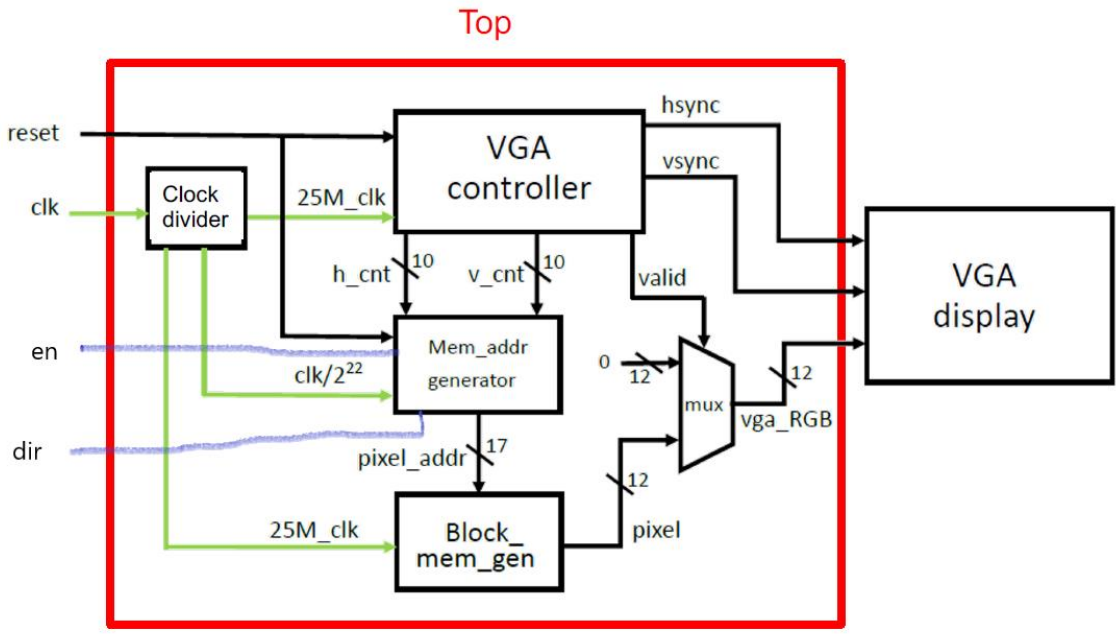
# Lab 7

學號: 109062318 姓名: 簡弘哲

1. 實作過程

7-1:




因為 7-1 跟上課講義裡的 demo2 差不多所以我就將 demo2 的 code 拿來做小修改，將 mem_addr_gen 這個 module 加上兩個 input signal en, dir 就完成了。

Scrolling:

```
if(rst) begin
    position<=0;
end else begin
    position<=position;
    if(en) begin
        if(!dir) begin  //scrolls up
            if(position<239) begin
                position<=position+1;
            end else begin
                position<=0;
            end
        end else begin  //scrolls down
            if(position>1) begin
                position<=position-1;
            end else begin
                position<=240;
            end
        end
    end
end
```

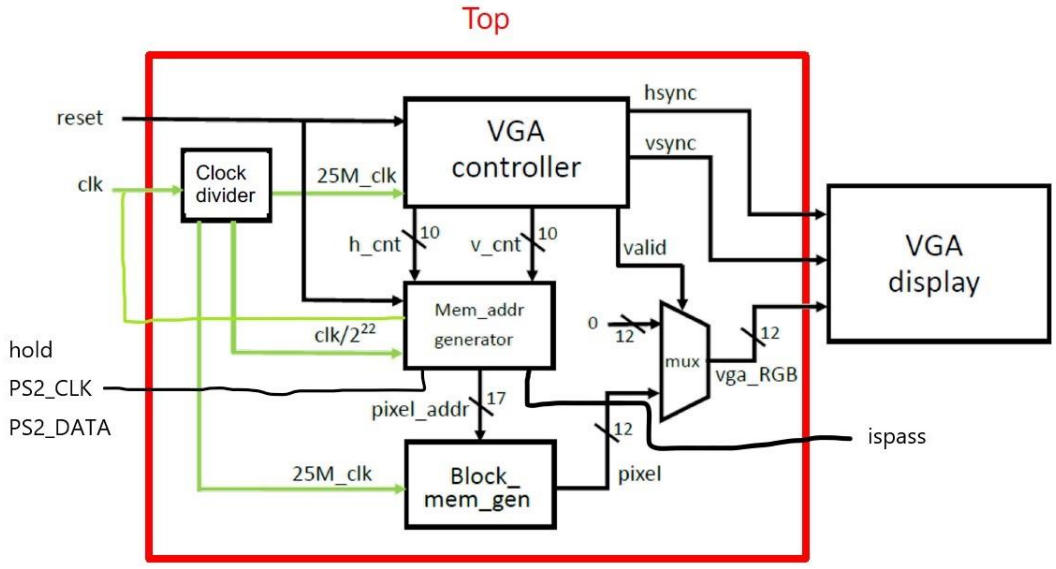
根據 dir 的值決定是往上還是往下，判斷 position 的值即可，僅須注意邊界 239,1。

