

Lab 7

學號：108071003

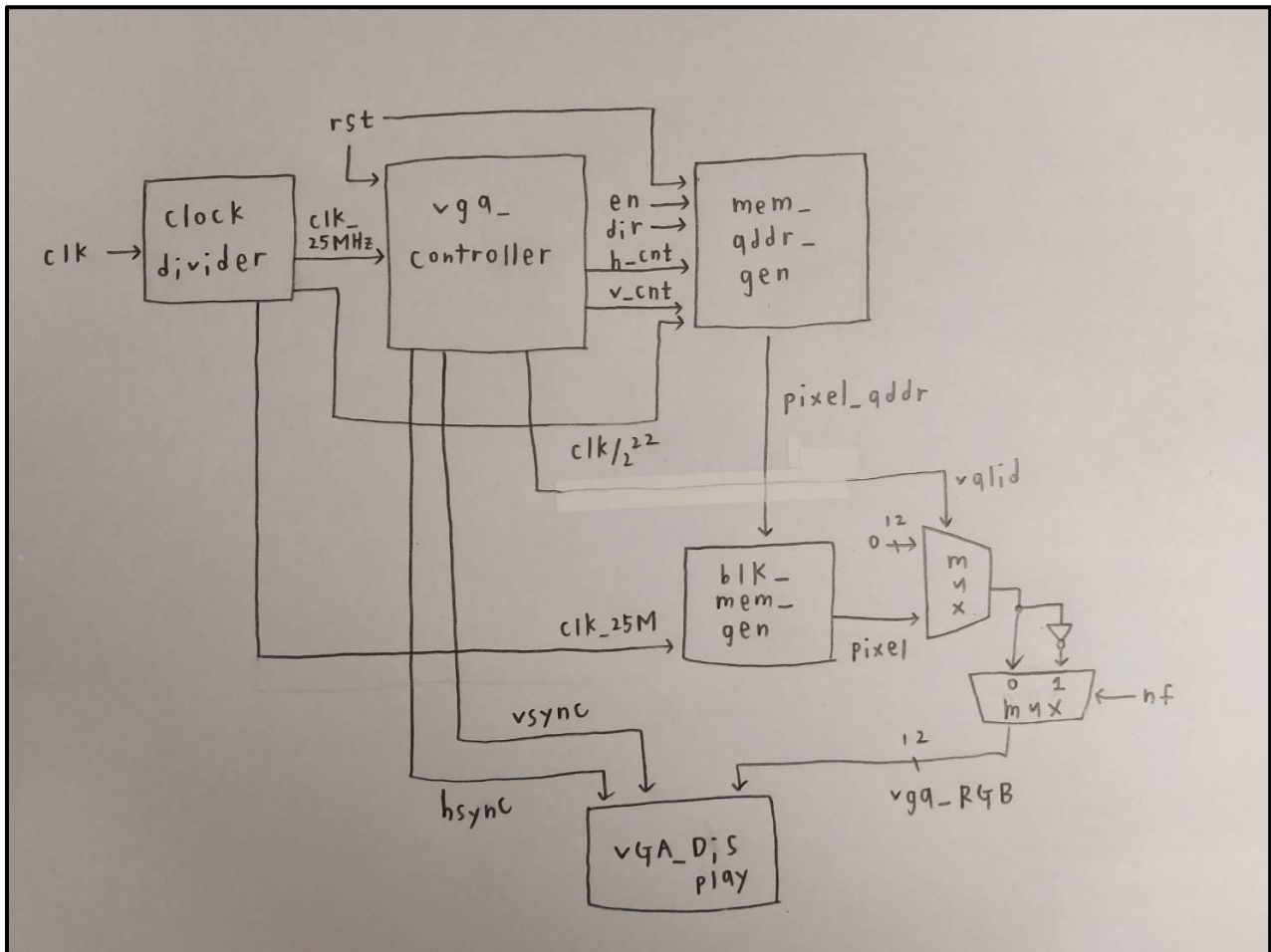
姓名：李彥璋

1. 實作過程

Lab7_1

Lab7_1 將圖片顯示於 VGA 螢幕，並實作向上、向下位移、負片等功能。

Block diagram:



Clock divider:

Input: clk.

output: clk_25M, clk/2²².

