LAB11

前言: 因 clk_generator 跟 vga_controller 都一樣，故不重複描述

第一題

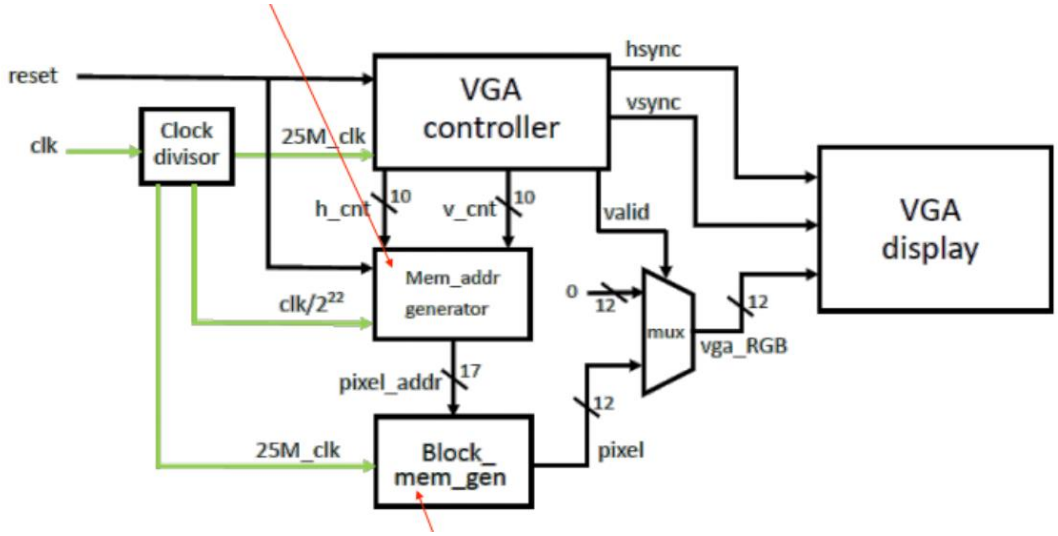
DESIGN SPECIFICATION

(1) INPUT / OUTPUT

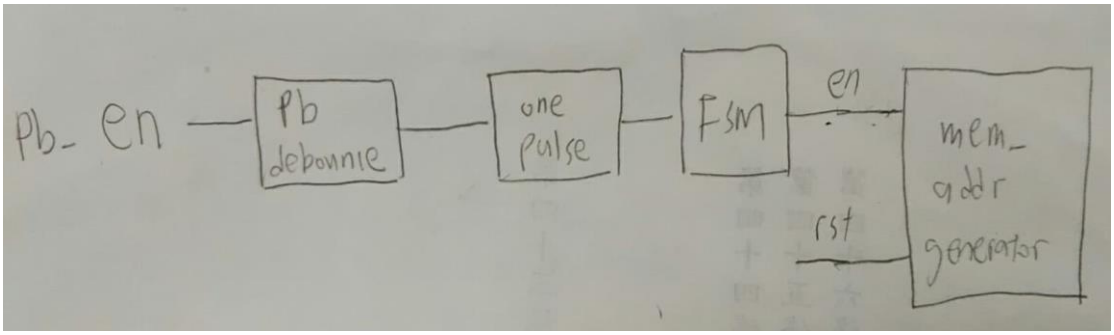
```
module top(
    input clk,
    input rst,
    input reset,
    input pb_en,
    output [3:0] vgaRed,
    output [3:0] vgaGreen,
    output [3:0] vgaBlue,
    output hsync,
    output vsync
);

module vga_controller
    (
        input wire pclk,reset,
        output wire hsync,vsync,valid,
        output wire [9:0]h_cnt,
        output wire [9:0]v_cnt
      );

module mem_addr_gen(
    input clk,
    input rst,
    input [9:0] h_cnt,
    input [9:0] v_cnt,
    input en,
    input reset,
    output [16:0] pixel_addr
);

module FSM2(in, clk, rst_n, count_enable);
```

input in;
input clk;
input rst_n;
output reg count_enable;

(2) BLOCK DIAGRAM



Mem_addr generator 再接



DESIGN IMPLEMENTATION

(1) I / O PINS

vgaBlue[3]	OUT		J18
vgaBlue[2]	OUT		K18
vgaBlue[1]	OUT		L18
vgaBlue[0]	OUT		N18
aGreen (4)	OUT		
vgaGreen[3]	OUT		D17
vgaGreen[2]	OUT		G17
vgaGreen[1]	OUT		H17
vgaGreen[0]	OUT		J17
aRed (4)	OUT		
vgaRed[3]	OUT		N19
vgaRed[2]	OUT		J19
vgaRed[1]	OUT		H19
vgaRed[0]	OUT		G19
alar ports (6)			
clk	IN		W5
hsync	OUT		P19
pb_en	IN		U18
reset	IN		V17
rst	IN		U17
vsync	OUT		R19

