lab11_1 姓名：廖泓全 學號：106061219

Design Specification

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].

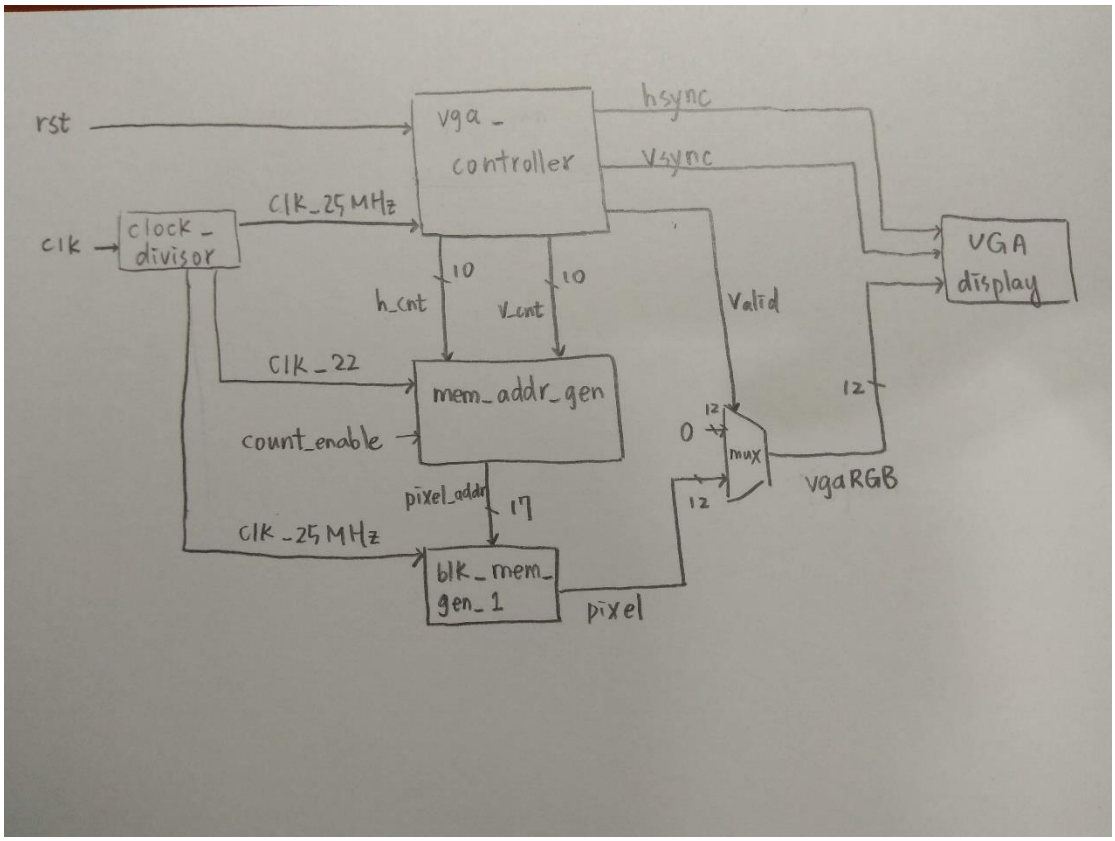
1.2 At the beginning or when reset (button) is pressed, the VGA display shows the image (e.g. amumu.jpg). The VGA image stay still until en (button) is pressed.

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.

Input: clk, rst, start(use to start or resume the scrolling function)

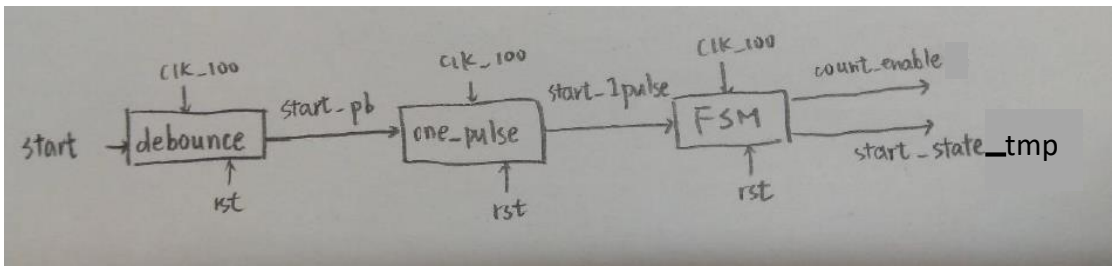
Output: vgaRed[3:0], vgaGreen[3:0], vgaBlue[3:0], hsync, vsync