Vga_controller:

Input: clk_25M, rst.

Output: h_cnt, v_cnt, valid, hsync, vsync.

Mem_addr_gen:

Input: rst, en, dir, h_cnt, v_cnt, clk/2²².

Output: pixel_addr.

Blk_mem_gen:

Input: pixel_addr, clk_25M.

Output: pixel.

Vga_display:

Input: vga_RGB, vsync, hsync.

Code explanation:

```

90 module mem_addr_gen(
91     input clk,
92     input rst,
93     input en,
94     input dir,
95     input [9:0] h_cnt,
96     input [9:0] v_cnt,
97     output [16:0] pixel_addr
98 );
99
100 reg [7:0] position;
101
102 assign pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1)+position*320) % 76800; //640*480 --> 320*240
103
104 always @ (posedge clk or posedge rst) begin
105     if(rst) begin
106         position <= 0;
107     end else begin
108         if(en == 1'b0) begin
109             position <= position;
110         end else begin
111             if(dir == 1'b0) begin
112                 if(position < 239) position <= position + 1;
113                 else position <= 0;
114             end else begin
115                 if(position > 0) position <= position - 1;
116                 else position <= 239;
117             end
118         end
119     end
120 end
121 endmodule

```

在 Always block 當中,決定 position 的值,用以將圖片上下位移。

If 按下 rst, position = 0,圖片按照原本位置輸出。

Else,

If en==0, position 維持原值,使圖片暫停移動,停留在當前位置。

Else,

If dir==0,代表圖片要向上位移,若 position 未超出範圍, +1 ; 否則, =0。


Else,代表圖片要向下位移,若 position 未超出範圍, -1 ; 否則, =239。

102 行, pixel_addr 的賦值:

- ✧ h_cnt & v_cnt 要/2(>>1)是因為我們要將 640*480 的圖片->320*240,可以想成原本的 4 小格變成 1 大格。
- ✧ v_cnt 要*320 是因為一個橫列共有 320 個 pixel, position 同理。
- ✧ 最後要%76800 是因為 addr 固定是 0 - 76799,有可能運算超過,mod 回正確值。

原本這邊我一直有一個地方想不通,就是為何向上位移 position 是+1?

後來理解後(要從下面開始印),

我覺得有一個很好的比喻:划手機要往上看的時候,手指是向下滑 .

```

83 | assign {vgaRed, vgaGreen, vgaBlue} = (valid == 1'b1) ? ((nf == 1'b0) ? (pixel):(~pixel)) : 12'd0;
84 | assign data = {vgaRed, vgaGreen, vgaBlue};
85 | blk_mem_gen_0 mem_gen(.addra(pixel_addr), .clka(clk_25M), .dina(data), .douta(pixel), .wea(0));

```

83 行, 決定 RGB 的值,

若 valid==1, nf 判斷是否需要負片, 如果不用 pixel; 如果要 ~pixel.

