Lab 6 Peripheral Components

實驗報告

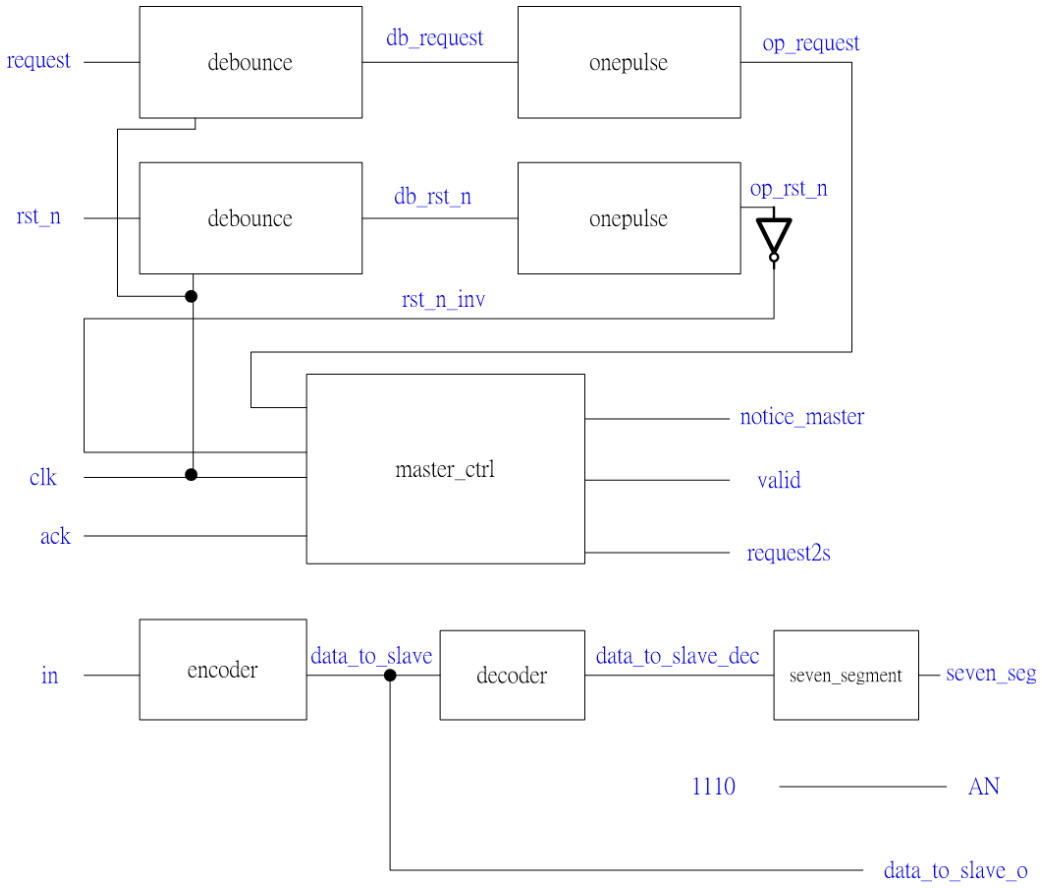
組長:劉奇泓 109033135

組員:洪聖祥 109062315

1. Dual FPGA communication

I. master:

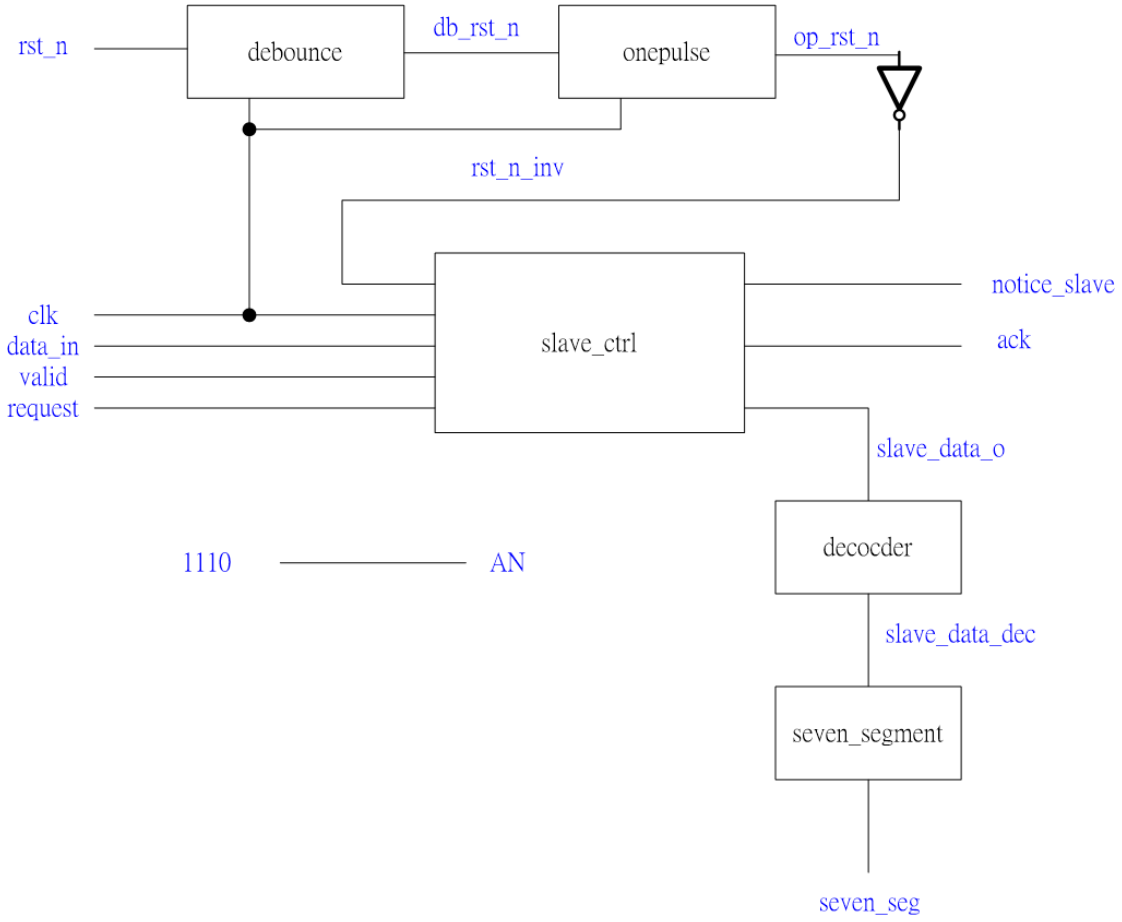
Block diagram:



Dual FPGA 在 master 的部分會根據 switch 輸入的 in 以及 button 的 request、rst_n，經由 master_ctrl 輸出 notice_master 顯示在 LED，以及輸出 data_to_slave_o、valid、request2s 到 slave，並將 in 經由 encoder、decoder、seven_segment 由七段顯示器顯示 switch 輸入的數字。