The block diagram:



Design Implementation

・To generate the count_enable:



The code of start/resume scrolling in the mem_addr_gen module:

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

The I/O pin assignment for design:

Input	**clk**	**rst**	**start**
LOC	W5	U17	T18

output	**hsync**	**vsync**
LOC	P19	R19

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

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

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

Discussion

・首先是用 clock_dividor 做出掃描螢幕需要用到的 clk_25MHz 和 scroll up 需要的 clk_22 的頻率。

・vga_controller (用 clk_25MHz) 是用來控制目前掃描到螢幕上的哪一個 pixel，分別由 hsync 和 vsync 控制。

・mem_addr_gen (用 clk_22) 是用來取圖片中的哪一個點，由於 COE file 中是用一維的形式儲存圖片，所以我們需要用接入此 module 的 h_cnt 和 v_cnt 去計算二維轉成一維的 pixel_addr，這樣我們才有辦法把這個輸入到 blk_mem_gen 中，然後再把我們要的圖片顯示到 VGA 上。此外，為了要讓圖片產生向上捲動的樣子，所以此題先自訂一個參數 position，然後根據 clk_22，遇到 posedge 就

讓 position 加一，然後再經由 module 中的其他處理，這樣就能完成上捲的樣子。

・count_enable 的部分則是先將 start 經由 debounce, one_pulse, FSM 三個 module 處理後，就可以產生出 count_enable。

・outcome: the picture of scrolling down

Conclusion

・此題由於老師有提供主要的程式給我們，所以只要自己加上 decounce, one_pulse, FSM 這三個 module 去產生出 count_enable，然後再接入 mem_addr_gen module 中作為一個控制條件，這樣就能完成了。

Reference

・參考講義中的 block diagram 和老師提供的範例程式。

lab11_2

Design Specification

Calculator display.

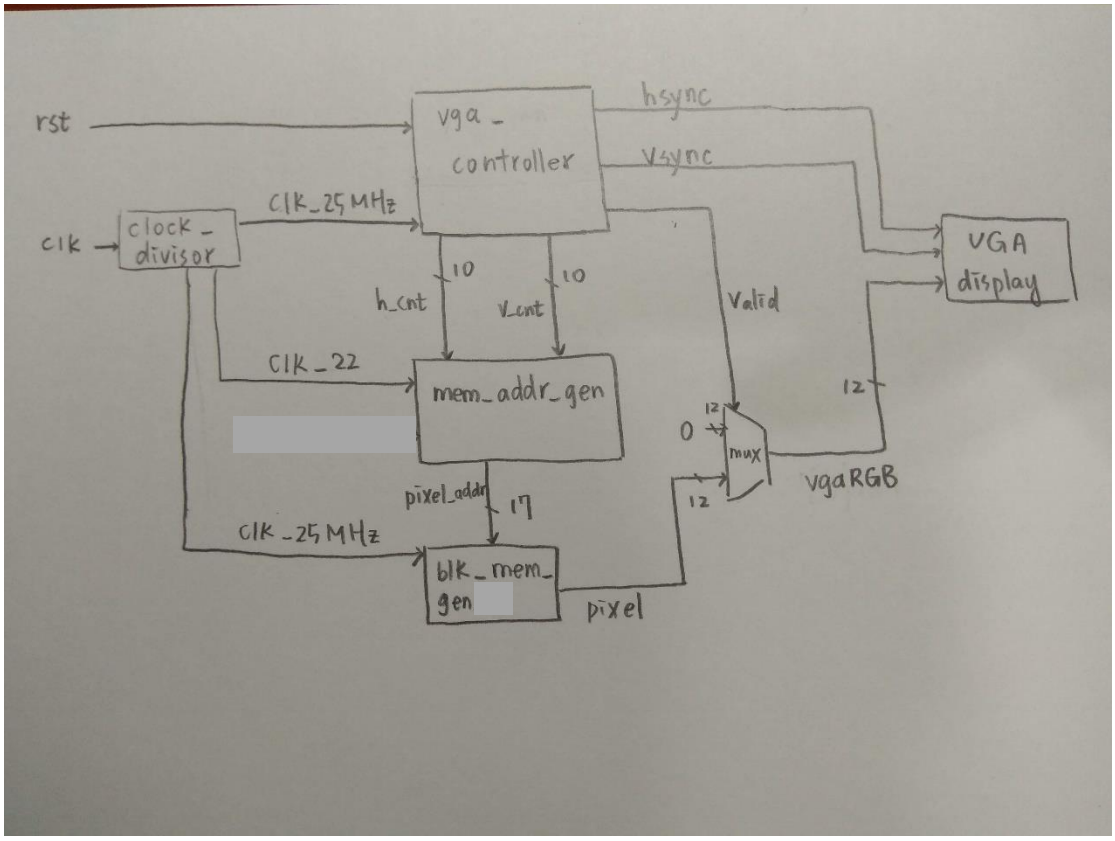
2.1 Combine the key board controller and VGA displaying controller to design a calculator with 2-digit addition/subtraction/multiplication. The display function should be the same as usual calculator or APP in the smartphone.