Negative film:

```
//assign {vgaRed, vgaGreen, vgaBlue} = (valid==1'b
always @(*) begin
    if(valid) begin
        if(nf) begin
            {vgaRed, vgaGreen, vgaBlue} = ~pixel;
        end else begin
            {vgaRed, vgaGreen, vgaBlue} = pixel;
        end
    end else begin
        {vgaRed, vgaGreen, vgaBlue} = 12'h0;
    end
end
assign data = {vgaRed, vgaGreen, vgaBlue};
```

Negative film 非常簡單，用 bit complement(~)就可完成。

## 7-2:




從 7-1 修改而來，在 mem_addr_gen 加一些 input(hold, PS2_CLK, PS2_DATA), output(ispass)，因為我在 mem_addr_gen 裡有放 keyboardDecoder，所以需要傳入 PS2_CLK, PS2_DATA，檢查遊戲完成的機制也是實作在該 module 裡面，因此需要一個 ispass 的 output 信號。

變數意義解釋:

cnt 是一個長度為 12 的陣列，每個元素皆為 2bit，我定義 cnt[i]==0 就是不轉(0 度)、1 是順時針轉 90 度、2 是轉 180、3 是逆時針轉 90。

Index 0 記錄圖片左上角那格的方向

3 右上角

9 左下角

11 右下角

```
if(key_num==4'b0000) begin
    cnt_next[0]=cnt[0]-1;
    if(cnt[0]==0) begin  //0
        pixel_addr = ( 320*((h_cnt>>1)-0) + (79-(v_cnt>>1)) )%76800; //270
    end else if(cnt[0]==1) begin //90
        pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))%76800;    //0
    end else if(cnt[0]==2) begin //180
        pixel_addr = ( 320*(79-(h_cnt>>1)) + (0+v_cnt>>1) )%76800; //90
    end else if(cnt[0]==3) begin //270
        pixel_addr = ( 320*(79-(v_cnt>>1)) + (0+79-(h_cnt>>1)) )%76800; //180
    end else begin
        pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))%76800;
    end
```

上圖是實作逆時針旋轉的 code，以左上角那格為例子，每次逆時針轉 90 度，cnt[0]的值就-1，然後根據該格目前的擺放狀態決定下一個該呈現在螢幕上的角度公式，例如: 目前圖片是擺正的(cnt[0]==0)，那逆時針旋轉後就要變成 270 度，4 條公式都是預先算好的。

Hold, pass, 順時針旋轉:

```
if(hold || ispass) begin
    pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))%76800;
end else begin
    if(key_valid && key_down[last_change]) begin
        if(key_num!=4'b1111) begin
            if(!shift_down) begin
                if(key_num==4'b0000) begin
                    cnt_next[0]=cnt[0]+1;
                    if(cnt[0]==0) begin  //0
                        pixel_addr = ( 320*(79-(h_cnt>>1)) + (0+v_cnt>>1) )%76800; //90
                    end else if(cnt[0]==1) begin //90
                        pixel_addr = ( 320*(79-(v_cnt>>1)) + (0+79-(h_cnt>>1)) )%76800; //180
                    end else if(cnt[0]==2) begin //180
                        pixel_addr = ( 320*((h_cnt>>1)-0) + (79-(v_cnt>>1)) )%76800; //270
                    end else if(cnt[0]==3) begin //270
                        pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))%76800;    //0
                    end else begin
                        pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))%76800;
                    end
```

因為 hold 跟 pass 的效果都是顯示原圖，所以可以把他們寫在一起。Else 的部分則是遊戲還在進行中，以左上角那格為例，每次順時針旋轉 cnt[0]就+1，然後再根據目前圖片的狀態決定旋轉後該呈現在螢幕上的公式。