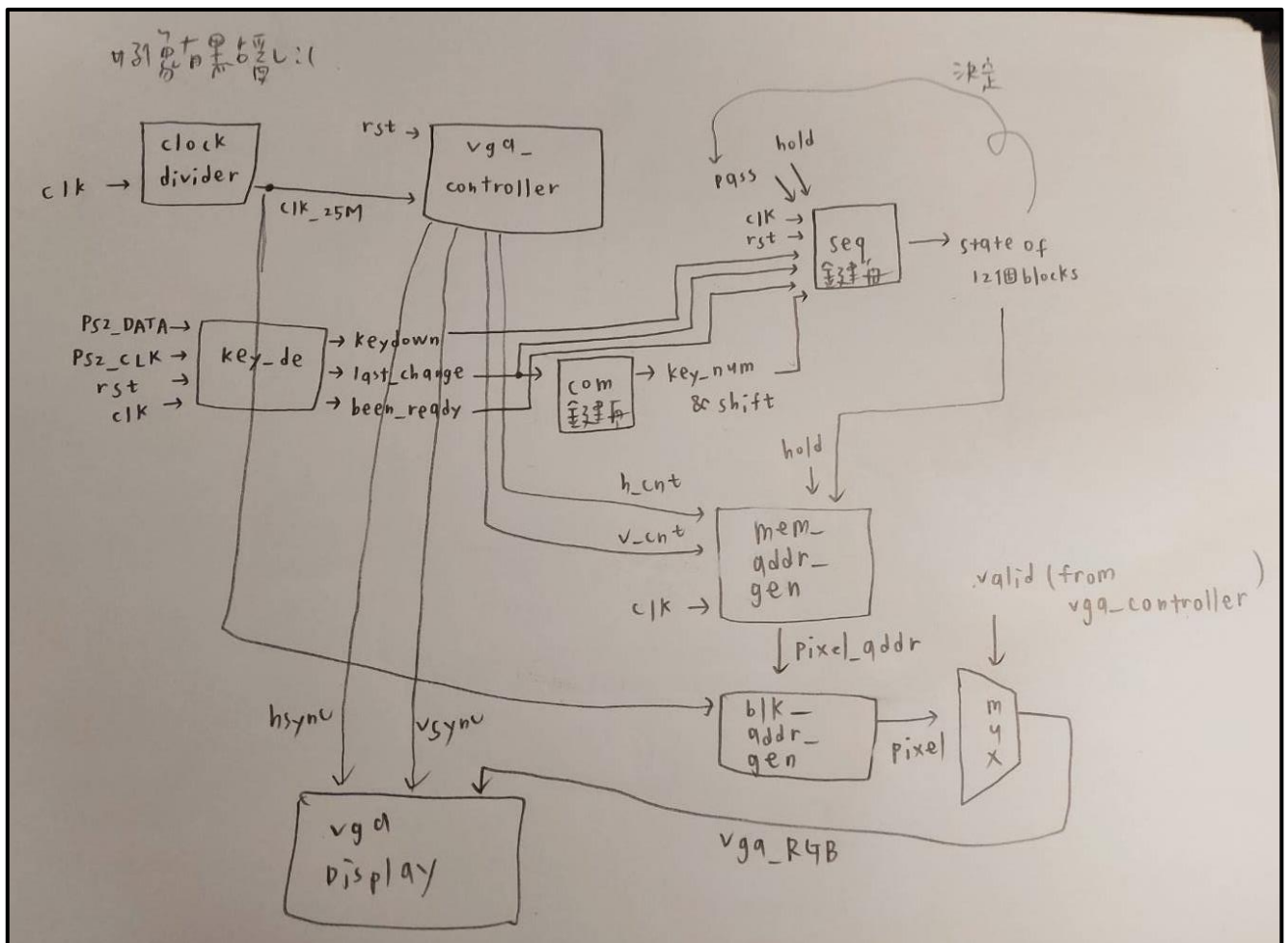
若 valid==0, 12'd0.