(2) VERILOG CODE

Module vga controller

同附件裡附的，讓 h_cnt、v_cnt 不斷掃描

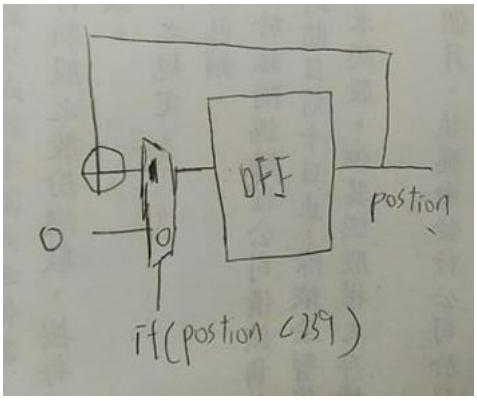
Module mem_addr_gen

放大圖片 ，並滾動

```
pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1)+ position*320 )% 76800;
```

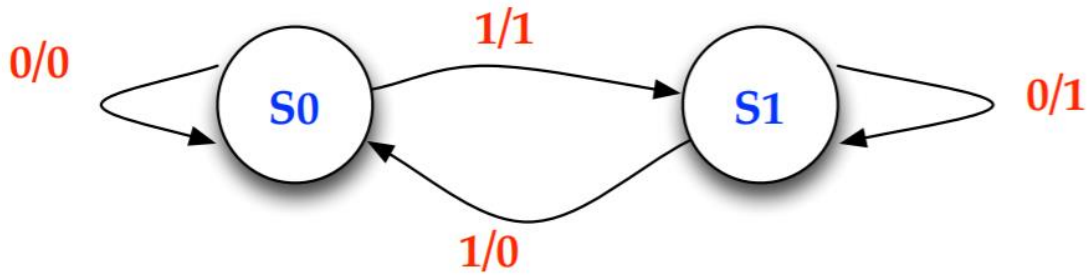
把 h_cnt，v_cnt 除 2，所以掃到 2 個等於圖片的一格

並利用累加的 postion 來滾動



Module FSM2

控制 enable



DISSCUSSION

這題遇到的困難就是理解吧，了解 VGA 的原理之後就不難了，而且附件裡都幫你做好了，只要自己加個 RESET 跟 ENABLE 即可。

結果就是按下 RESET 圖片回到最初的位置，按下 ENABLE，控制要滾還是不動。

CONCLUSION

第一次接觸到螢幕，其原理真是不好理解，問同學之後才會的。

第二題

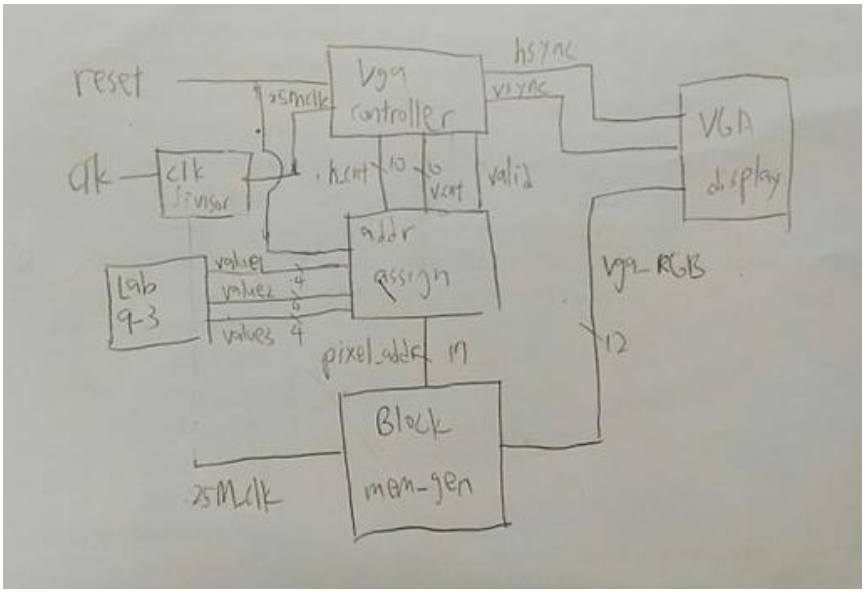
DESIGN SPECIFICATION

(1) INPUT / OUTPUT

```
module top(
    input clk,
    input rst,
    input rst_n,
    inout PS2_DATA,
```

```
    inout PS2_CLK,
    output [3:0] vgaRed,
    output [3:0] vgaGreen,
    output [3:0] vgaBlue,
    output hsync,
    output vsync
);

 module calculator // LAB9-3
      input clk;
      input rst_n;
      inout PS2_DATA;
      inout PS2_CLK;
      output neg;
      output [1:0]state;
      output [1:0]ASD;
      output [3:0]value1_1, value1_2, value1_3, value1_4;
      output [3:0]value2_1, value2_2;
     output [3:0]value3_1, value3_2, value3_3, value3_4;

 module addr_assign
      input clk;
      input rst;
      input valid;
      input [1:0]ASD;
      input neg;
      input [9:0]h_cnt;
      input [9:0]v_cnt;
      input [3:0]value1_1, value1_2, value1_3, value1_4;
      input [3:0]value2_1, value2_2;
      input [3:0]value3_1, value3_2, value3_3, value3_4;
      output reg [16:0]addr;
      reg [16:0]addr_temp;
```

(2) BLOCK DIAGRAM



DESIGN IMPLEMENTATION

(1) I / O PINS

vgaBlue (4)	OUT		
vgaGreen (4)	OUT		
vgaRed (4)	OUT		
Scalar ports (0)			
Scalar ports (7)			
clk	IN		W5
hsync	OUT		P19
PS2_CLK	INOUT		C17
PS2_DATA	INOUT		B17
rst	IN		U17
rst_n	IN		V17
vsync	OUT		R19