判斷遊戲結束:

```
if( cnt[0]==0 && cnt[1]==0 && cnt[2]==0 && cnt[3]==0 &&
    cnt[4]==0 && cnt[5]==0 && cnt[6]==0 && cnt[7]==0 &&
    cnt[8]==0 && cnt[9]==0 && cnt[10]==0 && cnt[11]==0) begin
    ispass_next=1;
end else begin
    ispass_next=0;
end
```

因為我怕用 loop + if 寫可能會造成 latch 的問題，所以我就直接列出 12 個 cnt 的值，當所有值都是 0，也就是每一格都是擺正的，才可以宣告遊戲結束。

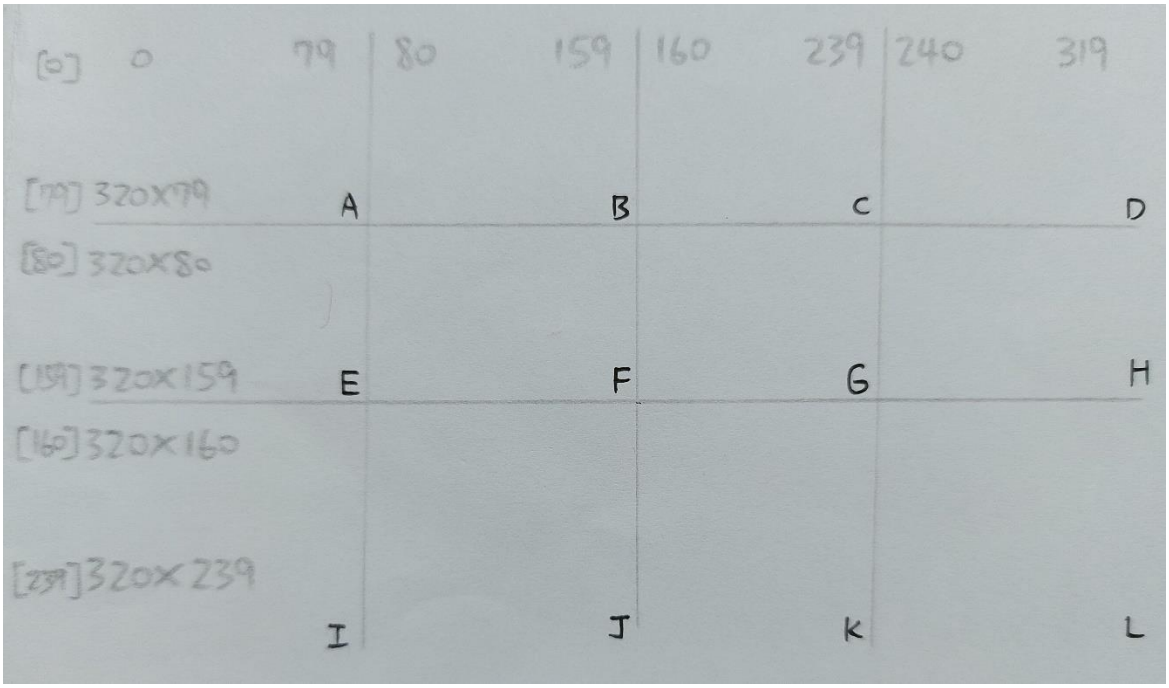
當沒有按鍵按下的時候:

```
if(0 <= h_cnt>>1 && h_cnt>>1 < 80 && 0 <= v_cnt>>1 && v_cnt>>1 < 80) begin  //o, m=0, M=79
    if(hold || ispass) begin
        pixel_addr = ( (h_cnt>>1)+320*(v_cnt>>1) )%76800;
    end else begin
        if(cnt[0]==0) begin  //0
            pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))%76800;    //0
        end else if(cnt[0]==1) begin //90
            pixel_addr = ( 320*(79-(h_cnt>>1)) + (0+v_cnt>>1) )%76800; //90
        end else if(cnt[0]==2) begin //180
            pixel_addr = ( 320*(79-(v_cnt>>1)) + (0+79-(h_cnt>>1)) )%76800; //180
        end else if(cnt[0]==3) begin //270
            pixel_addr = ( 320*((h_cnt>>1)-0) + (79-(v_cnt>>1)) )%76800; //270
        end else begin
            pixel_addr = ((h_cnt>>1)+320*(v_cnt>>1))%76800;
        end
    end
```

