EECS 2070 02 Digital Design Labs 2019 Lab 4
學號：107062107　姓名：王領崧

1.前言

這次的 lab 加入了 FPGA 板子上面的 7 segment 顯示器，分成三個作業，第一個是利用 16 個 switch，4 個一組來控制數字的顯示。第二個則為基本的 counter，加入 enable direction 等等參數來控制。第三個則為倒數計時的功能，需有設定模式和倒數模式，Bonus 的部份是要將時間調整為準確的 1 秒。

2.實作過程

lab4_1

lab4_1 拆成三個部分實作，顯示的位置、要顯示的數值、assign 數值。

顯示的位置是利用 FBGA 板子上 DIGIT 參數的控制，由於一次只能顯示其中一個 DIGIT，因此利用短時間的循環，造成視覺佔留的效果，看起來好像 4 個 DIGIT 同時在亮。我的方法是採用講義上的模式，利用 case 來判斷 0(顯示的位置)的位置，同時給予 value 值，並且讓 digit 的 0 往左移一位。

顯示的數值則是依據不同的 value 值，給予 7 segment(Display) 正確的亮暗值，另外，如果 value 值>9 的話，則都是顯示數字 9。

assign 數值則是直接將 switch，4 個一組，每組 4 個 bit 直接對應 value 的 4 個 bit，即可完成。

```
always @(posedge clk_div13) begin
    case (digit)
        4'b1110: begin
        value = (reset == 1) ? 4'd0 : BCD1;
        digit = 4'b1101;
        end
        4'b1101: begin
        value = (reset == 1) ? 4'd0 : BCD2;
        digit = 4'b1011;
        end
        4'b1011: begin
        value = (reset == 1) ? 4'd0 : BCD3;
        digit = 4'b0111;
        end
        4'b0111: begin
        value = (reset == 1) ? 4'd0 : BCD0;
        digit = 4'b1110;
        end
        default: begin
        value = (reset == 1) ? 4'd0 : BCD0;
        digit = 4'b1110;
        end
    endcase
end
```

```
always @* begin
    case (value)
        4'd0: display = 7'b1000000;
        4'd1: display = 7'b1111001;
        4'd2: display = 7'b0100100;
        4'd3: display = 7'b0110000;
        4'd4: display = 7'b0011001;
        4'd5: display = 7'b0010010;
        4'd6: display = 7'b0000010;
        4'd7: display = 7'b1111000;
        4'd8: display = 7'b0000000;
        4'd9: display = 7'b0010000;
        default: display = 7'b0010000;
    endcase
end

assign BCD0 = SW[3:0];
assign BCD1 = SW[7:4];
assign BCD2 = SW[11:8];
assign BCD3 = SW[15:12];
assign DIGIT = digit;
assign DISPLAY = display;
```

lab4_2

lab4_2 大致可分成三個部分，counter 數值、record 數值、參數 state 的切換。

counter 數值分成兩個 always block，一個是 sequential logic，在指定的 clk23 trigger 下，如果是 reset 則進行重置，否則接著判斷如果是 enable 的狀態，則等於 next_digit，disable 則 digit = digit。另一個則是 combinational logic，如果 dir 是累加的話，則先判斷是否已到 99，否則+1，如果 dir 是遞減的話，則判斷是否為 0，不是則-1。

record 數值的部分僅使用一個 always block，一樣先判斷是否需要 reset 重置，接著當 record 按鈕為 1 的時候，record 的數值則等於現在 counter 的數值，反之則等於自己。