II. slave:

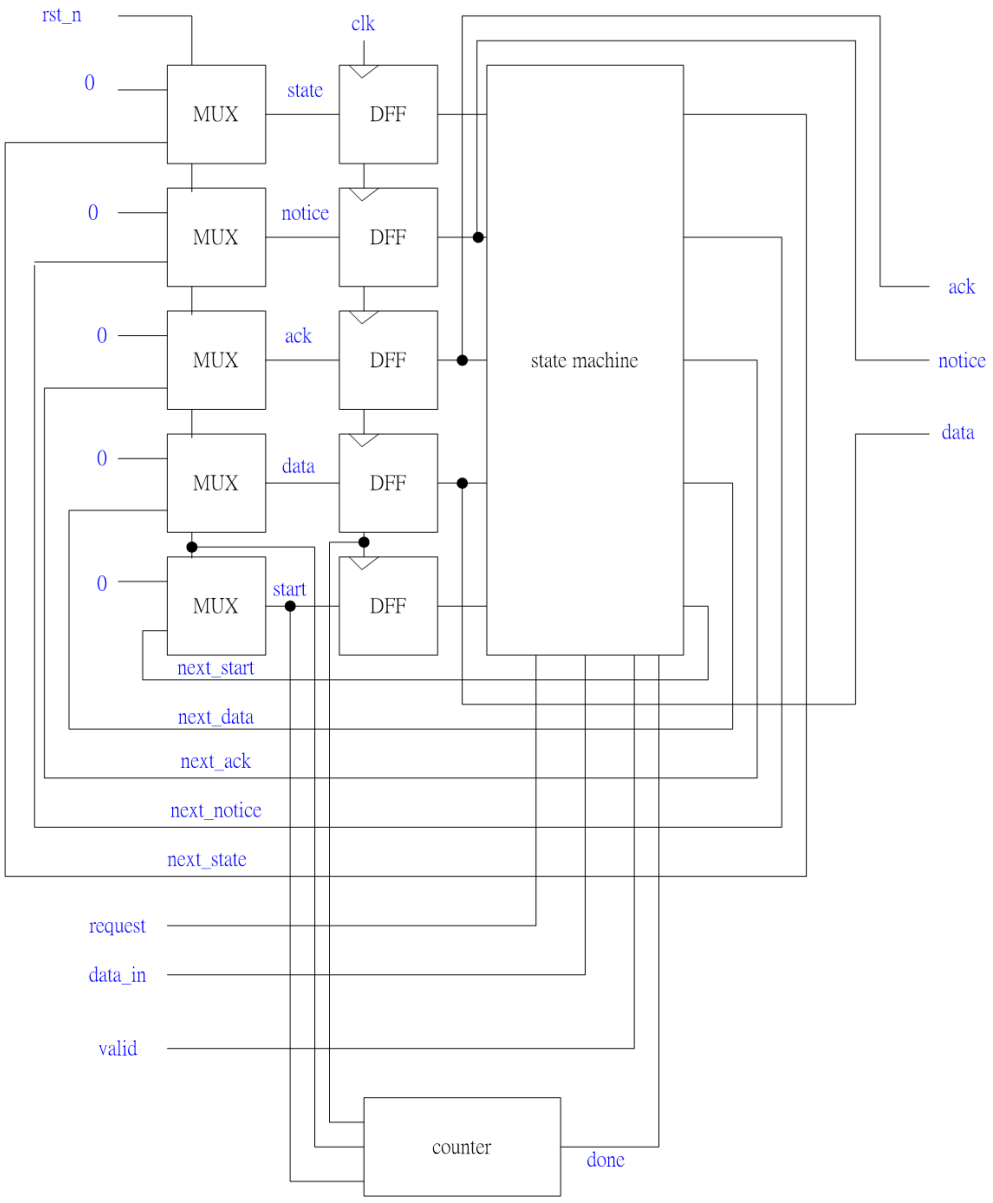
block diagram:



Slave 的運作與 master 相似，會接收 master 輸出的 data_in、valid、request，經由 slave_ctrl 輸出 notice_slave、ack 回傳給 master，並把 slave 收到的 data 由 decoder 和 seven_segment 輸出到七段顯示器顯示出來。這次我們主要設計的 module 是 slave_ctrl，因此再對此 module 進行詳細說明