Lab7_2

Lab7_2 將圖片顯示於 VGA 螢幕, 分成 12 塊, 每一塊可以各自順、逆時鐘 90 度旋轉(以 12 個鍵盤+shift 控制)。

另有 hint mode 提示玩家原圖片, 需特別注意於 hint mode 圖片不可旋轉(因為我錯了)。

Block diagram:



Code explanation:

```

54     parameter [8:0] LEFT_SHIFT_CODES  = 9'b0_0001_0010;
55     parameter [8:0] RIGHT_SHIFT_CODES = 9'b0_0101_1001;
56     parameter [8:0] KEY_CODES [0:11] = {
57         9'b0_0100_0100, // O => 44
58         9'b0_0100_1101, // P => 4D
59         9'b0_0101_0100, // [ => 54
60         9'b0_0101_1011, // ] => 5B
61         9'b0_0100_0010, // K => 42
62         9'b0_0100_1011, // L => 4B
63         9'b0_0100_1100, // ; => 4C
64         9'b0_0101_0010, // ' => 52
65         9'b0_0011_1010, // M => 3A
66         9'b0_0100_0001, // , => 41
67         9'b0_0100_1001, // . => 49
68         9'b0_0100_1010 // / => 4A
69     };

```

兩個 shift、OP[]KL;'M,./的 keycodes.

```

87     always @ (*) begin
88         case (last_change)
89             KEY_CODES[00] : key_num = 4'b0000;
90             KEY_CODES[01] : key_num = 4'b0001;
91             KEY_CODES[02] : key_num = 4'b0010;
92             KEY_CODES[03] : key_num = 4'b0011;
93             KEY_CODES[04] : key_num = 4'b0100;
94             KEY_CODES[05] : key_num = 4'b0101;
95             KEY_CODES[06] : key_num = 4'b0110;
96             KEY_CODES[07] : key_num = 4'b0111;
97             KEY_CODES[08] : key_num = 4'b1000;
98             KEY_CODES[09] : key_num = 4'b1001;
99             KEY_CODES[10] : key_num = 4'b1010;
100            KEY_CODES[11] : key_num = 4'b1011;
101            default: key_num = 4'b1111;
102        endcase
103    end

```

以 0,1,...,11 的 key_num 分別對應 OP[]KL;'M,./

```

106     wire shift;
107     assign shift = (key_down[LEFT_SHIFT_CODES] == 1'b1 || key_down[RIGHT_SHIFT_CODES] == 1'b1) ? 1'b1 : 1'b0;

```

以 shift 變數紀錄左或右其一的 shift 現在是否被按下.

```

109 reg [1:0] zero = 2'd0;
110 reg [1:0] one = 2'd1;
111 reg [1:0] two = 2'd2;
112 reg [1:0] three = 2'd3;
113 reg [1:0] four = 2'd3;
114 reg [1:0] five = 2'd2;
115 reg [1:0] six = 2'd1;
116 reg [1:0] seven = 2'd0;
117 reg [1:0] eight = 2'd1;
118 reg [1:0] nine = 2'd0;
119 reg [1:0] ten = 2'd3;
120 reg [1:0] ele = 2'd2;

```

以 one,two,three,...,ele 分別記錄 12 個塊圖片各自的 state(頭朝哪)。

```

124 always @(posedge clk or posedge rst) begin
125     if(rst == 1) begin
126         zero <= 2'd0;
127         one <= 2'd1;
128         two <= 2'd2;
129         three <= 2'd3;
130         four <= 2'd3;
131         five <= 2'd2;
132         six <= 2'd1;
133         seven <= 2'd0;
134         eight <= 2'd1;
135         nine <= 2'd0;
136         ten <= 2'd3;
137         ele <= 2'd2;
138     end else begin
139         if(been_ready && key_down[last_change] == 1'b1) begin
140             if(key_num != 4'b1111 && pass == 0 && hold == 0) begin
141                 if(key_num == 4'd0) begin
142                     if(shift == 1'b1) zero <= zero - 1;
143                     else zero <= zero + 1;
144                 end else if(key_num == 4'd1) begin
145                     if(shift == 1'b1) one <= one - 1;
146                     else one <= one + 1;
147                 end else if(key_num == 4'd2) begin
148                     if(shift == 1'b1) two <= two - 1;
149                     else two <= two + 1;
150                 end else if(key_num == 4'd3) begin
151                     if(shift == 1'b1) three <= three - 1;
152                     else three <= three + 1;
153                 end else if(key_num == 4'd4) begin
154                     if(shift == 1'b1) four <= four - 1;
155                     else four <= four + 1;
156                 end else if(key_num == 4'd5) begin