參數 state 的切換則是宣告了兩個 wire 代表 enable 和 direction 的狀態，當 reset 的時候，先初始化為 disable 和累加的狀態，而當 en 按鈕按下去時，enable state 則做一個 not 的動作，dir 按鈕按下去時，direction 同理也執行 not 的動作。

max 和 min 則是利用簡單的 assign 方式，如果 99 則 max=1，如果 0 則 min=1。

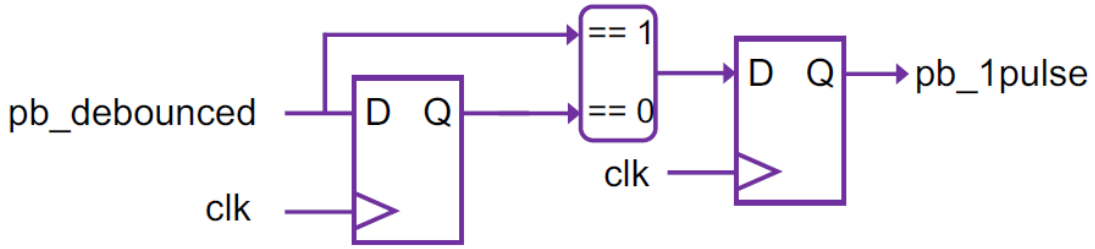
按鈕的部分會經過 debounced 和 one_pulse 兩階段。debounced 是為了使按鈕的訊號能夠穩定，不會因為彈簧的回彈而產生雜訊。one_pulse 則是使得不論按鈕按的時間長短，皆只會產生一個脈衝波而已，兩者的程式碼都是直接參考上課的講義。

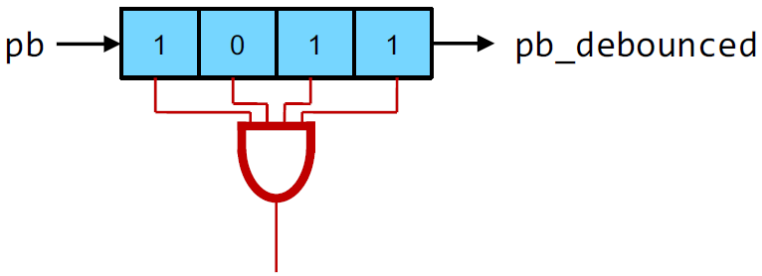
```
always @(posedge clk_div25) begin
    if (direction == 1) begin
        next_digit1 = (digit2 == 4'd9 && digit1 != 4'd9) ? digit1 + 1 : digit1;
        next_digit2 = (digit2 == 4'd9) ? ((digit1 == 4'd9) ? digit2 : 4'b0000) : digit2 + 1;
    end
    else begin
        next_digit1 = (digit2 == 4'd0 && digit1 != 4'd0) ? digit1 - 1 : digit1;
        next_digit2 = (digit2 == 4'd0) ? ((digit1 == 4'd0) ? digit2 : 4'd9) : digit2 - 1;
    end
end

assign max = (digit1 == 4'd9 && digit2 == 4'd9 && direction == 1) ? 1'b1 : 1'b0;
assign min = (digit1 == 4'd0 && digit2 == 4'd0 && direction == 0) ? 1'b1 : 1'b0;
```

```
always @(posedge clk_div16) begin
    if (reset == 1) begin
        direction = 1'b1;
        enable = 1'b0;
    end
    else begin
        direction = (direc == 1) ? ~direction : direction;
        enable = (ena == 1) ? ~enable : enable;
    end
end
```

```
always@(posedge clk_div23) begin
    if (reset == 1) begin
        digit1 = 4'b0000;
        digit2 = 4'b0000;
    end
    else begin
        if (enable == 1) begin
            digit1 = next_digit1;
            digit2 = next_digit2;
        end
        else begin
            digit1 = digit1;
            digit2 = digit2;
        end
```

```
always @(posedge clk_div16) begin
    if (reset == 1) begin
        rdigit1 = 4'b0000;
        rdigit2 = 4'b0000;
    end
    else begin
        if (rec == 1) begin
            rdigit1 = digit1;
            rdigit2 = digit2;
        end
        else begin
            rdigit1 = rdigit1;
            rdigit2 = rdigit2;
```





lab4_3 & lab4_bonus

lab4_3 分為三個部分，設定與計時，參數 state 的切換，bonus 的部分，每一個按鈕都有先經過 debounced 和 one_pulse 的處理。

參數 state 的切換與 lab4_2 同理，這邊只有一個 en 的按鈕，因此只有新增一個 enable 的 wire，當 reset 和設定模式則為 disable，反之則 en 按鈕按下去時，enable state 做一個 not 的動作。

