LAB08

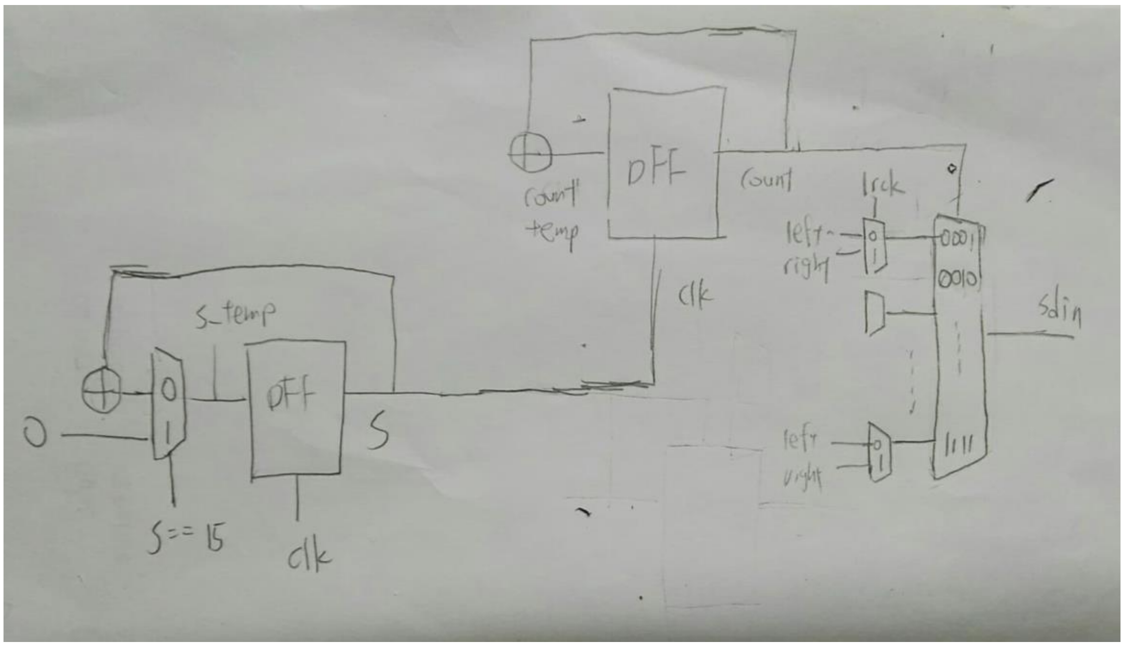
第一題

DESIGN SPECIFICATION

(1) INPUT / OUTPUT

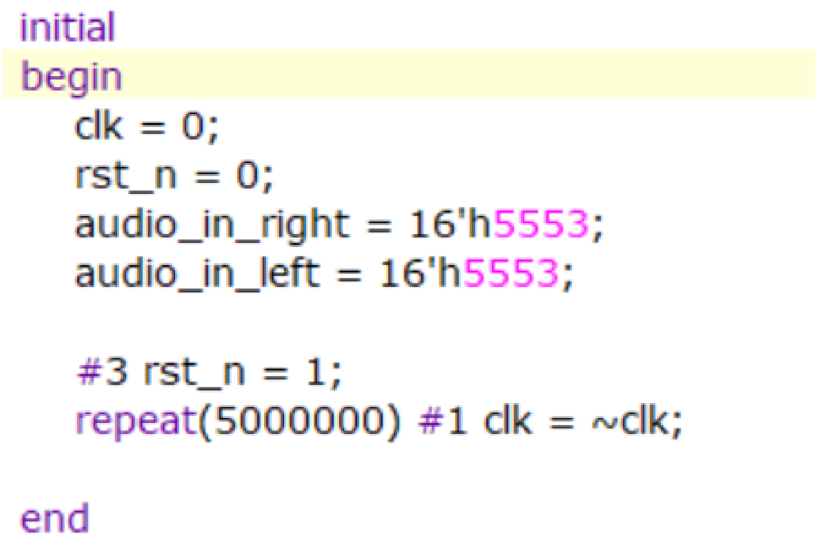
```
module speaker(clk, rst_n, audio_mclk, audio_lrck, audio_sck,
audio_sdin, audio_in_left, audio_in_right);
    input clk;
    input rst_n;
    input [15:0]audio_in_left;
    input [15:0]audio_in_right;
    output reg audio_mclk;
    output reg audio_lrck;
    output reg audio_sck;
    output reg audio_sdin;
```

