reset to zero when **reset** is pressed.

Design Specification

✓ I/O

```
Input clk; // crystal clock
Input rst; // high active reset
Output [3:0]vgaRed;      //vga red output
Output [3:0]vgaGreen;    // vga red output
Output [3:0]vgaBlue;     // vga red output
Output hsync;            // time interval for vga display
Output vsync;            // time interval for vga display
```

✓ Logic Block:



Top:

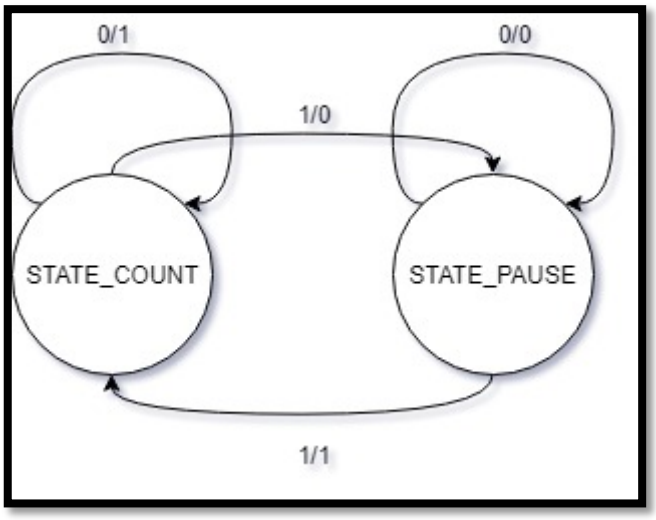
這一題的最終目標是要在螢幕上使圖片產生捲動的效果，由於有按鈕的規定，因此在前面我們必須放上一些處理按鈕訊號的模組，debounce + one pulse。有兩組的原因是需要處理reset以及捲動的enable。

後半部有一個fsm，由於是透過button來控制，因此需要一個fsm來維持state，後面會有fsm的詳細說明圖。然後後面的vga，以及mem_addr_gen皆為老師提供的模組，vga模組可以讓螢幕掃描維持應有的格式，也就是必須留有vsync，hsync等時間空格,還有遇到螢幕的邊界必須換行等等vga的格式要求。

再來是mem_addr_gen，根據使用者目前設定的圖片來產生相應的address。Vivado預設是將使用者的圖片以一維方式儲存pixel，使用者可以在這個模組中讓電腦顯示自己想要的pixel格子。Blk_mem_gen即是block ram，儲存使用者設定的圖片，透過address模組來顯示自己想要的pixel。

最後的mux可以用來生成rgb輸出，只要不是螢幕的邊界外(valid = 0)就產生block ram內的數值。

FSM:



非常簡單的fsm，基本上只是維持button state而設立的。Input為訊號in_pb debounce出來的。Output為後面mem_addr_gen開始滾動的enable。

Mem_addr_gen:

為了讓圖片滾動，我們使用在這個模組內度使用了一個counter來數position。再來因為記憶體不夠的關係，我們只能儲存320*240的圖片。為了將圖片填滿整個螢幕，我們必須將一格當作四格使用。

✓ Design Implementation

Input	
clk	*rst*
W5	T17

Output vgaRed			
vgaRed[0]	*vgaRed[1]*	*vgaRed[2]*	*vgaRed[3]*
G19	H19	J19	N19

Output vgaGreen			
vgaGreen[0]	*vgaGreen[1]*	*vgaGreen[2]*	*vgaGreen[3]*
J17	H17	G17	D17

Output vgaBlue			
vgaBlue[0]	*vgaBlue[1]*	*vgaBlue[2]*	*vgaBlue[3]*
N18	L18	K18	J18

✓ Discussion

這一題雖然很簡單，但做完真的超級有成就感的！第一題雖然簡單但有很重要的觀念，必須很清楚的瞭解圖片是如何被顯示在螢幕上的原理，對於後面

的題目很有幫助。雖然簡單可是花了我很多時間瞭解block ram儲存圖片的方法。我一直看不懂為什麼圖片的位置要除以二，後來才發現原來是fpga內部的儲存空間不足，然後又享要顯示權螢幕的圖片造成的。總之第一題雖然簡單但是做好vga的重要關鍵。瞭解了內部後面題目應該不會太難。

Experiment

2 Calculator display.

2.1 Combine the key board controller and VGA displaying controller to design a calculator with 2-digit addition/subtraction/multiplication. The display function should be the same as usual calculator or APP in the smartphone.

