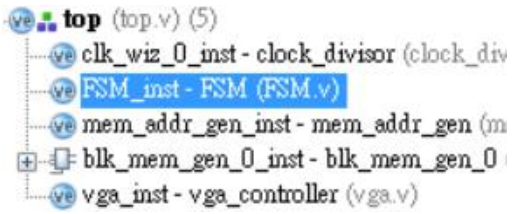
106061146_陳兆廷_lab11_report

1. 用 en 來控制圖片的滾動

依照 ilms 上面公布的 demo code，加上 memory IP 的設定，已經可以輸出完整的圖片。因此現在需要改變的是增加 enable 選項，及改變 mem_addr_gen.v 中的 position，因此只要修改 demo 中的 position 選項，當 enable 時 position 才會增加，disable 時 position 維持原狀，就會停止滾動。但要依照題目要求，按一下滾動，第二下停止的功能，需要 lab05 的 FSM，如圖(一)，因此新增兩個 state，經過 FSM 的 enable 會輸出當下要開始捲動或停止捲動的 enable，如圖(二)。如此即可完成第一題。



圖(一)

```
FSM FSM_inst(.in(en), .clk(clk_22), .rst_enable(en_en));
```

圖(二)

2. 利用鍵盤，在螢幕上寫出兩位數的計算機

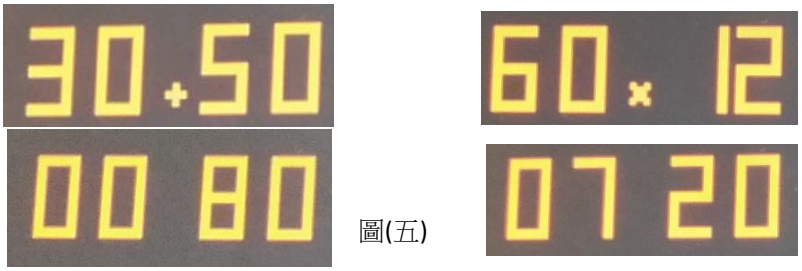
這題我利用 demo1 中的 code，就是在螢幕上作畫的原理。先利用 lab09 寫好的二位數計算機，keyboard.v 會輸入 USB_DATA、USB_CLK，輸出會輸出四個數字及一個加減乘除的代號，前兩位及後兩位數字分別是各兩位的數字，按下 enter 後會變成解。因此我在 pixel_gen.v 中，新增輸入值四個數字及一個符號，如圖(三)，並開始依照 num、num1、num2、num3 開始在螢幕上作畫，依照 h_cnt 及 v_cnt 的位置，更改 vgaRed 變成紅色，畫成 0、1、2 等等的形狀，如圖(四)，再將畫好的數字分別往右移(h_cnt 增加)，就可以放在右邊。

```
module pixel_gen(
    input [9:0] h_cnt, //640
    input [9:0] v_cnt, //480
    input [3:0]num,
    input [3:0]num1,
    input [3:0]num2,
    input [3:0]num3,
    input [2:0]mark,
    input valid,
    output reg [3:0] vgaRed,
    output reg [3:0] vgaGreen,
    output reg [3:0] vgaBlue
    );
```

圖(三)

```
case(num)
        4'd0: begin
        shift = 10'd10;
           if(((h_cnt >= (shift + 10'd10))&&(h_cnt <= (shift + 10'd22))&&(v_cnt >= 10'd10)&&(v_cnt <= 10'd12))||
              ((h_cnt >= (shift + 10'd10))&&(h_cnt <= (shift + 10'd22))&&(v_cnt >= 10'd30)&&(v_cnt <= 10'd32))||
              ((h_cnt >= (shift + 10'd20))&&(h_cnt <= (shift + 10'd22))&&(v_cnt >= 10'd10)&&(v_cnt <= 10'd32))||
              ((h_cnt >= (shift + 10'd10))&&(h_cnt <= (shift + 10'd12))&&(v_cnt >= 10'd10)&&(v_cnt <= 10'd32))) begin
               {vgaRed, vgaGreen, vgaBlue} = 12'hf00;
           end
        end
        4'd1: begin
        shift = 10'd10;
           if(((h_cnt >= (shift + 10'd20))&&(h_cnt <= (shift + 10'd22))&&(v_cnt >= 10'd10)&&(v_cnt <= 10'd32))) begin
               {vgaRed, vgaGreen, vgaBlue} = 12'hf00;
           end
        end
```

圖(四)

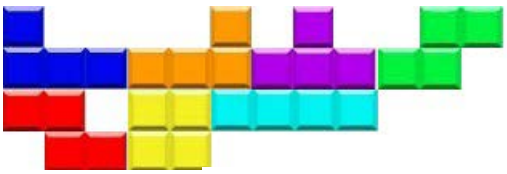


圖(五)

在作圖的時候，原本想寫當一個 module 然後用四次，鍵入要輸出的座標即可，但好像沒那麼順利，vga 輸出總是會出現錯誤，因此還是寫在同一個大 module 裡面。

3. 不斷掉落的俄羅斯方塊

在這題需要用到 demo2 的 memory IP 及往下捲動的方法，因此需要將俄羅斯方塊的.jpg 轉成 coe 檔並輸入。但並不是要將所有的磚塊一次向下移，因此在 mem_addr_gen.v 需要修改。因為要無限捲動，因此 position 要在 posedge clk 上從 0 至 239 無限循環(320*240)，接著便是要如何讀取不同方塊。總共有七種方塊，如圖(六)，因此要知道圖片中的磚塊在甚麼地方，並用 h_cnt、v_cnt 及磚塊的位置來產出 pixel_addr。如圖(七)。



圖(六)

```
always @* begin
    if(bricktype_tmp == 0) begin
        if(h_cnt>=0 && h_cnt<78 && v_cnt>=position && v_cnt<(position+52))
            pixel_addr = h_cnt + (v_cnt-position) * 320;
        else
            pixel_addr = 30;
    end
    else if(bricktype_tmp == 1) begin
        if(h_cnt>=0 && h_cnt<78 && v_cnt>=position && v_cnt<(position+52))
            pixel_addr = 78 + h_cnt + (v_cnt-position) * 320;
        else
            pixel_addr = 30;
    end
    else if(bricktype_tmp == 4) begin
        if(h_cnt>=0 && h_cnt<78 && v_cnt>=position && v_cnt<(position+52))
            pixel_addr = 16640 + h_cnt + (v_cnt-position) * 320;
        else
            pixel_addr = 30;
    end
    else if(bricktype_tmp == 5) begin
        if(h_cnt>=0 && h_cnt<52 && v_cnt>=position && v_cnt<(position+52))
            pixel_addr = 16718 + h_cnt + (v_cnt-position) * 320;
        else
            pixel_addr = 30;
    end
```

在螢幕上寬度為 78(3 格)

在螢幕上長度為 52(2 格)+向下行數

從 0 開始讀，因此是左上的藍色方塊

在螢幕上寬度為 78(3 格)

在螢幕上長度為 52(2 格)+向下行數

從 78(3 格)開始讀，因此是第二個，橘色方

在螢幕上寬度為 78(3 格)

在螢幕上長度為 52(2 格)+向下行數

從 320*52(第二行)開始讀，因此是左下的紅色方塊

在螢幕上寬度為 52(2 格)

在螢幕上長度為 52(2 格)+向下行數

從 320*52+78 開始讀，因此是 2*2 的黃色方塊

圖(七)