設定與計時的部分，只有使用一個 always block 來運作，先判斷是否需要 reset，接著分為設定模式和倒數模式。設定模式下，如果按的是分鐘按鈕，則 minute+1，需要注意的是用 60 作為一個循環，因此如果 minute 為 59 時，則會變回 00，如果按的是秒鐘按鈕，則 second+1，與分鐘相同，使用 60 為單位做循環，多了當 second 進位時，minute 也要做 +1 的動作，mode 部分有做一個 negedge_one_pulse 的處理，使得當 mode 為 0，產生出一個 negedge 的脈衝波，用來當 mode 切換時，進行類似 reset 的功能。倒數模式下，如果是 enable，則進行 second-1，minute 的部分則是判斷當 second 從 00 變成 59 時，才要 minute-1，否則維持原本的值，當倒數到 0000 時才停止，反之如果為 disable，則 hold 住原本的值。FPGA 板上的每個數字則是利用分鐘秒鐘的除法與取餘數來賦值。

Bonus 的部分，則是多做一個 1HZ counter module，因為原本的 clk 是 100MHz，所以我做了一個加至 49999999 的 counter，當 cnt 加至 49999999 時，進行一個歸零的動作，並且讓 clk_1HZ = ~clk_1HZ，反之則 cnt 進行累加，clk_1HZ = clk_1HZ，這樣就可以產生以 50M 為循環的 1HZ clk。

```
always @(posedge clk_div16) begin
    if (reset == 1) begin
        enable = 1'b0;
    end
    else begin
        if (mode == 0) begin
            enable = 1'b0;
        end
        else begin
            enable = (en_1pulse == 1) ? ~enable : enable;
        end
    end
end
```

```
module clock_divider_1HZ(clk, clk_div);
    parameter n = 27;

    input clk;
    output clk_div;
    reg [n-1:0]cnt;
    reg clk_div;
    wire dir;
    always@(posedge clk) begin
        if(cnt == 49999999) begin
            cnt <= 0;
            clk_div <= ~clk_div;
        end
        else begin
            cnt <= cnt + 1;
            clk_div <= clk_div;
        end
    end
endmodule
```

```
if (mode == 0) begin
    if (mode_1pulse == 0) begin
        minute = 7'd0;
        second = 7'd0;
    end
    else if (min_plus_1pulse == 1) begin
        minute = (minute == 7'd59) ? 7'd0 : minute + 1;
        second = second;
    end

    else if (sec_plus_1pulse == 1) begin
        if (second == 7'd59) begin
            if (minute == 7'd59) begin
                minute = 7'd0;
                second = 7'd0;
            end
            else begin
                minute = minute + 1;
                second = 7'd0;
            end
        end
        else begin
            minute = minute;
            second = second + 1;
        end
    end
    else begin
        minute = minute;
        second = second;
    end
```

```
else begin
    if (enable == 1) begin
        if (second == 7'd0) begin
            if (minute == 7'd0) begin
                minute = 7'd0;
                second = 7'd0;
            end
            else begin
                minute = minute - 1;
                second = 7'd59;
            end
        end
        else begin
            minute = minute;
            second = second - 1;
        end
    end
    else begin
        minute = minute;
        second = second;
    end
end
```

3.學習到的東西與困難

這次 lab 是第一次使用 7 segment 顯示器，了解到原來是利用快速地循環來呈現每一位的數字，而非一次顯示四個數字，非常有意思。我也慢慢的學會把 combinational 和 sequential logic 分成不同的 always block 來寫，一方面程式更有結構化，另一方面每個 always block 變得更乾淨易懂，對於多參數的程式很有幫助，debug 也比較方便。另外，因為這次多了很多不同頻率的 clk，會因為使用錯誤的 clk 或是沒使用相對應的 clk 來執行，而導致顯示的錯誤，希望下次的 lab 能更加注意來避免這種小錯誤重複的發生。

4.想對老師和助教說的話

希望期中上機不會太難:D