Lab 11: VGA

106010006 黃詩瑜

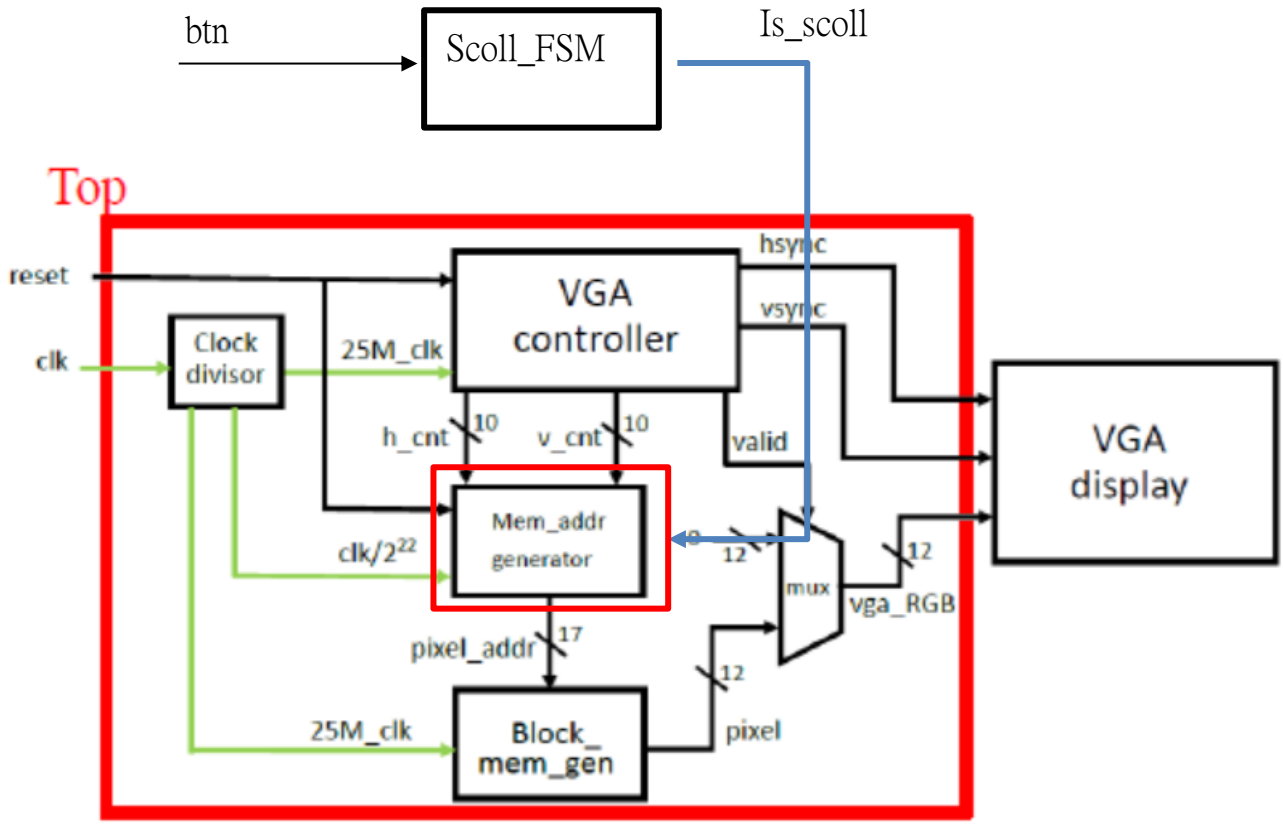
1. VGA displaying functions.
1.1 Inputs of the VGA controller are clk, reset, en and outputs of the VGA controller are hsync, vsync, vga_red[3:0], vga_green[3:0], vga_blue[3:0].
1.2 At the beginning or when reset (button) is pressed, the VGA display shows the image (e.g. amumu.jpg). The VGA image stay 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.

Specification:

Input: clk, rst, btn
Output: vgaRed[3:0], vgaGreen[3:0], vgaBlue[3:0], hsync, vsync

I/O port:

W5 [get_ports clk] U17 [get_ports rst] U18 [get_ports btn] G19 [get_ports {vgaRed[0]}] H19 [get_ports {vgaRed[1]}] J19 [get_ports {vgaRed[2]}] N19 [get_ports {vgaRed[3]}] N18 [get_ports {vgaBlue[0]}] L18 [get_ports {vgaBlue[1]}] K18 [get_ports {vgaBlue[2]}] J18 [get_ports {vgaBlue[3]}] J17 [get_ports {vgaGreen[0]}] H17 [get_ports {vgaGreen[1]}] G17 [get_ports {vgaGreen[2]}] D17 [get_ports {vgaGreen[3]}] P19 [get_ports hsync] R19 [get_ports vsync]