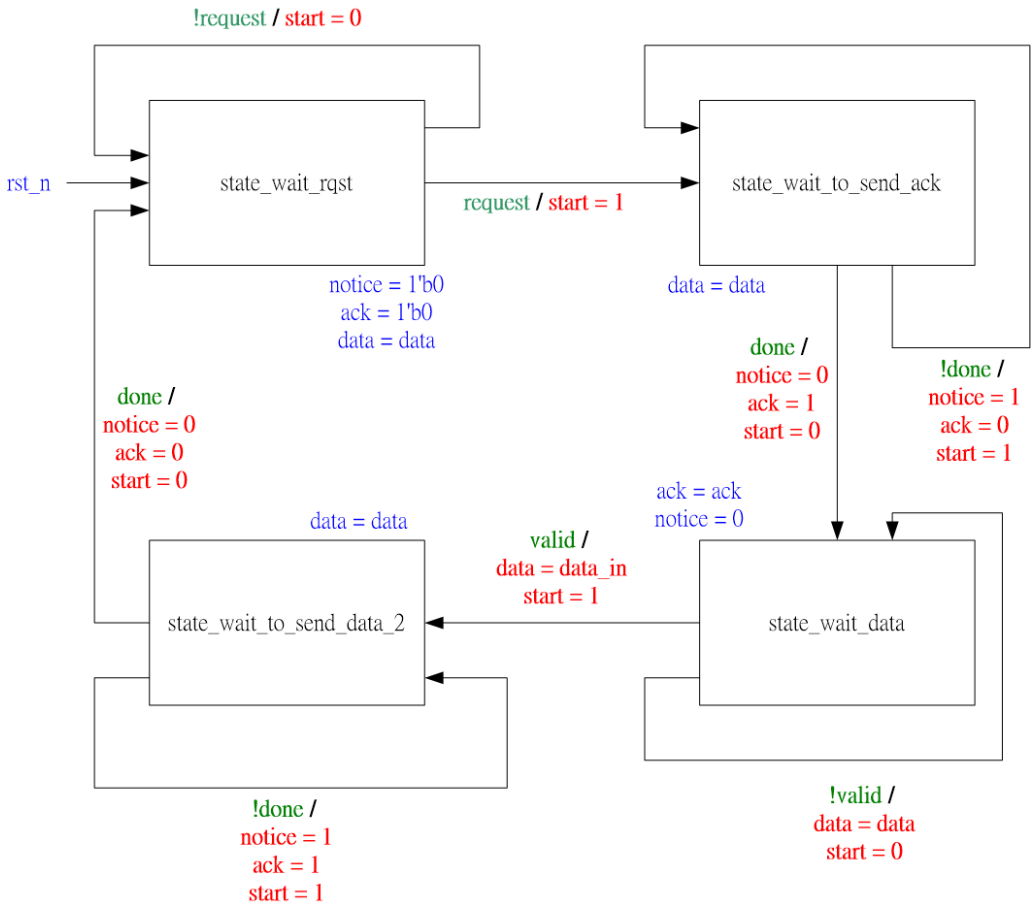
- slave_ctrl:

block diagram:



Slave_ctrl 中有來自 master 的 input request、data_in、valid，並用一個 5 個 MUX 和 DFF 來 reset state、notice、ack、data、start 的訊號並隨著 posedge clk 改變，然後把這些訊號傳到 state machine 中得到下個 cycle 的訊號，其中 ack 會傳回給 master，notice 會由 LED 亮起顯示，data 會由七段顯示器顯示出數字。另外還有一個 counter 可以計算 notice 拉起的時間。

state diagram:




```
always@(*) begin
    next_state = state;
    next_notice = notice;
    next_ack = ack;
    next_data = data;
    next_start = start;
    case(state)
        state_wait_rqst: begin
            next_state = (request)? state_wait_to_send_ack: state_wait_rqst;
            next_notice = 1'b0;
            next_ack = 1'b0;
            next_data = data;
            next_start = (request)? 1'b1: 1'b0;
        end
        state_wait_to_send_ack: begin
            next_state = (done)? state_wait_data : state_wait_to_send_ack;
            next_notice = (done)? 1'b0: 1'b1;
            next_ack = (done)? 1'b1: 1'b0;
            next_data = data;
            next_start = (done) ? 1'b0 : 1'b1;
        end
        state_wait_data: begin
            next_state = (valid)? state_wait_to_send_ack_2 : state_wait_data;
            next_notice = 1'b0;
            next_ack = ack;
            next_data = (valid)? data_in : data;
            next_start = (valid)? 1'b1: 1'b0;
        end
        state_wait_to_send_ack_2: begin
            next_state = (done)? state_wait_rqst : state_wait_to_send_ack_2;
            next_notice = (done)? 1'b0: 1'b1;
            next_ack = (done)? 1'b0: 1'b1;
            next_data = data;
            next_start = (done) ? 1'b0 : 1'b1;
        end
        default: begin
        end
    endcase
```

reset 時 state 會到 state_wait_rqst，這時 notice、ack = 0，data 不變，start 則會隨著 request 的訊號改變，start = 1 時會