(2) BLOCK DIAGRAM



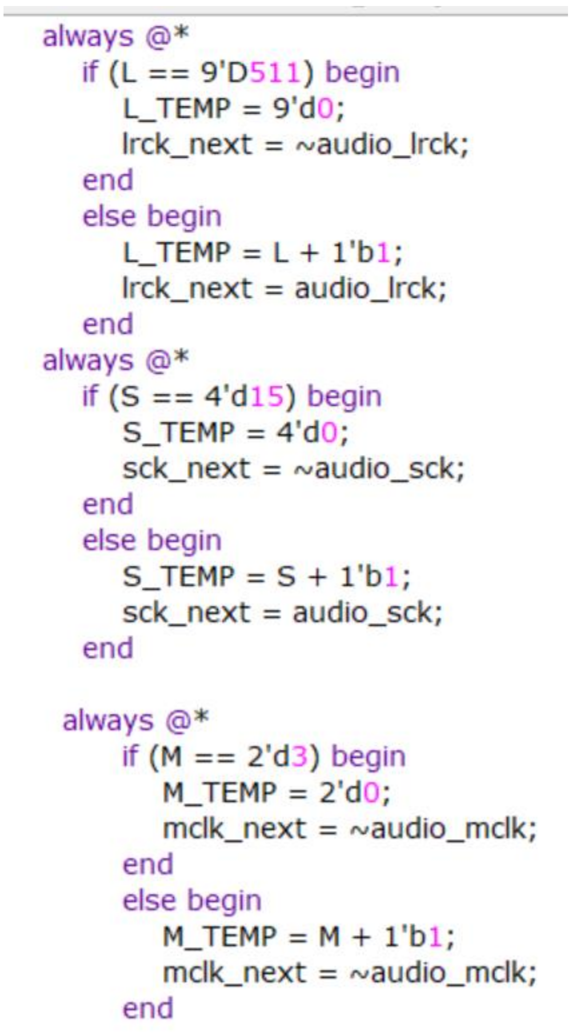
DESIGN IMPLEMENTATION

(1) TEST BENCH

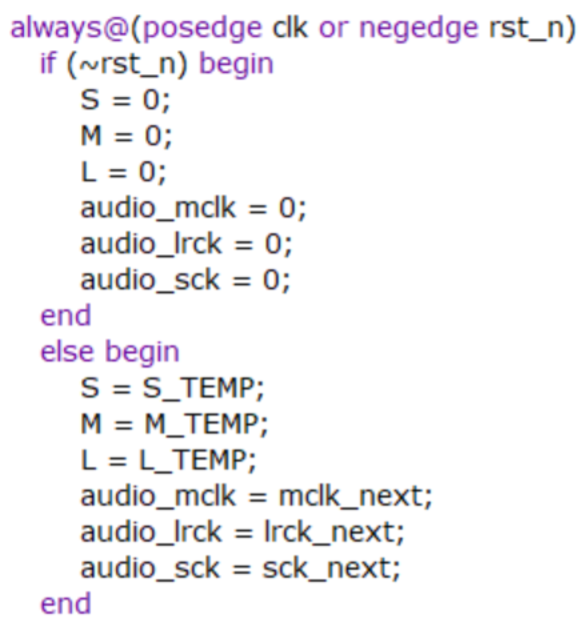


```
initial
begin
   clk = 0;
   rst_n = 0;
   audio_in_right = 16'h5553;
   audio_in_left = 16'h5553;

   #3 rst_n = 1;
   repeat(5000000) #1 clk = ~clk;

end
```

(2) VERILOG CODE

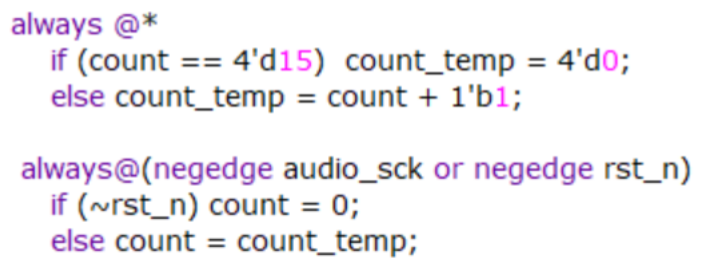


```
always @*
   if (L == 9'D511) begin
      L_TEMP = 9'd0;
      lrck_next = ~audio_lrck;
   end
   else begin
      L_TEMP = L + 1'b1;
      lrck_next = audio_lrck;
   end
always @*
   if (S == 4'd15) begin
      S_TEMP = 4'd0;
      sck_next = ~audio_sck;
   end
   else begin
      S_TEMP = S + 1'b1;
      sck_next = audio_sck;
   end

 always @*
    if (M == 2'd3) begin
       M_TEMP = 2'd0;
       mclk_next = ~audio_mclk;
    end
    else begin
       M_TEMP = M + 1'b1;
       mclk_next = ~audio_mclk;
    end
```