(2) VERILOG CODE

Module calculator

其實就是 9-3

value1_1,2,3,4 分別是 value1 的個十百千位

Value2_1,2 是 value2 的個十位

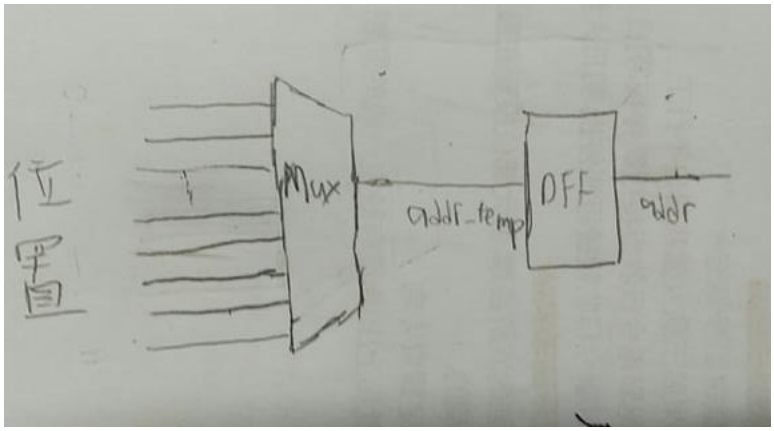
Value3_1,2,3,4 分別是 value3 的個十百千位

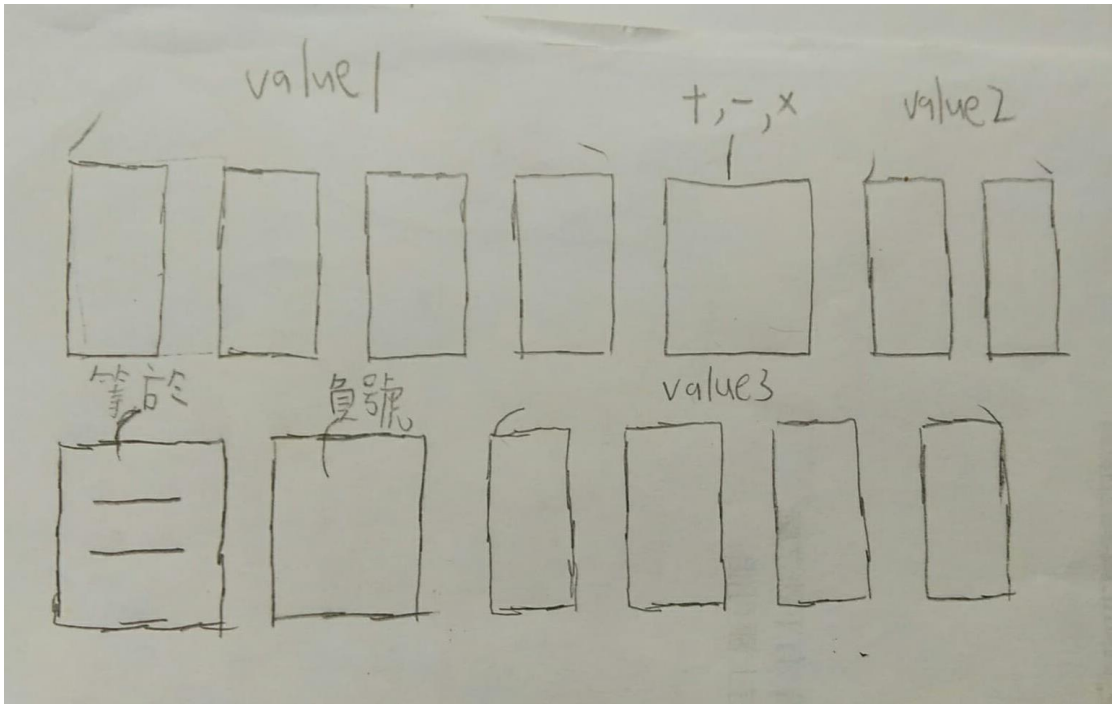
neg 是現在是否為負數

ASD 是加減乘 A 是加 S 是減 D 是乘

Module addr_assign

這個 module 要判斷在哪個位置要顯示哪個數字，在哪個位置我用 if / else if / else 去選，而要顯示哪個數字，我是根據 COE 黨的編排去處理，因為我的 0~5000 是圖片 0，5001~10000 是圖片 1，以此類推，所以我直接讓 value*5000 就知道從哪裡開始讀





DISSCUSSION

打了這題才發現第一題懂得根本只是皮毛而已，這題又花很多時間完全搞懂 VGA，而且我一開始打的很不簡潔，我多打了一個 MODULE 去選擇 ADDR 的值，那是因為我沒有善用 COE 黨的編排模式，後來發現都是 5000 得倍數就把它簡化了。

這題遇到最久的 BUG 就是我忘記改圖片檔的大小了，還是照著範例 76800，但我的圖片大小是 90000，這個 BUG 是意外抓到的，不然我本來也想放棄了。