Design Specification

✓ I/O

Input clk; // crystal clock

Input rst; // high active reset

Inout PS2_CLK; // for keyboard

Inout PS2_DATA; // for keyboard

Output [3:0]vgaRed; // vga red output

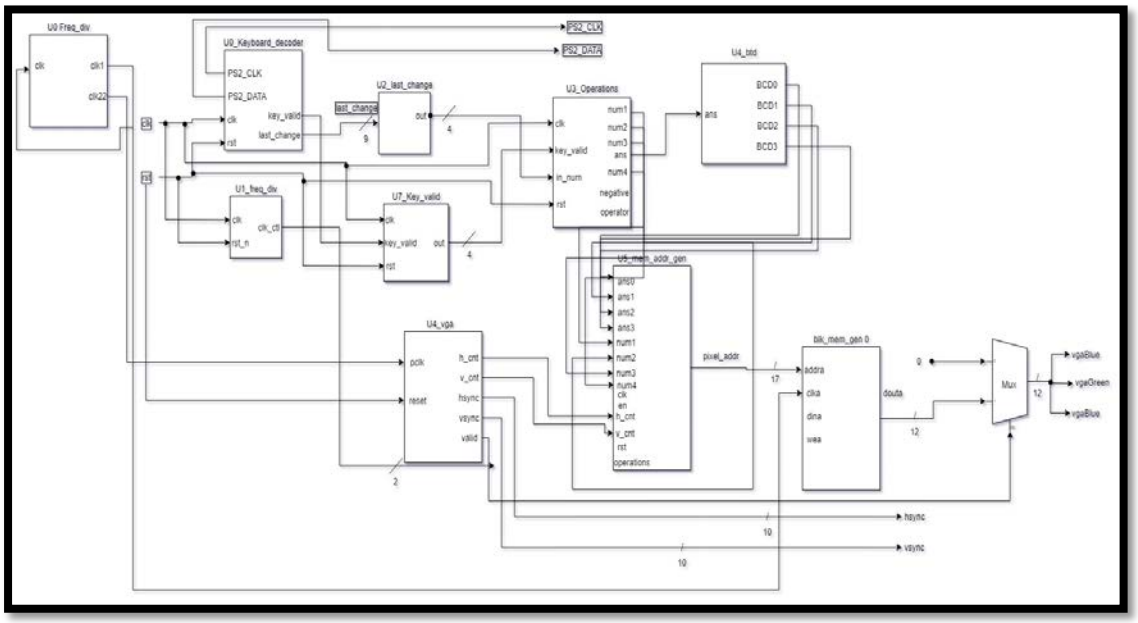
Output [3:0]vgaGreen; // vga red output

Output [3:0]vgaBlue; // vga red output

Output hsync; // time interval for vga display

Output vsync; // time interval for vga display

✓ Logic Block:



這一題計算機方面的功能來自lab09，包括前面關於鍵盤的設定以及operations的設定都是。我僅是把operations中產生的數字輸出出來到mem_addr_gen裡面。這一題的模組之中，vga模組一樣是用來控制螢幕掃描的部分，不能夠掃出邊界還有vsync hsync的輸出。

關於這一題我計算機的設定是將螢幕橫向分成十格，五格使用者輸入，包括兩個two digit number還有算數符號（加減乘除），在機上等號還有4 digit answer。每一個圖片皆為64*64，橫向剛好可以填滿整個螢幕。那我是將所有數字還有符號連起來，horizontally，如下圖

重頭戲在mem_addr_gen裡面，我們將前面operation模組得到使用者目前的輸出和答案，然後在mem_addr_gen裡面根據使用者的輸入，輸出圖片的位子即可。

✓ Design Implementation

Input	
clk	*rst*
W5	R2

Inout	
PS2_CLK	*PS2_DATA*
C17	B17

Output vgaRed			
vgaRed[0]	*vgaRed[1]*	*vgaRed[2]*	*vgaRed[3]*
G19	H19	J19	N19

Output vgaGreen			
vgaGreen[0]	*vgaGreen[1]*	*vgaGreen[2]*	*vgaGreen[3]*
J17	H17	G17	D17

Output vgaBlue			
vgaBlue[0]	*vgaBlue[1]*	*vgaBlue[2]*	*vgaBlue[3]*
N18	L18	K18	J18