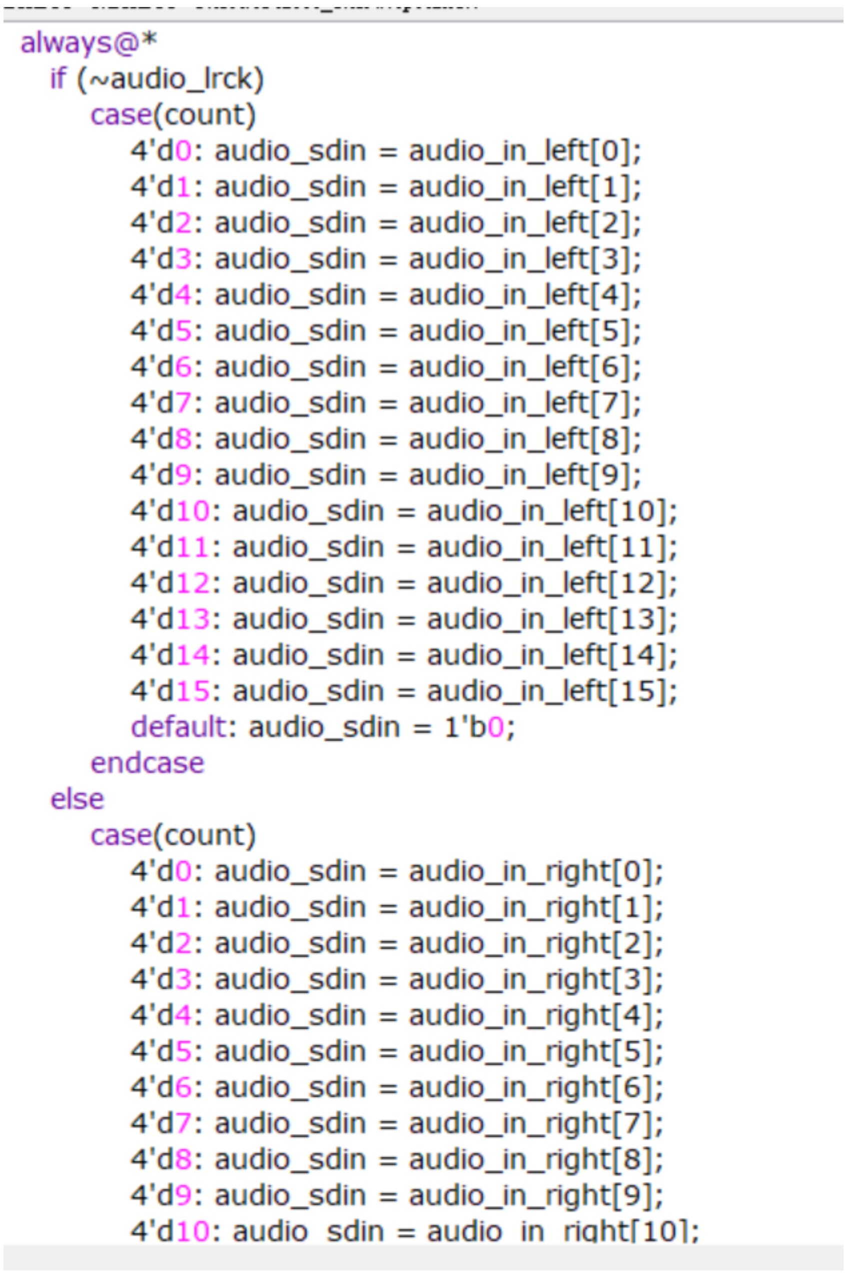


```
always@(posedge clk or negedge rst_n)
  if (~rst_n) begin
     S = 0;
     M = 0;
     L = 0;
     audio_mclk = 0;
     audio_lrck = 0;
     audio_sck = 0;
  end
  else begin
     S = S_TEMP;
     M = M_TEMP;
     L = L_TEMP;
     audio_mclk = mclk_next;
     audio_lrck = lrck_next;
     audio_sck = sck_next;
  end
```