Implementation:

老師給的 demo 檔就是會讓圖片一直滾動，所以要改的是用紅色圈起來的地方，再加上一個控制按鈕的 FSM 輸出 is_scoll 判斷有沒有要捲動，再傳給 mem_addr_gen 做處理，產生不同的 addr 讓記憶體輸出。

FSM 裡，由按鈕控制，每按一下後會將 is_scoll 互換，利用 filp_flop 記住前一 clk 的值並根據這個值改變 is_scoll。

mem_addr_gen 裡，Position 是一個 0 到 239 的 counter 再根據 is_scoll 決定要不要加 1。

```
assign pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1)+ position*320 )% 76800;  //640*480 --> 320*240

   always @ (posedge clk or posedge rst) begin
      if(rst)
          position <= 0;
      else if(~is_scoll)
           position <= position;
      else if(position < 239)
           position <= position + 1;
       else
           position <= 0;
   end
```

討論：

其實我從第一題就卡住很久了，因為看不懂老師 demo 檔裡面 pixel_addr 那裡是怎麼設計的，有問一個助教才稍微了解，雖然跟這題需要更改的地方沒有關係，但是我還是想先搞懂再設計。

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.

Specification:

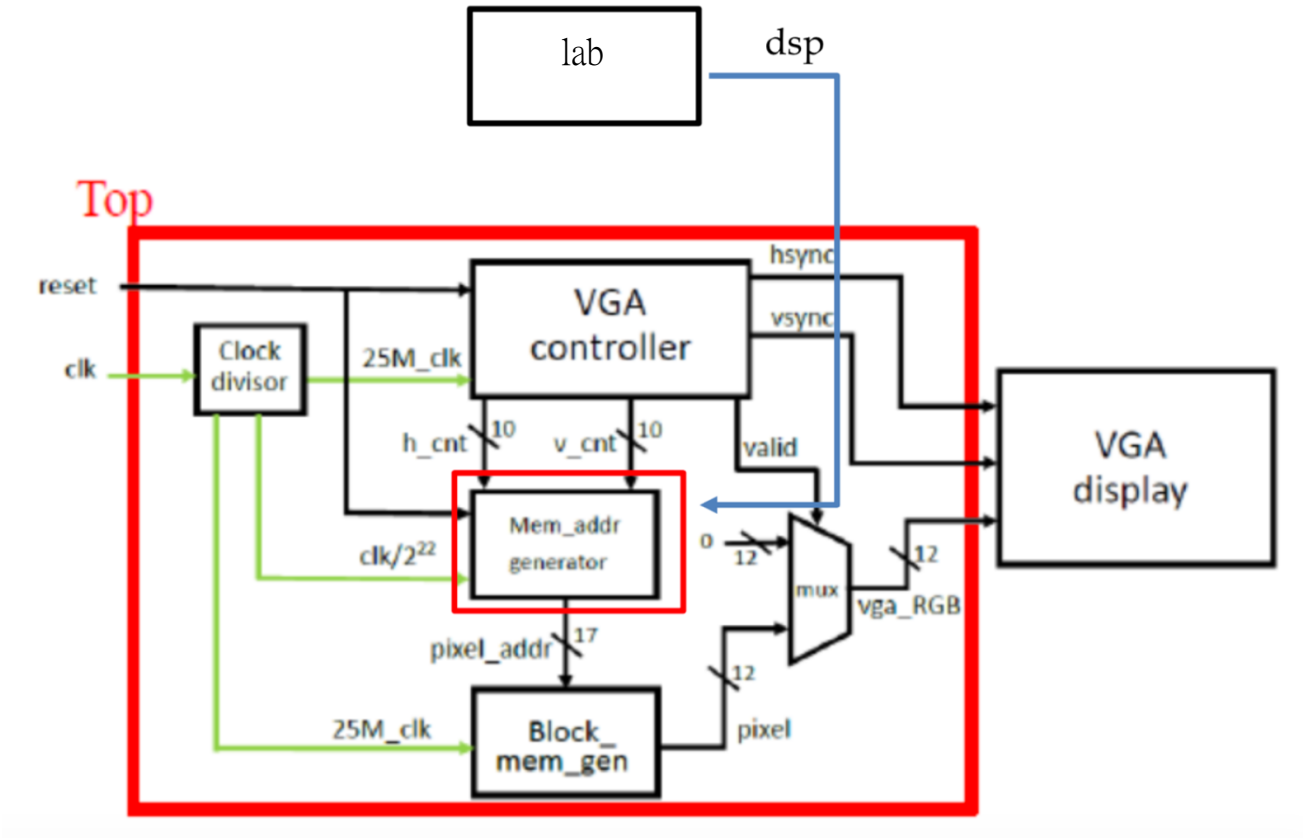
Input: clk, rst_n

Inout: PS2_DATA, PS2_CLK

Output: vgaRed[3:0], vgaGreen[3:0], vgaBlue[3:0], hsync, vsync, bit_dsp[3:0], BCD_dsp[7:0]