Inout: PS2_CLK, PS2_DATA

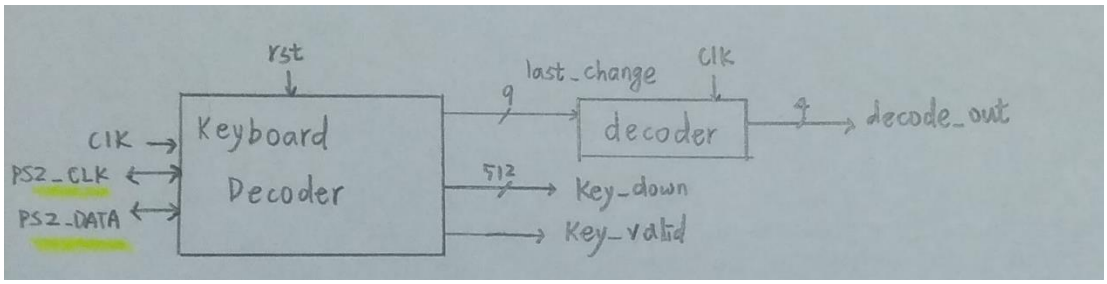
Input: clk, rst

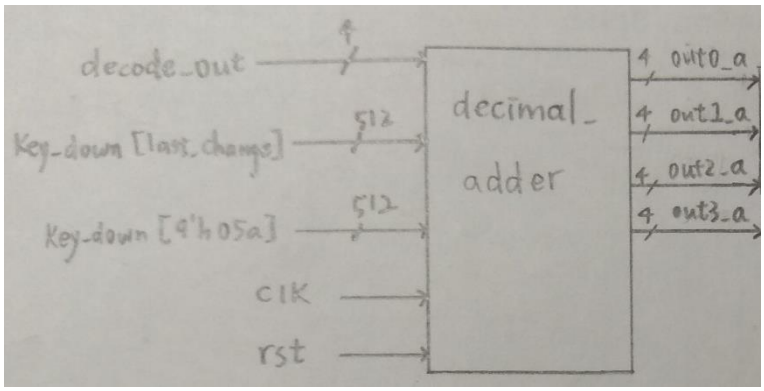
Output: vgaRed[3:0], vgaGreen[3:0], vgaBlue[3:0], hsync, vsync

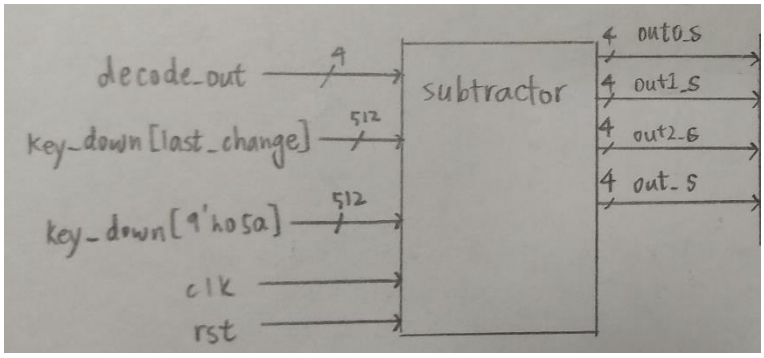
The block diagram and the code of mux:

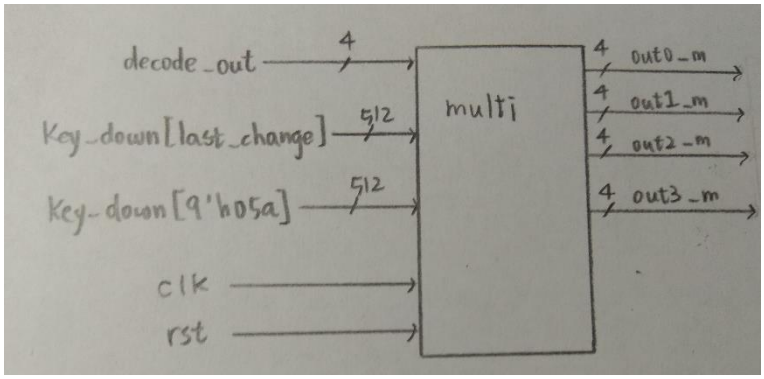


其中的 blk_mem_gen 是由 IP catalog 設定出來的，因為此題中有 13 張圖片(即 13 個 blk_mem_gen)，所以在 block diagram 中僅畫出一個作為代表。










```
always @ (posedge clk or posedge rst)
  if (rst) begin
    out3 <= 4'd0;
    out2 <= 4'd0;
    out1 <= 4'd0;
    out0 <= 4'd0;
    mode <= 2'd0; end
  else if (key_down[9'h079]) begin
    out3 <= out3_a;
    out2 <= out2_a;
    out1 <= out1_a;
    out0 <= out0_a;
    mode <= 2'd1; end
  else if (key_down[9'h07B]) begin
    out3 <= out3_s;
    out2 <= out2_s;
    out1 <= out1_s;
    out0 <= out0_s;
    mode <= 2'd2; end
  else if (key_down[9'h07C]) begin
    out3 <= out3_m;
    out2 <= out2_m;
    out1 <= out1_m;
    out0 <= out0_m;
    mode <= 2'd3; end
else
  if (mode==2'd1) begin
    out3 <= out3_a;
    out2 <= out2_a;
    out1 <= out1_a;
    out0 <= out0_a;
    mode <= mode; end
  else if (mode==2'd2) begin
    out3 <= out3_s;
    out2 <= out2_s;
    out1 <= out1_s;
    out0 <= out0_s;
    mode <= mode; end
  else if (mode==2'd3) begin
    out3 <= out3_m;
    out2 <= out2_m;
    out1 <= out1_m;
    out0 <= out0_m;
    mode <= mode; end
  else begin
    out3 <= out3_a;
    out2 <= out2_a;
    out1 <= out1_a;
    out0 <= out0_a;
    mode <= mode; end
```

◎此處是用 MUX 來選擇 out0, out1, out2, out3 是 adder, subtractor, multiplier 的輸出中的哪一個。

Design Implementation

・KeyboardDecoder 是由 KeyboardCtrl 和 Ps2Interface 一層一層包起來的。

The I/O pin assignment for design:

Input	**clk**	**rst**
LOC	W5	U17

Inout	**PS2_CLK**	**PS2_DATA**
LOC	C17	B17

output	**hsync**	**vsync**
LOC	P19	R19

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

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

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

Discussion

・第一步：由 KeyboardDecoder 產生出 last_change[8:0], key_down[511:0], key_valid，以及 inout: PS2_CLK, PS2_DATA。並且用 clock_dividor module 做出需要的控制頻率，如：clk_25MHz, clk_22。

・第二步：將 last_change[8:0]輸入至 decoder module 中，產生出 4-bit 的 decode_out。

・第三步：分成 adder, subtractor, multiplier 三部分。

1. 把 decode_out, key_down[last_change], key_down[9'h05a], clk, rst 輸入到 decimal_adder module 中，作為內部程式的 input 和控制訊號，之後再輸出 out3_a, out2_a, out1_a, out0_a。
2. 把 decode_out, key_down[last_change], key_down[9'h05a], clk, rst 輸入到 subtractor module 中，作為內部程式的 input 和控制訊號，之後再輸出 out3_s, out2_s, out1_s, out0_s。
3. 把 decode_out, key_down[last_change], key_down[9'h05a], clk, rst 輸入到 multi module 中，作為內部程式的 input 和控制訊號，之後再輸出 out3_m,

out2_m, out1_m, out0_m。

・利用 MUX 選擇 adder, subtractor, multiplier 三者分別產生的 output，並搭配 mode 來使 out3, out2, out1, out0 的值穩定為 adder, subtractor, multiplier 三者其中之一。