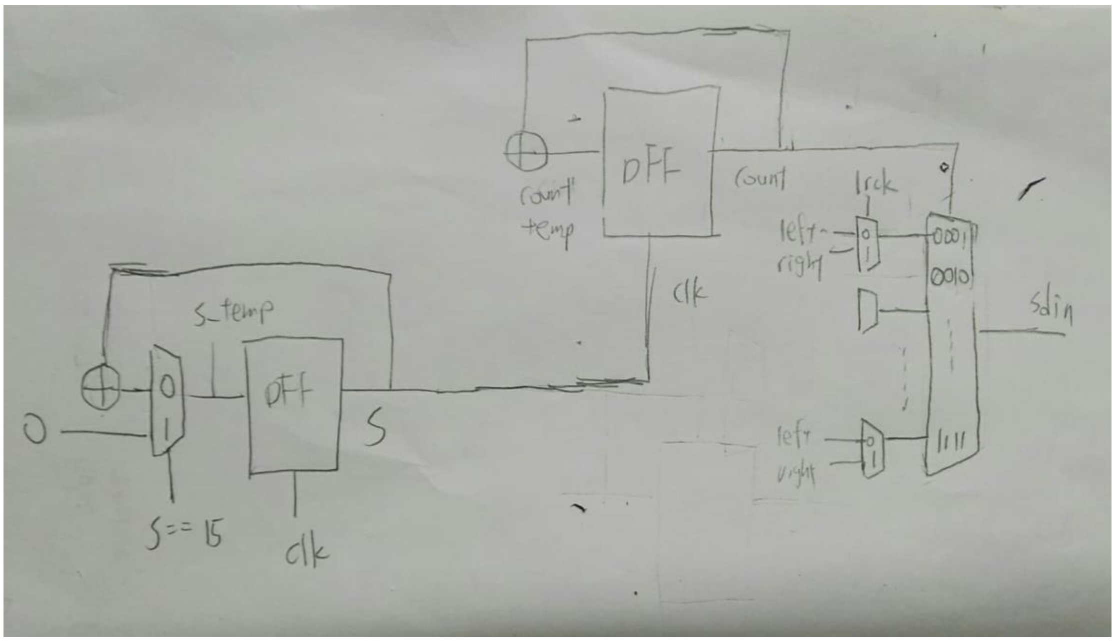
這裡只是普通的 frequency divider，分別產生 mclk, lrck, sck



```
always @*
   if (count == 4'd15)  count_temp = 4'd0;
   else count_temp = count + 1'b1;

 always@(negedge audio_sck or negedge rst_n)
   if (~rst_n) count = 0;
   else count = count_temp;
```



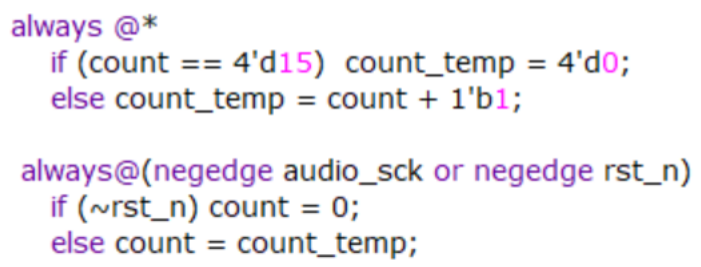
```
always@*
   if (~audio_lrck)
      case(count)
         4'd0: audio_sdin = audio_in_left[0];
         4'd1: audio_sdin = audio_in_left[1];
         4'd2: audio_sdin = audio_in_left[2];
         4'd3: audio_sdin = audio_in_left[3];
         4'd4: audio_sdin = audio_in_left[4];
         4'd5: audio_sdin = audio_in_left[5];
         4'd6: audio_sdin = audio_in_left[6];
         4'd7: audio_sdin = audio_in_left[7];
         4'd8: audio_sdin = audio_in_left[8];
         4'd9: audio_sdin = audio_in_left[9];
         4'd10: audio_sdin = audio_in_left[10];
         4'd11: audio_sdin = audio_in_left[11];
         4'd12: audio_sdin = audio_in_left[12];
         4'd13: audio_sdin = audio_in_left[13];
         4'd14: audio_sdin = audio_in_left[14];
         4'd15: audio_sdin = audio_in_left[15];
         default: audio_sdin = 1'b0;
      endcase
   else
      case(count)
         4'd0: audio_sdin = audio_in_right[0];
         4'd1: audio_sdin = audio_in_right[1];
         4'd2: audio_sdin = audio_in_right[2];
         4'd3: audio_sdin = audio_in_right[3];
         4'd4: audio_sdin = audio_in_right[4];
         4'd5: audio_sdin = audio_in_right[5];
         4'd6: audio_sdin = audio_in_right[6];
         4'd7: audio_sdin = audio_in_right[7];
         4'd8: audio_sdin = audio_in_right[8];
         4'd9: audio_sdin = audio_in_right[9];
         4'd10: audio_sdin = audio_in_right[10];
```