I/O port:

W5 [get_ports clk] U17 [get_ports rst] G19 [get_ports {vgaRed[0]}] H19 [get_ports {vgaRed[1]}] J19 [get_ports {vgaRed[2]}] N19 [get_ports {vgaRed[3]}] N18 [get_ports {vgaBlue[0]}] L18 [get_ports {vgaBlue[1]}] K18 [get_ports {vgaBlue[2]}] J18 [get_ports {vgaBlue[3]}] J17 [get_ports {vgaGreen[0]}] H17 [get_ports {vgaGreen[1]}] G17 [get_ports {vgaGreen[2]}] D17 [get_ports {vgaGreen[3]}] P19 [get_ports hsync] R19 [get_ports vsync] W7 [get_ports {BCD_dsp[7]}] W6 [get_ports {BCD_dsp[6]}] U8 [get_ports {BCD_dsp[5]}] V8 [get_ports {BCD_dsp[4]}] U5 [get_ports {BCD_dsp[3]}] V5 [get_ports {BCD_dsp[2]}] U7 [get_ports {BCD_dsp[1]}] V7 [get_ports {BCD_dsp[0]}] W4 [get_ports {bit_dsp[3]}] V4 [get_ports {bit_dsp[2]}] U4 [get_ports {bit_dsp[1]}] U2 [get_ports {bit_dsp[0]}]



Implementation:

使用之前 lab 的 code，就是一個有加減乘功能的兩位數計算機，產生的 dsp 再加減乘的時候，最外面那一位數會顯示符號，前兩位顯示數字，按下 enter 後顯示答案，再把他接到 mem_addr_gen 做處理。我的 dsp 0~9 是一般數字，但是我是把+-*分別設成 10,11,12 好讓他去顯示。

加入 7-seg LED 是想把答案顯示在 SSD 上面看有沒有正常運作。

這是 word 打字再截圖下來的圖片，預設每一個數字的寬是 45pixel 高是 90，因此再轉檔成是要輸入的長/寬就是 45*13 / 90，並依照講義上的步驟把它加進 去 project 裡。

mem_addr_gen 裡，當 h_cnt 在 0 到 45，v_cnt 是 0 到 90 時，代表是要輸出最左上角那塊區域，要顯示 dsp3，如果是要顯示 3 的那個數字，就代表說要從記憶體 3*45 開始讀取，而高要乘以 585 來換行，加 h_cnt % 45 是讓它的 address 也要跟著 h_cnt 跑，其他的概念也是相

同的。如果都不在的話，就把 address 都指定成 0 因為第一個點是白色的，所以其他地方都會是白的。

```
always@*
    if(h_cnt >= 0 && h_cnt <45 && v_cnt < 90 )
        pixel_addr= dsp3*45 + v_cnt*585 +h_cnt % 45;
    else if(h_cnt >= 45 && h_cnt <90 && v_cnt < 90 )
        pixel_addr= dsp2*45 + v_cnt*585+h_cnt % 45;
    else if(h_cnt >= 90 && h_cnt <135 && v_cnt < 90 )
        pixel_addr= dsp1*45 + v_cnt*585+h_cnt % 45;
    else if(h_cnt >= 135 && h_cnt <180 && v_cnt < 90 )
        pixel_addr= dsp0*45 + v_cnt*585+h_cnt % 45;
    else
        pixel_addr= 0;
```

討論：

這題就必須了解真正的運作原理，才能取得正確的 address，但是圖沒有截的很精準，沒有讓它剛好等寬，所以在顯示一些數字的時候會留一點邊邊。

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.

3.3 (Bonus) The same function of 3.1 and 3.2 are designed except that basic elements are stacked up until they are higher than the height of the playing space.