✓ Discussion

如果瞭解了上面的圖片之後這一題應該不會太難。這一題最麻煩的地方大概就是剪圖片，還有控制使用者輸入數字之後必須讓掃描的部分移動到正確的位子，如果設定錯誤圖片顯示不出來其實很難debug。其實剛開始我用了多ram

的作法，每一個數字存一個ram。後來合電路的時候一直合不出來，卡了好久，原來是我block ram設定的時候完全按照老師的作法，並沒有更改每一個ram的大小，總大小超過了fpga的容量，導致合不出來。後來改完之後終於成功。但是我還想挑戰一下用一個ram來做，因此這一題我做了兩個版本，只用一個ram來做，bitstream的速度明顯快很多。

Experiment

3 TETRIS element generator

3.1 Generate basic elements of TETRIC (as follows) randomly in the VGA monitor, and plot each of them in the center of the first row of the display, which is a 10 x 20 (WxH) square 2D playing space.

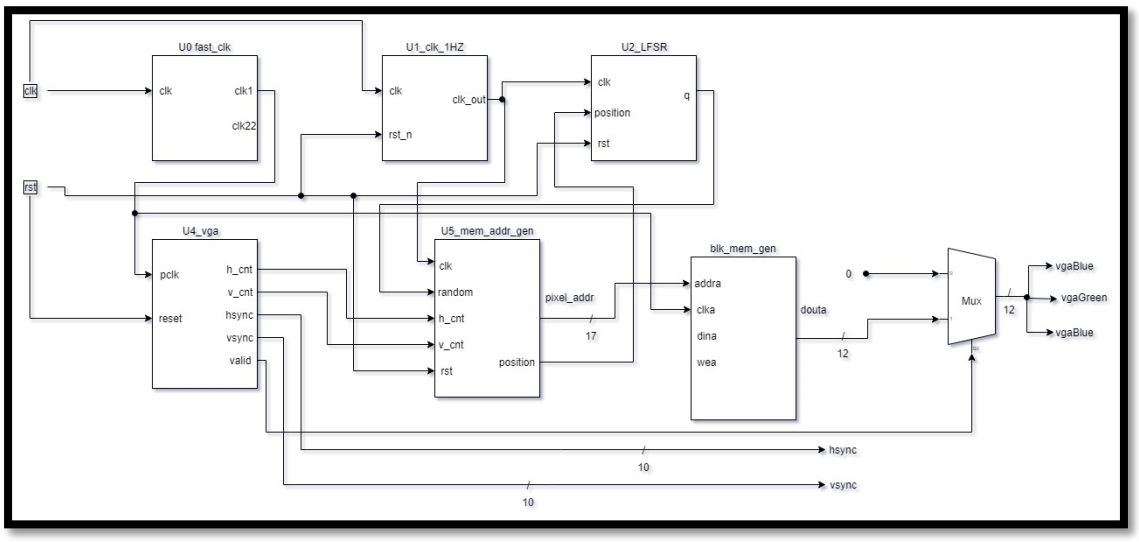
3.2 Each generated basic element moves down by the step of a square at the speed of 1Hz. Finally, they disappear below the playing space. When a basic element disappears, a new basic element is generated again and fall down again repeatedly.

Design Specification

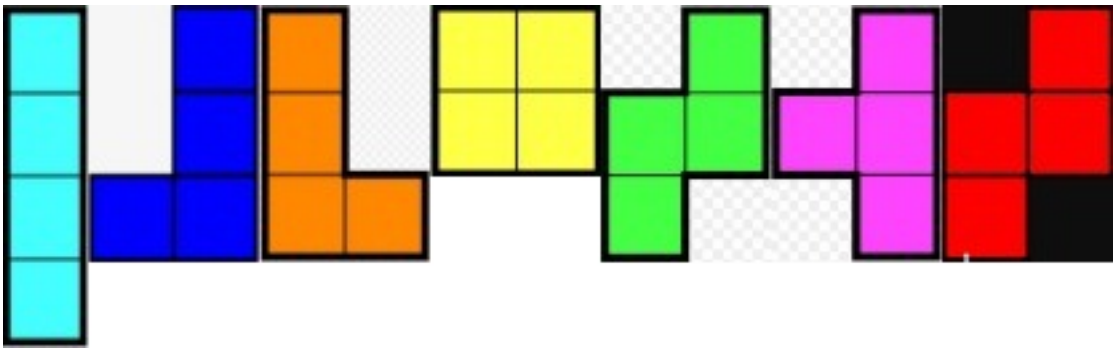
✓ I/O

```
Input clk; // crystal clock
Input rst; // high active reset
Output [3:0]vgaRed;       //vga red output
Output [3:0]vgaGreen;     // vga red output
Output [3:0]vgaBlue;      // vga red output
Output hsync;             // time interval for vga display
Output vsync;             // time interval for vga display
```