然後 sck 在當另一個 counter 的 clk，counter 的值再接到 mux，把 left 跟 right 做 serial input 到 sdin，至於要 left 或 right 就用 lrck 去選。

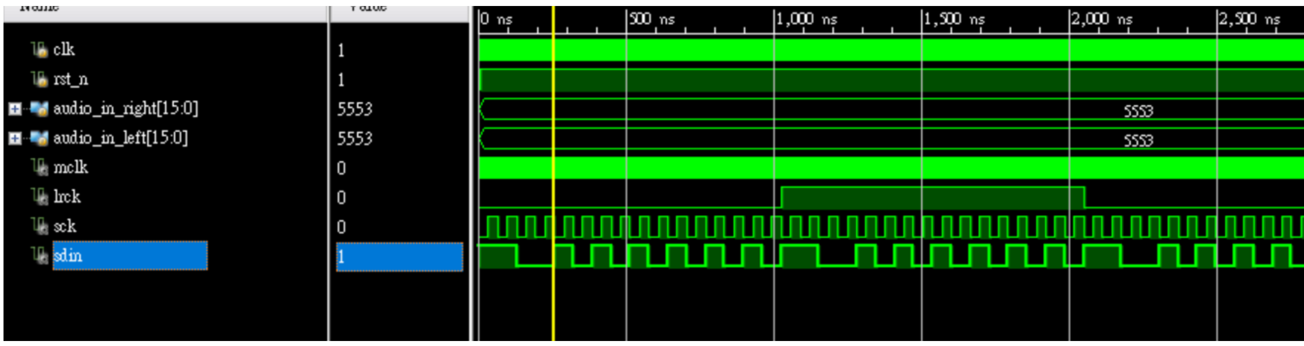
DISSCUSSION

這題有碰到一個問題，我原本只有寫 speaker 跟 tsetbench，但這樣是錯的，於是我只好再打一個 top module 去接 speaker，testbench 再接 top，我猜會出問題的原因是我把 output 拿去當 clk



```
always @*
   if (count == 4'd15)  count_temp = 4'd0;
   else count_temp = count + 1'b1;

 always@(negedge audio_sck or negedge rst_n)
   if (~rst_n) count = 0;
   else count = count_temp;
```

執行結果如下



我 left, right 的值都是設 0101010101010011，可以看到在 sck negedge 時值就輸近來，sck 的值就是以 1100101010101010 的週期在跑。