・vga_controller (用 clk_25MHz) 是用來控制目前掃描到螢幕上的哪一個 pixel，分別由 hsync 和 vsync 控制。

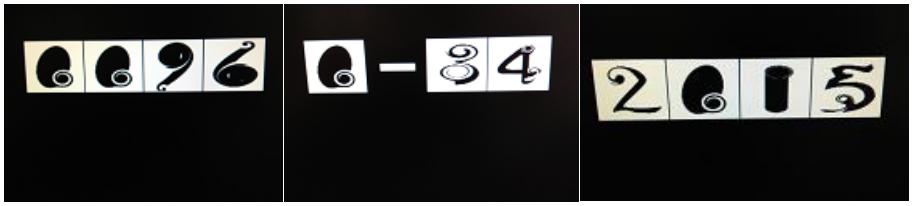
・mem_addr_gen (用 clk_22) 是用來取圖片中的哪一個點，由於 COE file 中是用一維的形式儲存圖片，所以我們需要用接入此 module 的 h_cnt 和 v_cnt 去計算二維轉成一維的 pixel_addr，這樣我們才有辦法把這個輸入到 blk_mem_gen 中，然後再把我們要的圖片顯示到 VGA 上。另外，由於要做出和之前 lab9_3 功能相似的 2 位數的計算機，所以我在螢幕上設定了 4 個方框(64*64/個)來模擬七段顯示器的功能，此外，我也做出 13 張 64*64 的圖片(分別是 0 1 2 3 4 5 6 7 8 9 + −×)。

・在 top module 中，我用一個 1 個大的 MUX (用來定義四個方框的位置) 包 4 個小的 case (選擇現在要顯示哪些圖案)。最後就能做出和 lab9_3 一樣的功能了。

Discussion

・outcome:

輸入 31　　輸入 65



31+65　　31-65　　31*65

Conclusion

・這題真的複雜許多，因為剛開始還沒有很了解螢幕到底是怎麼運作的，所以在計算二維轉一維的時候，就會一直算錯，讓顯示出來的圖形有點怪怪的，然後在最後的 vgaRed, vgaGreen, vgaBlue 的部分，起初也不知道如何去選擇要的圖片，之後再去研究老師給的範例程式和詢問助教和同學後，才知道原來是要用 MUX 的方式去選擇要顯示的圖片。

Reference

・參考老師的 KeyboardDecoder, KeyboardCtrl, Ps2Interface，以及講義中 last_change[8:0], key_down[511:0], key_valid 的使用方法。此外，也參考了講義中的 block diagram 和老師提供的範例程式。

lab11_3

Design Specification

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.

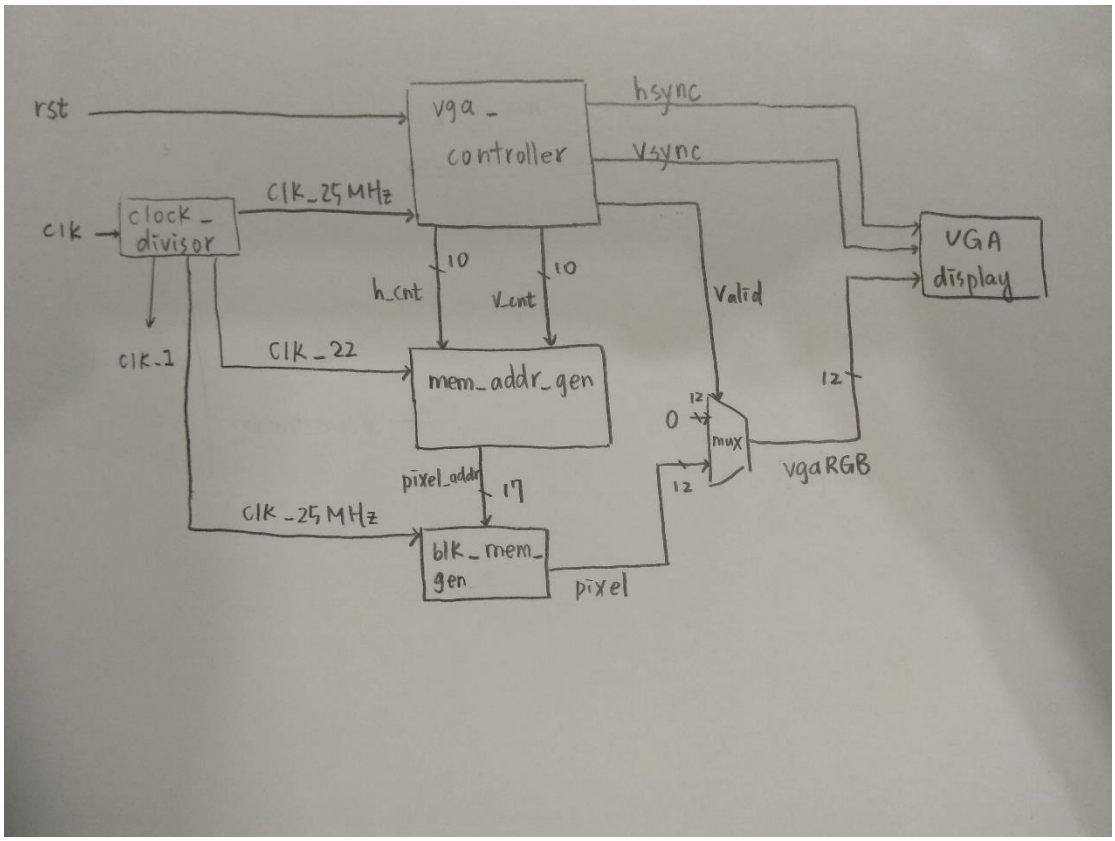
3.2 Each generated basic element moves down by the step of a square at the speed of 1Hz. Finally, they disappear below the playing space. When a basic element disappears, a new basic element is generated again and fall down again repeatedly.