```

If 按下 rst,12 塊圖片回到原本的狀態。

Else,if been_ready && key_down[last_change]==1,

If 按下的是 12 個鈕之一 && 拼圖還沒拼好(pass==0) && 不是 hint mode(hold==0),

根據鍵盤(OP[]KL;'M,./、shift or not)調整 12 塊的 state。

接下來先說明我如何讓圖片旋轉：

我以 4 個大小為 6400(因為 1 大格是 80*80)的 array 紀錄 12 大格中最左上角那大格的 pixel_addr, 分別記錄的是圖片頭朝上、頭朝右、頭朝下、頭朝左的 pixel_addr.

這樣一來我只需要知道當前小格(h_cnt&v_cnt 算出來的 0~76799)是在 12 大格的哪一大格->將當前小格轉換成 0~6399 作為 array 的 index, 就能得到對應的 12 大格中最左上角那格的 pixel_addr, 接著再位移回正確的大格的 pixel_addr.

用文字表達好複雜🤔🤔🤔.

得到 array 的方式是使用 C++, code 如下,

```

8 void rotate(vector<vector<int> > &matrix) {
9     reverse(matrix.begin(), matrix.end());
10    for (int i = 0; i < matrix.size(); ++i) {
11        for (int j = i + 1; j < matrix[i].size(); ++j)
12            swap(matrix[i][j], matrix[j][i]);
13    }
14 }
15
16 int main() {
17     for(int i=0 ; i<80 ; i++) {
18         vector<int> tmp;
19         for(int j=0 ; j<80 ; j++) {
20             tmp.push_back(j+i*320);
21         }
22         matrix.push_back(tmp);
23     }
24     //rotate(matrix);
25     //rotate(matrix);
26     //rotate(matrix);
27     for(int i=0 ; i<80 ; i++) {
28         for(int j=0 ; j<80 ; j++) {
29             cout << "15'd" << matrix[i][j] << ",";
30         }
31         cout << "\n";
32     }
33     return 0;
34 }

```

[24-26] 是往右轉一次 90、二次 90、三次 90.

方法就是第 1 行跟尾行先互換位置, 然後所有[i][j]、[j][i]互換🤖.

```

* clockwise rotate
* first reverse up to down, then swap the symmetry
* 1 2 3      7 8 9      7 4 1
* 4 5 6  => 4 5 6  => 8 5 2
* 7 8 9      1 2 3      9 6 3

```



```

547 always @ (posedge clk) begin
548
549     if(hold == 1'b1) begin
550         if( (h_cnt>>1)+320*(v_cnt>>1) < 25600 ) begin
551             if( (((h_cnt>>1)+320*(v_cnt>>1)) %320) / 80 == 0 ) begin
552                 pixel_addr <= state_0[ (((h_cnt>>1)+320*(v_cnt>>1)) %320)%80 + (((h_cnt>>1)+320*(v_cnt>>1)) /320)*80 ];
553             end
554             else if( (((h_cnt>>1)+320*(v_cnt>>1)) %320) / 80 == 1 ) begin
555                 pixel_addr <= state_0[ (((h_cnt>>1)+320*(v_cnt>>1)) %320)%80 + (((h_cnt>>1)+320*(v_cnt>>1))-80) /320)*80 ]+80;
556             end
557             else if( (((h_cnt>>1)+320*(v_cnt>>1)) %320) / 80 == 2 ) begin
558                 pixel_addr <= state_0[ (((h_cnt>>1)+320*(v_cnt>>1)) %320)%80 + (((h_cnt>>1)+320*(v_cnt>>1))-160) /320)*80 ]+160;
559             end
560             else if( (((h_cnt>>1)+320*(v_cnt>>1)) %320) / 80 == 3 ) begin
561                 pixel_addr <= state_0[ (((h_cnt>>1)+320*(v_cnt>>1)) %320)%80 + (((h_cnt>>1)+320*(v_cnt>>1))-240) /320)*80 ]+240;
562             end
563         end else if( (h_cnt>>1)+320*(v_cnt>>1) < 51200 ) begin
564             if( (((h_cnt>>1)+320*(v_cnt>>1)) %320) / 80 == 0 ) begin
565                 pixel_addr <= state_0[ ( ((h_cnt>>1)+320*(v_cnt>>1))%25600 %320)%80 + ( ((h_cnt>>1)+320*(v_cnt>>1))%25600 /320)*80 ]+25600;
566             end