CONCLUSION

　這個 LAB 不難，但要先理解，上課其實沒有聽很懂，所以花比較多時間在了解 SPEAKER 的功能何要我們做的事，理解了就很快了。

第二題

PS: 只討論與之前不同的

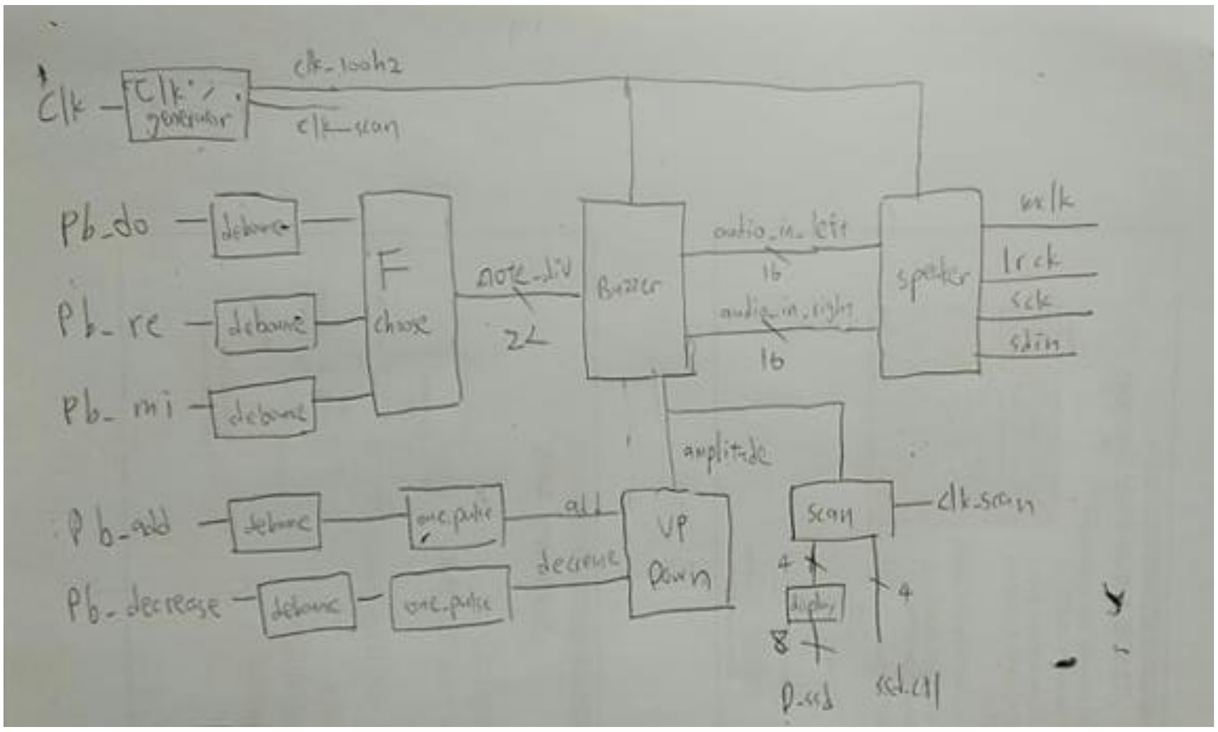
DESIGN SPECIFICATION

(1) INPUT / OUTPUT

```
module top(clk, rst_n, pb_do, pb_re, pb_mi, pb_add, pb_decrease,
mclk, lrck, sck, sdin, ssd_ctl, D_ssd);
    input clk;
    input rst_n;
    input pb_do;
    input pb_re;
```

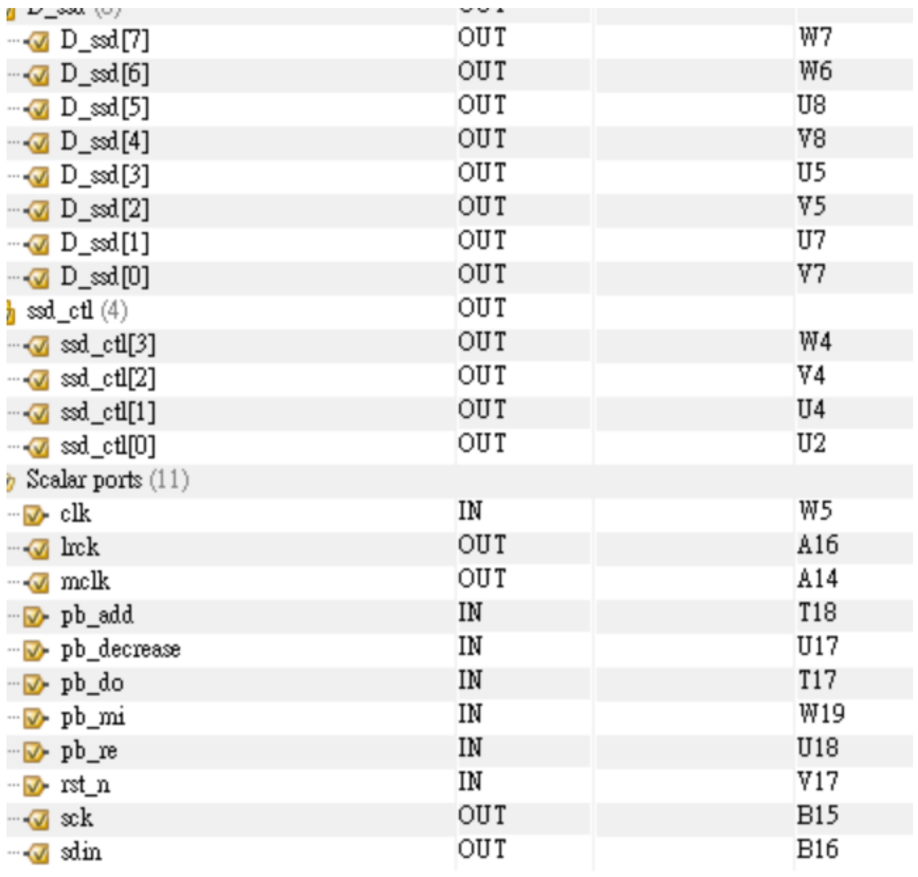
```
        input pb_mi;
        input pb_add;
        input pb_decrease;
        output mclk;
        output lrck;
        output sck;
        output sdin;
        output [3:0]ssd_ctl;
        output [7:0]D_ssd;

module F_choose(DO, RE, MI, DIV);
        input DO;
        input RE;
        input MI;
        output reg [21:0]DIV;

module UP_DOWN(clk, rst_n, add, decrease, q);
        input clk;
        input rst_n;
        input add;
        input decrease;
        output reg [3:0]q;
        reg [3:0]q_temp;

module buzzer(clk, rst_n, note_div, audio_left, audio_right,
amplitude);
        input clk;
        input rst_n;
        input [21:0]note_div;
        input [3:0]amplitude;
        output reg [15:0]audio_left;
        output reg [15:0]audio_right;

module speaker(clk, rst_n, audio_mclk, audio_lrck, audio_sck,
audio_sdin, audio_in_left, audio_in_right);
        input clk;
        input rst_n;
        input [15:0]audio_in_left;
        input [15:0]audio_in_right;
        output reg audio_mclk;
        output reg audio_lrck;
        output reg audio_sck;
        output reg audio_sdin;
```