讓 counter 開始計時 1 sec，request = 1 時 state 會變成 state_wait_to_send_ack。

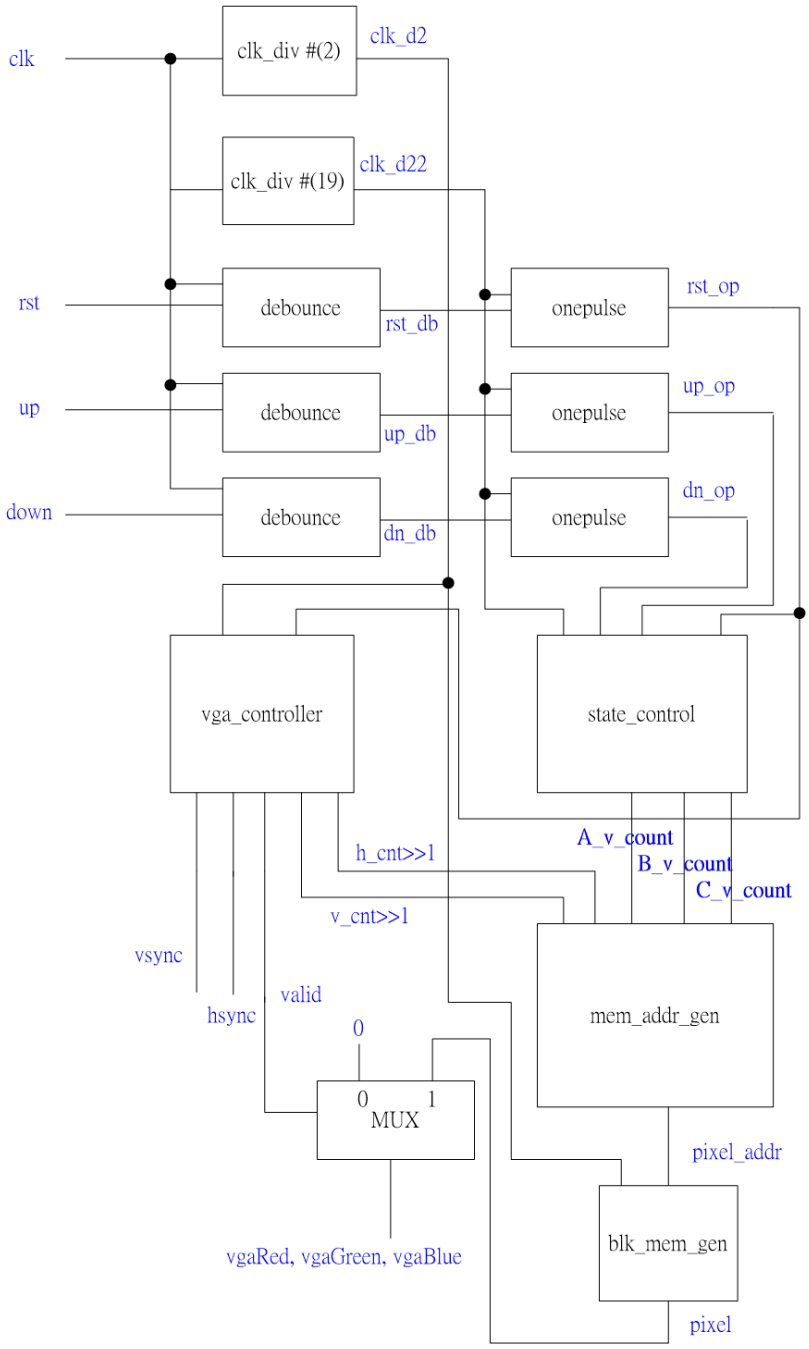
state = state_wait_to_send_ack 時 notice = 1 讓 LED 燈保持亮起，start 持續為 1，如果 done = 1 的話表示 counter 已經跑完 1sec 了，notice 和 start 會拉下來，並將 ack 訊號傳回到 master，state 變成 state_wait_data。