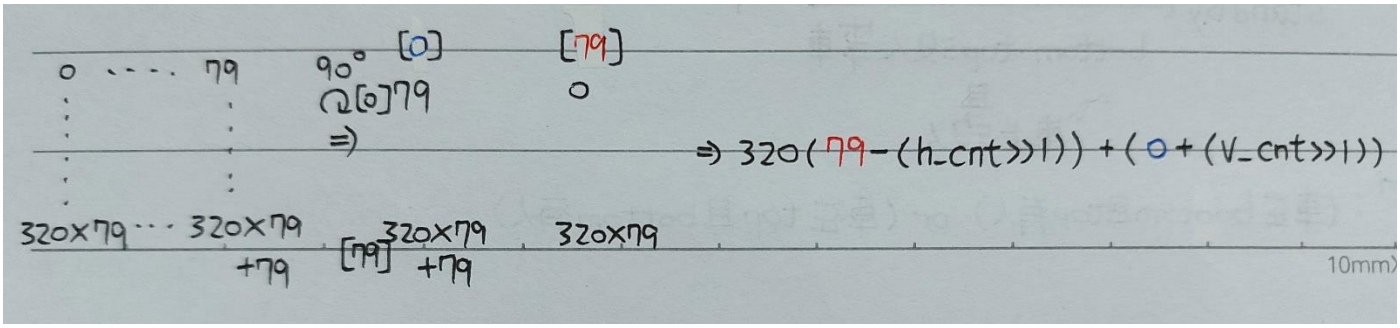
根據 v_cnt 與 h_cnt 的所在位置判斷目前 vga.v 在畫哪一格的 pixel，以左上角那格為例子，如果按下 hold 或是遊戲結束就顯示原圖，否則的話就看該格目前的狀態並將公式寫進去，像是該格目前是順時針 90 度(cnt[0]==1)，那就將 90 度的公式套進去。

算公式:

因為照片的二維資料是存在一維陣列裡面的，所以要推算該去一維陣列的哪個 index 取出正確的資料。一張照片可以分成 12 格，如下圖所示，然後我將每一格分別命名為 A 到 L，先寫出邊上 pixel 在一維陣列裡面的 index 值以方便計算。



以左上角那格為例，它原本四個角的 index 分別為 0, 79, 320*79, 320*79+79，將該格順時針旋轉 90 度後如下圖所示，四個角所在的 index 會改變，然後將旋轉過後的 index 分布用 column 的 index(0,79)去表達出公式，這需要一些時間觀察，最後得出右邊的那條式子，求其他格也是用類似的方法。