(2) BLOCK DIAGRAM



DESIGN IMPLEMENTATION

(1) I / O PINS

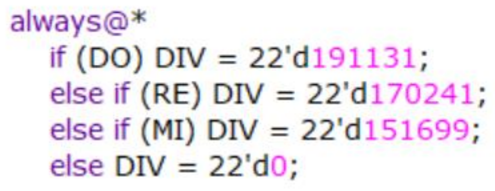


D_ssd[7]	OUT		W7
D_ssd[6]	OUT		W6
D_ssd[5]	OUT		U8
D_ssd[4]	OUT		V8
D_ssd[3]	OUT		U5
D_ssd[2]	OUT		V5
D_ssd[1]	OUT		U7
D_ssd[0]	OUT		V7
ssd_ctl (4)	OUT		
ssd_ctl[3]	OUT		W4
ssd_ctl[2]	OUT		V4
ssd_ctl[1]	OUT		U4
ssd_ctl[0]	OUT		U2
Scalar ports (11)			
clk	IN		W5
lrck	OUT		A16
mclk	OUT		A14
pb_add	IN		T18
pb_decrease	IN		U17
pb_do	IN		T17
pb_mi	IN		W19
pb_re	IN		U18
rst_n	IN		V17
sck	OUT		B15
sdin	OUT		B16

(2) VERILOG CODE

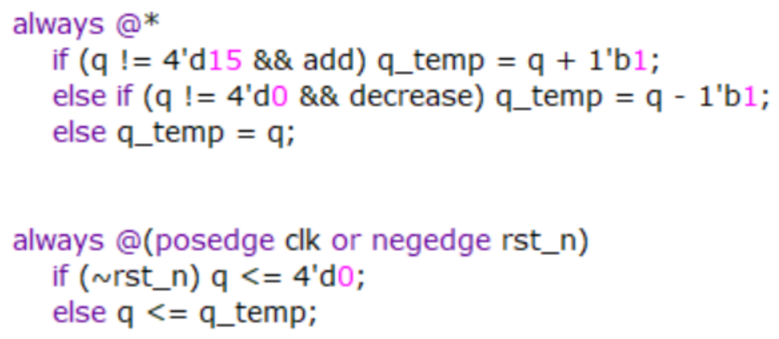
Module F_choose

用來選擇週期，收到 do、re、me debounce 出來的訊號就傳給 buzzer 特定的週期



```
always@*
   if (DO) DIV = 22'd191131;
   else if (RE) DIV = 22'd170241;
   else if (MI) DIV = 22'd151699;
   else DIV = 22'd0;
```

Module UP_DOWN



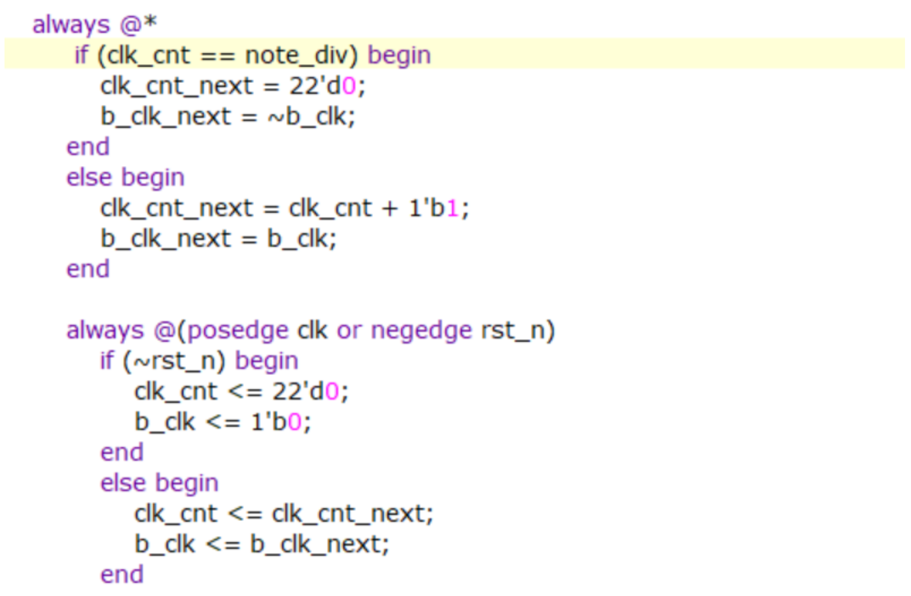
```
always @*
   if (q != 4'd15 && add) q_temp = q + 1'b1;
   else if (q != 4'd0 && decrease) q_temp = q - 1'b1;
   else q_temp = q;


always @(posedge clk or negedge rst_n)
   if (~rst_n) q <= 4'd0;
   else q <= q_temp;
```