state = state_wait_data 時會等待 master 傳來 valid 訊號，如果收到了就讓 start=1 再次開始在 counter 計時 1sec，並把 state 變成 state_wait_to_send_ack_2。

state = state_wait_to_send_ack_2 時 notice = 1 讓 LED 燈保持亮起，ack、start 持續為 1，如果 done = 1 的話表示 counter 已經跑完 1sec 了，notice 和 start 會拉下來，並將 ack = 0 傳回到 master，state 變成 state_wait_rqst。

2. The slot machine

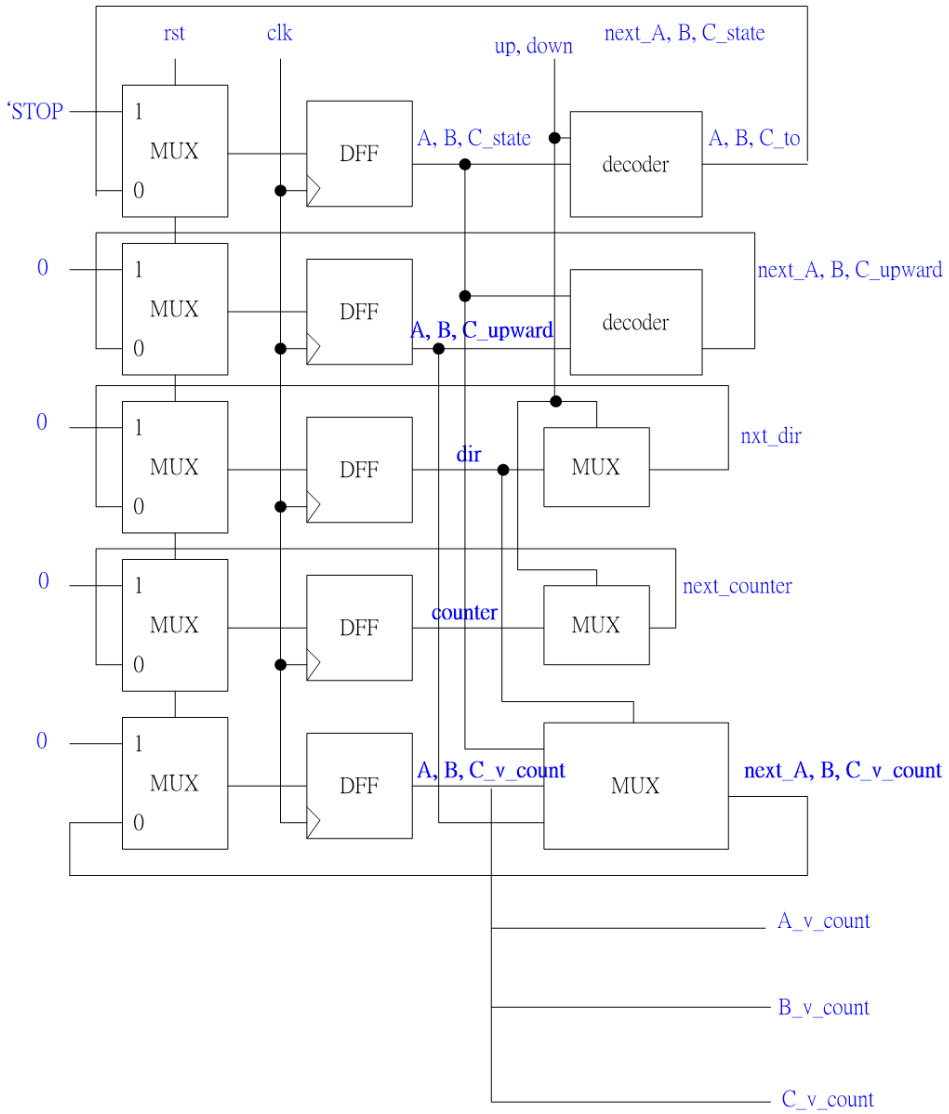
Block diagram:



Slot machine 有 up、down、rst 三種按鍵，並連接到 state_control 中得到三個圖片移動的方式，再用其他的 module 來得到 output 訊號 vgaRed、vgaGreen、vgaBlue、vsync、hsync，用 vga 傳輸線顯示到螢幕上。這個題目我們主要設計的 module 是 state_control，因此以下會再做詳細說明。

- State_control

Block diagram:



在 state_control 中，我們增加了 dir 可以分辨現在進行的是 up 或是 down，

```
always @ (*) begin
    if(up)
        nxt_dir = 1'b1;
    else if(down)
        nxt_dir = 1'b0;
    else
        nxt_dir = dir;
    end
```

並且新增了 A_upward、B_upward、C_upward 表示圖形要往上移動時圖片的位移，在超過 239 後會回到 0 繼續往上加，

```
always @ (*) begin
    if(A_upward >= 10'd240)
        nxt_A_upward = A_upward + A_state - 10'd240;
    else
        nxt_A_upward = A_upward + A_state;
    if(B_upward >= 10'd240)
        nxt_B_upward = B_upward + B_state - 10'd240;
    else
        nxt_B_upward = B_upward + B_state;
    if(C_upward >= 10'd240)
        nxt_C_upward = C_upward + C_state - 10'd240;
    else
        nxt_C_upward = C_upward + C_state;
  end
```

另外我們改變了計算 counter 的 combinational circuit，讓 up、down 按下時都能使 counter 開始計數，

```
always @ (*) begin
    if((up|down) == 1'b0) begin
        if(counter == 10'd0)
            next_counter = counter;
        else if(counter == 10'd1000)
            next_counter = 10'd0;
        else
            next_counter = counter + 1'b1;
        end
    else
        next_counter = counter + 1'b1;
    end
```

我們也改變了 A_v_count、B_v_count、C_v_count 的 combinational circuit，dir = 1 時表示圖片要往上滾動，因此我們把 239 - upward 就可以讓圖片的動作與本來往下滾動的動作相反，就可以實現向上滾動了。

```
always @ (*) begin
    if(dir) begin
        next_A_v_count = 10'd239 - A_upward;
        next_B_v_count = 10'd239 - B_upward;
        next_C_v_count = 10'd239 - C_upward;
        end
    else begin
        next_A_v_count = (A_v_count + A_state >= 10'd240)? A_v_count + A_state - 10'd240: A_v_count + A_state;
        next_B_v_count = (B_v_count + B_state >= 10'd240)? B_v_count + B_state - 10'd240: B_v_count + B_state;
        next_C_v_count = (C_v_count + C_state >= 10'd240)? C_v_count + C_state - 10'd240: C_v_count + C_state;
    end
    end
```

7. 心得

劉奇泓:這次 lab 其中一個題目是要讓兩個 fpga 互動，一開始有了 sample code 的幫助我很快就把 slave 的程式碼完成了，在測試 master 的 input 的時候本來也一帆風順，但是我們在輸入 0 時在 slave 卻得到不穩定的訊號，認真追蹤過 decoder 和 encoder 的 module 也找不出來錯誤的地方，還好助教從寬讓我們通過這題了，我推測應該是因為 pmod 的訊號本來就不太穩定，可能在輸入 0 的時候傳輸出現了問題。另一題是做 slot machine，我照著 sample code 花了一些時間才理解怎麼讓圖片呈現滾動的畫面，並做出了往相反方向轉動的 combinational circuit，這一題相對比較順利一點。終於完成最後一個 lab 了，但是還有更困難的 final project，希望在這六個 lab 中學到的技能可以讓我順利的完成 project!

8. 分工

洪聖祥: advanced question 3

劉奇泓: advanced question 1、2