```

[549] 如果 hold==1, 是為 hint mode, 所有圖片頭朝上(所以全部使用 state_0 array).
p.s. state_0、state_1、state_2、state_3 是上述講過的 4 個矩陣(上右下左).

以[563-566]行為例, 講述判斷的過程:

[563] $(h_cnt \gg 1) + 320 * (v_cnt \gg 1) < 51200$.

代表當前小格在 4 or 5 or 6 or 7 大格當中(也就是第二列的 4 大格其一).

[564] $((h_cnt \gg 1) + 320 * (v_cnt \gg 1)) \% 320 / 80 == 0$.

%320 代表橫向位移, 再/80==0, 代表位於這列的最左邊那大格(也就是 4).

[565] $pixel_addr \leq state_0[(((h_cnt \gg 1) + 320 * (v_cnt \gg 1)) \% 25600 \% 320) \% 80 + (((h_cnt \gg 1) + 320 * (v_cnt \gg 1)) \% 25600 / 320) * 80] + 25600$.

%25600 是先位移回第一列(現在是 4 所以會位移回 0).

%320 是橫向位移.

%80 是避免溢位, 因為一大格的橫向最多是 80.

/320 是在第幾列的意思

*80 是根據在第幾列, 意思是加上上面所有列(一列 80)

+25600 是位移回原本列(4)

若 hold == 0, 則根據每大格各自的 state, 使用 state_0~3, 算法都一樣(上述).

```

545 assign pass = (zero==0 && one==0 && two==0 && three==0 && four==0 && five==0 && six==0 && seven==0 && eight==0 && nine==0 && ten==0 && ele==0) ? 1'b1 : 1'b0;

```

如果 12 大格都是頭朝上, pass = 1.

```
685    assign {vgaRed, vgaGreen, vgaBlue} = (valid==1'b1) ? pixel : 12'h0;  
686    assign data = {vgaRed, vgaGreen, vgaBlue};  
687    blk_mem_gen_0 mem_gen(.addra(pixel_addr), .clka(clk_25MHz), .dina(data), .douta(pixel), .wea(0));
```


685 行, 決定 RGB 的值,

若 `valid==1`, `pixel`.

若 `valid==0`, `12'h0`.

2. 學到的東西與遇到的困難

一開始要理解 `h_cnt`、`v_cnt`、`pixel_addr` 之間的關係花了很長時間，還問了朋友兩次 XD。`lab7_1` 因為和課堂範例很像，所以基本上沒遇到甚麼問題。

`lab7_2` 一開始我不太理解到底是怎麼轉圖片(因為我一直以為那個小格也要轉)，後來在想方法的時候也是怎麼想都想不到。最後就想說直接存成大矩陣好了(用比較熟悉的 C++ 輔助)。而在換算 `addr` 的時候，因為過於複雜也是 debug 了很久(過程中那個圖片真的是一直亂閃= =)。

這次作業做完很有成就感，`demo` 那天也問了朋友是怎麼做的，感覺這題應該有很多種做法，為什麼覺得別人的方法都比較聰明阿哈哈哈哈哈 XDD。

3. 想對老師或助教說的話

謝謝老師&助教。