✓ Logic Block:

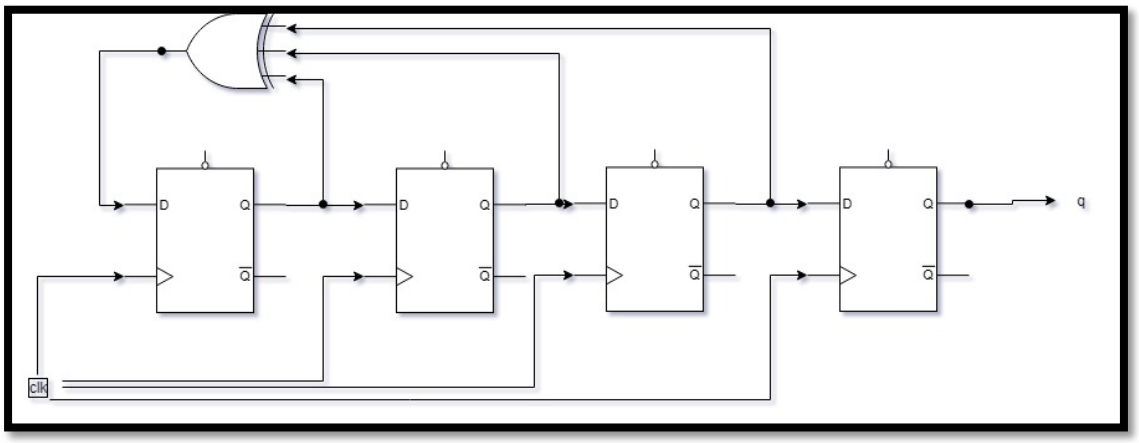


這一題基本上是使用第一題的框架，只是去掉button部分的處理。其中必須有產生隨機tetris block的功能，因此我必須使用一個LFSR的模組，可以隨機產生一個數字，然後對於每個數字我們給予他產生的方塊，比如說生成一的時候生成t-block，二的時候產生I-block，諸如此類。後面會有模組的說明。再來就是要讓他動的部分。我另每個block的一格方塊格為32*32，排成以下圖形：



只要掃描的時候每一秒加上在顯示的縱座標加上32就可以達到方塊下降的效果。其他部分關於顯示圖片以及rgb的部分都與上一題一樣。

LFSR:



利用以上回路的方式就可以生成random number。

✓ Design Implementation

Input	
clk	*rst*
W5	T17

Output vgaRed			
vgaRed[0]	*vgaRed[1]*	*vgaRed[2]*	*vgaRed[3]*
G19	H19	J19	N19

Output vgaGreen			
vgaGreen[0]	*vgaGreen[1]*	*vgaGreen[2]*	*vgaGreen[3]*
J17	H17	G17	D17

Output vgaBlue			
vgaBlue[0]	*vgaBlue[1]*	*vgaBlue[2]*	*vgaBlue[3]*
N18	L18	K18	J18

✓ Discussion

這一題剛開始我真的超級猶豫到底要不要做，後來在好奇心的驅使下我還是挑戰了一下，雖然第三小題沒有做出來。其實關於position的部分我還是稍微想了一下，不過後來發現大概就是第一小題的變形，只是clock改慢，然後一次調多一點，就可以產生下降的效果。這題讓我對於final project有更深入的思考，應用的面向可以更多。

✓ Conclusion

總算結束了這個學期最後一個lab，可以說是收穫滿滿，讓我一掃上學期對於邏輯設計糟糕的印象。透過上學期學到的東西做出實際功能的東西感覺真的不一樣。最讓我驚艷的就是最後的這個vga，其實瞭解內部運作原理之後就不難，唯一美中不足的大概就fpga實在是容量太小，可以塞的東西真的好少阿。期許final project可以做出我自己滿意的東西。