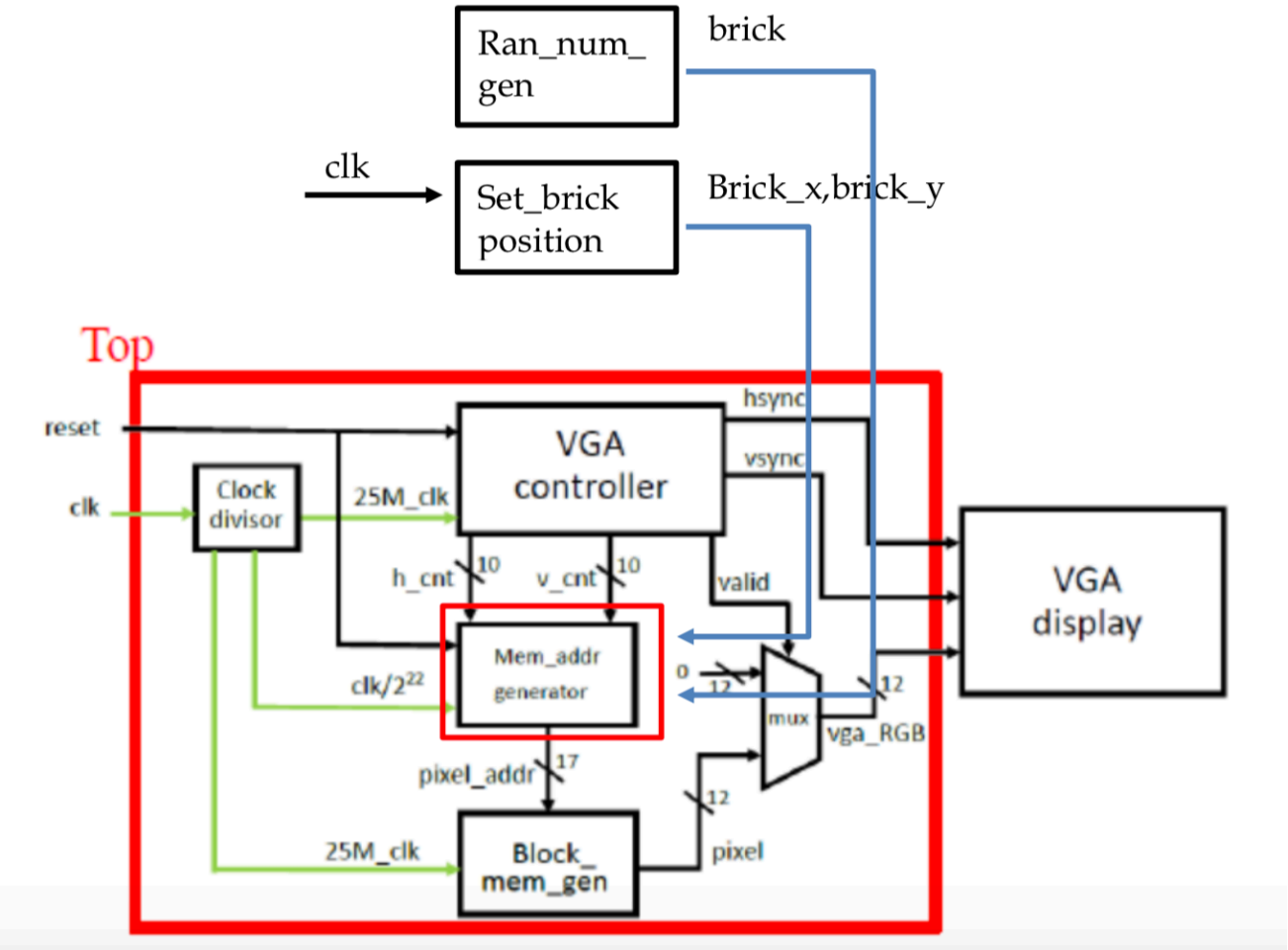
Specification:

Input: clk, rst

Output: vgaRed[3:0], vgaGreen[3:0], vgaBlue[3:0], hsync, vsync

I/O port:

W5 [get_ports clk] U17 [get_ports rst] G19 [get_ports {vgaRed[0]}] H19 [get_ports {vgaRed[1]}] J19 [get_ports {vgaRed[2]}] N19 [get_ports {vgaRed[3]}] N18 [get_ports {vgaBlue[0]}] L18 [get_ports {vgaBlue[1]}] K18 [get_ports {vgaBlue[2]}] J18 [get_ports {vgaBlue[3]}] J17 [get_ports {vgaGreen[0]}] H17 [get_ports {vgaGreen[1]}] G17 [get_ports {vgaGreen[2]}] D17 [get_ports {vgaGreen[3]}] P19 [get_ports hsync] R19 [get_ports vsync] L1 [get_ports {test_LED[15]}] P1 [get_ports {test_LED[14]}] N3 [get_ports {test_LED[13]}] P3 [get_ports {test_LED[12]}] U3 [get_ports {test_LED[11]}] W3 [get_ports {test_LED[10]}] V3 [get_ports {test_LED[9]}] V13 [get_ports {test_LED[8]}] V14 [get_ports {test_LED[7]}] U14 [get_ports {test_LED[6]}] U15[get_ports {test_LED[5]}] W18 [get_ports {test_LED[4]}] V19 [get_ports {test_LED[3]}] U19 [get_ports {test_LED[2]}] E19 [get_ports {test_LED[1]}] U16 [get_ports {test_LED[0]}]