CONCLUSION

這題讓我完全搞動 VGA 了，很有成就感

第三題 // 我只有打 3-2

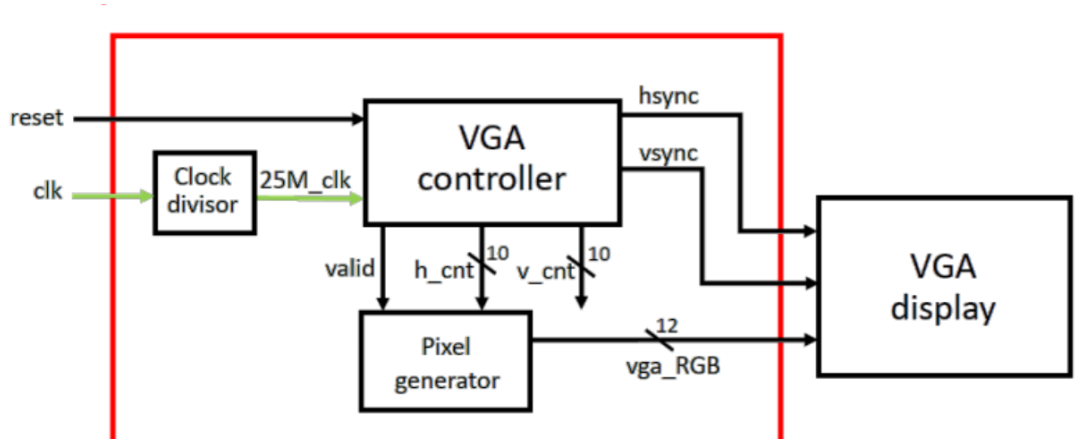
DESIGN SPECIFICATION

(1) INPUT / OUTPUT

```
module top
        input clk,
        input rst,
        output [3:0] vgaRed,
        output [3:0] vgaGreen,
        output [3:0] vgaBlue,
        output hsync,
        output vsync

module random(clk, num, rst, en);
        input clk;
        input rst;
        input en;
       output [2:0]num;
```

```
module pixel_generator
    input clk;
    input clk_1hz;
    input rst;
    input valid;
    input [9:0]h_cnt;
    input [9:0]v_cnt;
    input [2:0]num;
    output reg [11:0]pixel;
    output reg en;
```