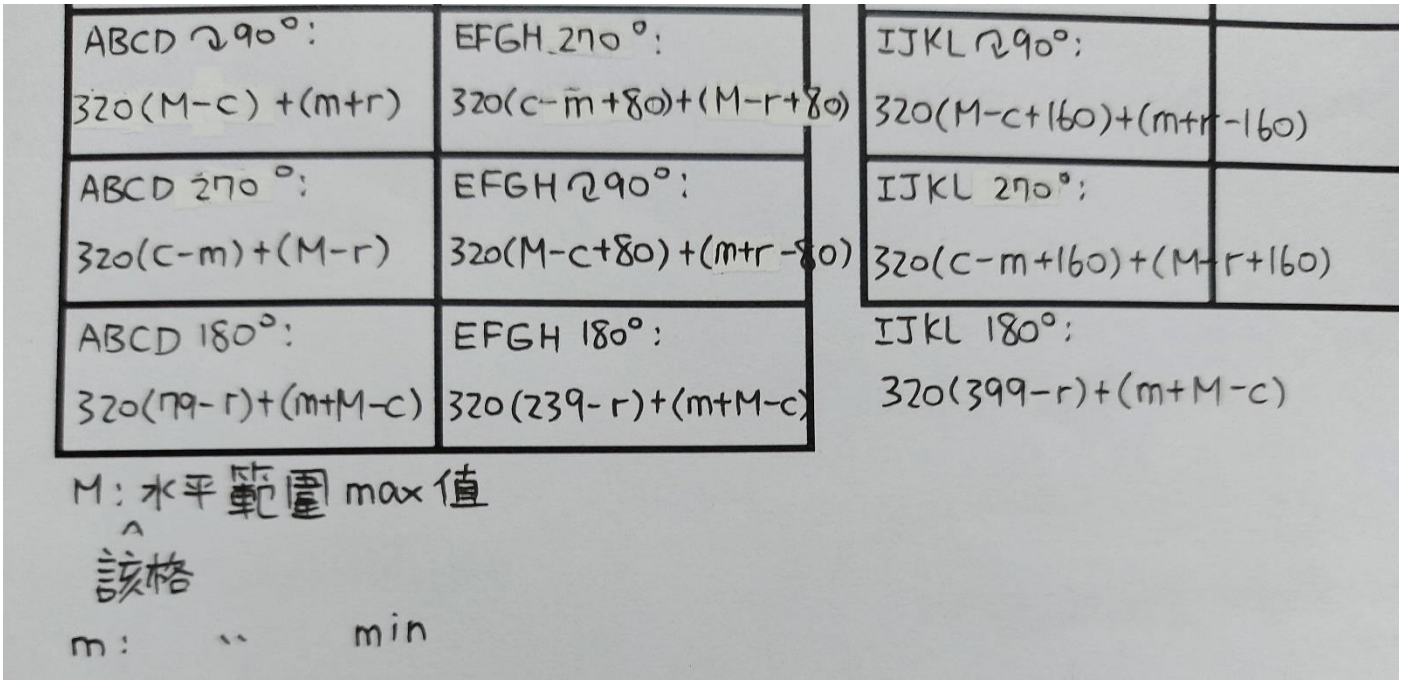

到最後歸納出 12 格在旋轉不同的角度(順時針 90,180,270)後推算出 pixel address 的公式，

其中 c 是(h_cnt>>1)、r 是(v_cnt>>1)

m 是該格水平範圍的較小值、M 是該格水平範圍的較大值

以 L 格(圖片右下角)來說，該格的水平範圍是 240~319，因此 m=240,M=319 帶入公式即可。

因為我觀察到在分成 12 格的圖片裡面，位於同一行的格子(ex.ABCD、EFGH、IJKL)他們的旋轉公式都一樣，例如第一行(ABCD)順時針旋轉 90 度的公式皆為 320(M-c)+(m+r)，再套入該格水平範圍的較大、較小值就可得出該格的公式。




2. 學到的東西與遇到的困難

1. 寫 7-2 時一開始完全不知道怎麼下手，漸漸學會如何算 pixel

一開始完全無法理解圖片要怎麼旋轉，所以就往修改 vga.v 的方向思考，想說應該不再是從左到右、從上到下的掃 pixel，但後來覺得要是圖片轉很亂的話根本就不知道下一個該掃的 pixel

在哪邊，所以就認為應該是修改 mem_addr_gen，當圖片旋轉時就去推算該出現的 pixel address 是多少。但我稍微想了想，有 12 格圖片，除了 0 度以外不用算，剩下的 3 個方向都要算，所以總共要算 3*12=36 次，這數字讓我覺得有點多，可能還有更好的方法。但再想了一段時間後還是覺得只能乖乖算 pixel address，別無他法了。幸好算了一些方向後有發現一些規律才不用 36 種都算，每次算的時候將翻轉過後 pixel 分布用 column 的 index 表示。

2. [7-2] latch 造成 reset 有問題

   一開始寫 reset 的時候永遠只能 reset 到最左上角的那格，其他格就重設不到，會維持原來的樣子，這困擾了我好一陣子，我一直在 if(rst)那邊打轉但終究不見起色，直到我注意到 vivado 的 warning 說有 infer latch，把它改掉之後一切就正常了。

3. [7-2] 的鍵盤不靈敏

   我仿造 lab6 的寫法，將鍵盤 code 的部分搬了過來，但沒想到在測試的時候鍵盤竟然是那麼的不靈敏，lab6 都沒有這種問題。這個問題也困擾了我許久，一直盯著 code 也不知道該改哪邊，正當我絕望之際，我腦中突然閃過了一個想法，”把這塊 code block 搬到別的地方去會怎樣?”，結果一實驗發現鍵盤就可以正常運作了。看來有時候寫不出來就把 code block 換個位置，說不定就會正常了。

   (因為我一開始的寫法是

   If(h_cnt>>1, v_cnt>>1 的值落在左上角那格){

   處理鍵盤的 code…

   }

   後來將它改成=>

   處理鍵盤的 code…(搬到 if 外面)

   If(h_cnt>>1, v_cnt>>1 的值落在左上角那格){

   其他的 code…

   }

   就正常了，看起來是處理鍵盤的 code 寫外面會比較好

   )

3. 想對老師或助教說的話

   在 verilog 裡的 for loop 有 break 這個關鍵字嗎?