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.

Input: clk, rst

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

The block diagram and the code of mux:



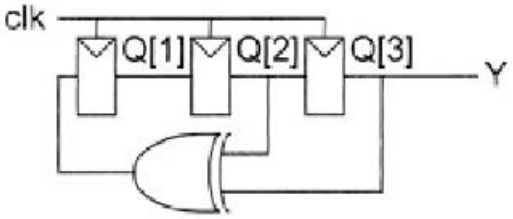
其中的 blk_mem_gen 是由 IP catalog 設定出來的，因為此題中有 7 張圖片(即 7 個 blk_mem_gen)，所以在 block diagram 中僅畫出一個作為代表。

Design Implementation

・The code and the logic diagram of linear feedback shift register:

```
assign tmp = q[2]^q[1];


always @(posedge pulse or posedge rst)
  if(rst)
  begin
    q[0] <= 1'b1;
    q[1] <= 1'b1;
    q[2] <= 1'b1;
  end
  else
  begin
    q[0] <= tmp;
    q[1] <= q[0];
    q[2] <= q[1];
  end
```



The I/O pin assignment for design:

Input	**clk**	**rst**
LOC	W5	U17

output	**hsync**	**vsync**
LOC	P19	R19

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

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

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

Discussion

・此題要求為了讓每次出現的俄羅斯方塊隨機顯示，可以用 LFSR (Linear feedback shift register) 來產生隨機數字，若是需要 0~7 共 3-bit 的亂數，可以用一個 3-bit 的 shift register，每間隔一個時週期就讓值位移一次，第一個的值則為後兩個的值 XOR 出來的結果，隨著 bits 數的拉長，就可以產生一個看似不規則的亂數結果。另外，因為要題目要求當方塊到底後，會消失並顯示新的方塊，所以像第一題設定了 position 來做下捲的動作。在利用 position 加上 pulse

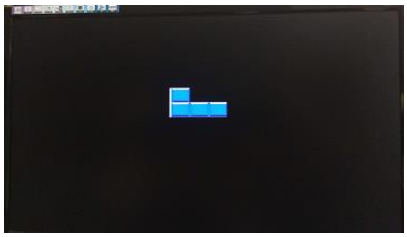
作為判斷是否產生出新的方塊的條件。

・vga_controller (用 clk_25MHz) 是用來控制目前掃描到螢幕上的哪一個 pixel，分別由 hsync 和 vsync 控制。

・mem_addr_gen (用 clk_22) 是用來取圖片中的哪一個點，由於 COE file 中是用一維的形式儲存圖片，所以我們需要用接入此 module 的 h_cnt 和 v_cnt 去計算二維轉成一維的 pixel_addr，這樣我們才有辦法把這個輸入到 blk_mem_gen 中，然後再把我們要的圖片顯示到 VGA 上。我做出每一個方塊的圖片(分別是 J, L, S, Z, line, O, T)，每一張圖片的大小皆有自己的設定值。

・在 MUX 的部分則和上一題差不多，也是利用 case 去選擇要顯示的圖片。

・outcome:



Conclusion

・這題和上一題的概念相似，先將每一張圖片放入 module 後，再用類似的選擇方法，就可以完成這題了。

Reference

・參考了講義中的 block diagram 和老師提供的範例程式。參考黃元豪老師所提供的 linear feedback shift register。