(2) BLOCK DIAGRAM



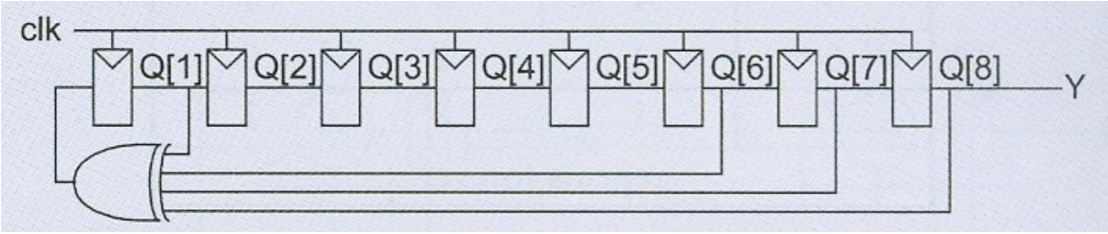
DESIGN IMPLEMENTATION

(1) I / O PINS

vgaBlue (4)	OUT		
vgaGreen (4)	OUT		
vgaRed (4)	OUT		
Scalar ports (4)			
clk	IN		W5
hsync	OUT		P19
rst	IN		U17
vsync	OUT		R19

(2) VERILOG CODE

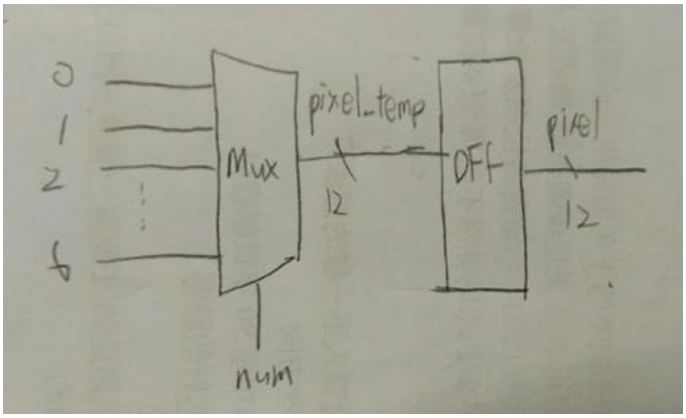
Module random



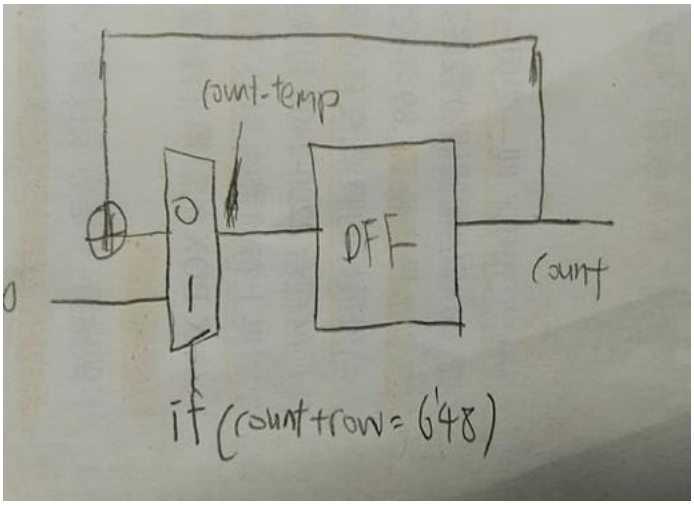
然後再 output 出{Q[2:0]}

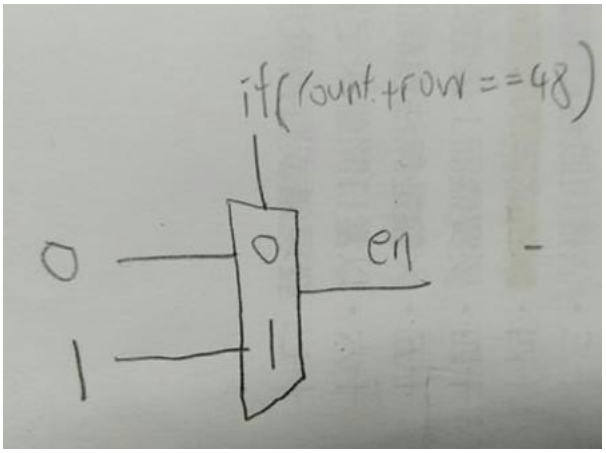
Module pixel_generator

先利用 random 傳出來的隨機數字去選擇方塊種類



再用 1hz 的 counter 讓它每秒下移，當 counter 數到底，就重製為 0，並讓 enable 打開，讓隨機數字進來。





DISSCUSSION

第二題對 VGA 有較深的了解之後，就比較容易，比較可惜的是沒時間打第三小題，後來有想到方法可是沒時間嘗試，還有我這題有一個很奇怪的 BUG，只有背景是黑色才能正常，其餘都怪怪的。

CONCLUSION

打這題是因為跟 FINAL PROJECT 有點相關，希望會有幫助!