如果收到加的訊號就加一，檢的就減一，再把 OUTPUT 傳給

buzzer 跟 scan

Module buzzer

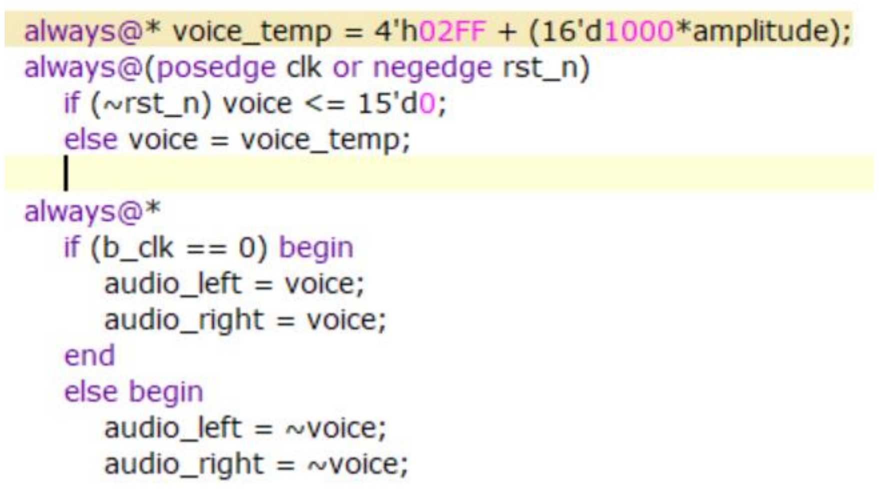
note_div 就是半周期，數到他就歸 0，用以控制音頻



```
always @*
    if (clk_cnt == note_div) begin
        clk_cnt_next = 22'd0;
        b_clk_next = ~b_clk;
    end
    else begin
        clk_cnt_next = clk_cnt + 1'b1;
        b_clk_next = b_clk;
    end

    always @(posedge clk or negedge rst_n)
        if (~rst_n) begin
            clk_cnt <= 22'd0;
            b_clk <= 1'b0;
        end
        else begin
            clk_cnt <= clk_cnt_next;
            b_clk <= b_clk_next;
        end
```

然後用一個 voice 去存音量，然後再輸進 audio left right 裡

Amplitude 是手動控制的音量



```
always@* voice_temp = 4'h02FF + (16'd1000*amplitude);
always@(posedge clk or negedge rst_n)
   if (~rst_n) voice <= 15'd0;
   else voice = voice_temp;
   |
always@*
   if (b_clk == 0) begin
      audio_left = voice;
      audio_right = voice;
   end
   else begin
      audio_left = ~voice;
      audio_right = ~voice;
```

Module speaker (同第一題)

DISSCUSSION

這題有一個基本的 BUG 我 DE 了很久，就是我按加或減他一直不理我，害我一直去檢查 DEBOUNCE 跟 ONE_PULSE 的問題，結果是 UP_DOWN 那裏我打成 if (rst_n) 難怪一直沒反應。

執行結果就是按下 do、re、mi，就產生各自的聲音，按 add 音量就變大，按 decrease，音量會變小，但我聲音變化不明顯，我有事著把初始的音量調小，但這樣聲音不明顯，會變怪怪的。

CONCLUSION

這題其實比以前的 lab 簡單，只是我一開始沒搞清楚 buzzer 是拿來幹嘛的，因為 speaker 就產生出 3 種頻率，害我搞不清楚音頻到底是由哪裡控制。