Implementation:

random_num_gen ，根據講義裡產生隨機的 module，利用 shift register 並且依值指定下一個[0]的值，就能產生看起來很隨機的值，這裡是取 8bits 的，最後再將那個數字除 7 取餘數，在 in 是 posedge 的時候就能隨機產生一個 0 到 6 的數字。

Set_brick position ，用來決定下落方塊的位置的 。

這是將 lab 附的方塊圖全部轉成橫的，設計每一個小的方格都是 20*20，在把它轉檔加進來。所以整個遊戲版面寬是 20*10 高是 20*20，並把它放在中間。

```
    wire in;
    assign test_LED[15] = in;
    assign in = (rst==1 || brick_y==400)? 1:0;
    always@(posedge clk_20HZ or posedge rst)
        if(rst==1 || brick_y==400)
            brick_y <=0;
        else
            brick_y <= brick_y_tmp;
    assign brick_y_tmp = brick_y+1;
    assign test_LED[12:3] = brick_y;
```

```
    random_num_gen U0(.clk(clk), .in(in), .f(brick));
    assign brick_x = 60;
```

這是一個 counter，brick_x 跟 brick_y 就是方塊最左上角在遊戲版面的位置，所以 y 是一個 0~399 的 counter，而 In 則代表在數到 399 時或按 rst 時就要重新隨機一個數字

mem_addr_gen 裡，先定義要取的方塊的開頭位置，

```
`define brcik1_x 0
`define brcik1_y 0
`define brcik2_x 100
`define brcik2_y 0
`define brcik3_x 180
`define brcik3_y 0
`define brcik4_x 260
`define brcik4_y 0
`define brcik5_x 340
`define brcik5_y 0
`define brcik6_x 420
`define brcik6_y 0
`define brcik7_x 500
`define brcik7_y 0
```

```
if(h_cnt > 220 && h_cnt <= 420 && v_cnt > 40 && v_cnt <= 440 )begin
        if(h_cnt > 220+brick_x && h_cnt <= 300+brick_x && v_cnt > 40+brick_y && v_cnt <= 80+brick_y)
            pixel_addr = sx + (v_cnt - brick_y - 40)*580 + (h_cnt - brick_x -220);
        else
            pixel_addr = 0;
        vavid_in = 1;
    end
    else begin
        pixel_addr = 0;
        vavid_in = 0;
    end
```

先在紙上打草稿，想想怎麼設計，vavid_in 是讓在遊戲區域外面的都變成黑色，所以多接了一條出去，在最後要顯示的時候，就把範圍外的變成黑色。

sx,sy 則是根據要顯示的方塊種類決定要從記憶體的哪裡讀取。

```
 case(brick)
    3'b001:begin
        sx = `brcik1_x;
        sy = `brcik1_y;
    end
    3'b010:begin
        sx = `brcik2_x;
        sy = `brcik2_y;
    end
    3'b011:begin
        sx = `brcik3_x;
        sy = `brcik3_y;
    end
    3'b100:begin
        sx = `brcik4_x;
        sy = `brcik4_y;
    end
    3'b101:begin
        sx = `brcik5_x;
        sy = `brcik5_y;
    end
    3'b110:begin
        sx = `brcik6_x;
        sy = `brcik6_y;
    end
    3'b111:begin
        sx = `brcik7_x;
        sy = `brcik7_y;
    end
```

```
    default:begin
        sx = `brcik7_x;
        sy = `brcik7_y;
    end

    endcase
```

討論：

一開始產生隨機的地方弄了很久，因為不知道那個是從 1111111 開始，又有很多奇怪的 bug，而在決定 address 的地方又要想很久才能取到對的位置。而且講義給的圖其實不是正方形的，所以邊邊有些會截到。

結論：

這次實驗要打的 code 很少，但是每一個都要想得很仔細，邏輯要很清晰，還要把圖先用成 coe 檔，再放入 project 裡，圖片的長寬和大小都很重要，這次實驗真的很難，但是 final project 也要用到 VGA，所以要好好弄懂。