Lab 8

學號：108071003 姓名：李彥璋

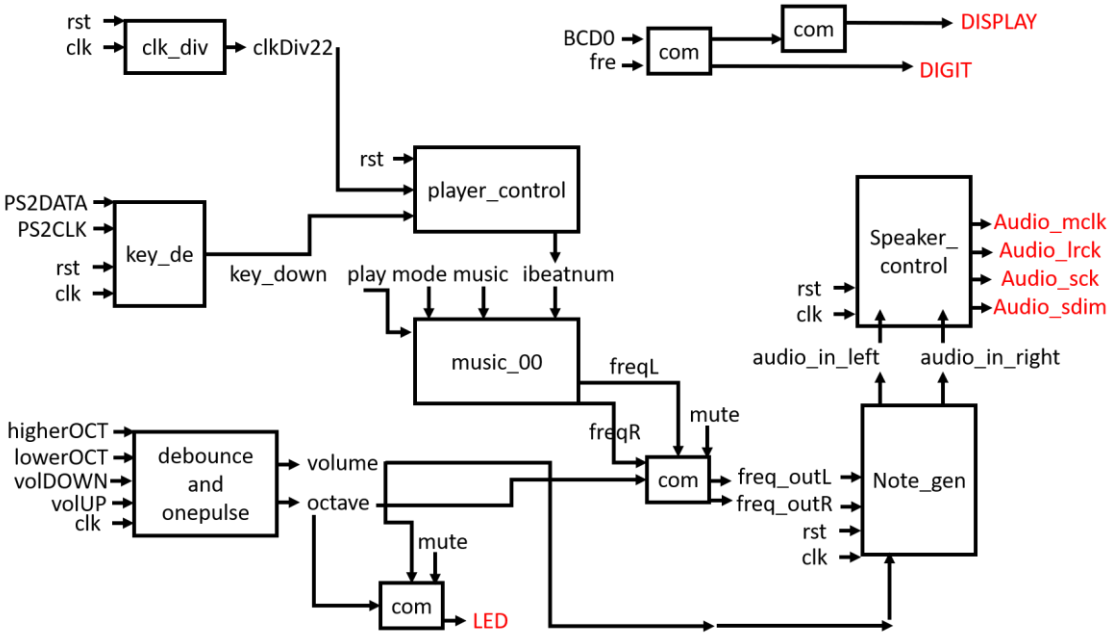
1. 實作過程

- Lab8 要實作一個電子鋼琴，有 play 和 demonstrate 兩種模式—
  Play mode：使用者能透過 keyboard 彈鋼琴；
  Demonstrate mode：鋼琴會示範已經寫好的歌曲。

- 共同功能—
  音量調大/小(5 level)、頻率調高/低(3 level)、靜音、7-segment 顯示音符。

- Demonstrate mode 專屬功能—
  暫停、慢速、切換音樂(bonus)。

- Block diagram—



- Code explaination—

音符

```
1  `define silence   32'd50000000
2  `define sil   32'd50000000
3  `define do   32'd262
4  `define re   32'd294
5  `define mi   32'd330
6  `define fa   32'd349
7  `define so   32'd392
8  `define la   32'd440
9  `define si   32'd494
10
11 `define hdo  32'd524
12 `define hre  32'd588
13 `define hmi  32'd660
14 `define hfa  32'd698
15 `define hso  32'd784
```

[1-15] 以頻率定義會用到的音符(前面加 h 的是高八度)、休止音。

LED

Led15-13 用以顯示當前頻率；Led4-0 用以顯示當前音量。

LED 0~4 indicates the current volume level:

Volume Level	LED 4	LED 3	LED 2	LED 1	LED 0
Muted	○	○	○	○	○
Level 1	○	○	○	○	●
Level 2	○	○	○	●	●
Level 3	○	○	●	●	●
Level 4	○	●	●	●	●
Level 5 (the loudest)	●	●	●	●	●

LED 13~15 indicates the current octave level:

Octave Level		LED 15	LED 14	LED 13
Level 1	Lower	●	○	○
Level 2	Normal	○	●	○
Level 3	Higher	○	○	●

```
66     assign led4 = (volume >= 3'd5) ? 1'b1 : 1'b0;
67     assign led3 = (volume >= 3'd4) ? 1'b1 : 1'b0;
68     assign led2 = (volume >= 3'd3) ? 1'b1 : 1'b0;
69     assign led1 = (volume >= 3'd2) ? 1'b1 : 1'b0;
70     assign led0 = (volume >= 3'd1) ? 1'b1 : 1'b0;
71     assign led15_13 = (octave == 3'd3) ? (3'b100) : ((octave == 3'd2) ? (3'b010):(3'b001));
72     assign led4_0 = (_mute == 1'b1) ? (5'b00000) : ({led4,led3,led2,led1,led0});
73     assign _led = {led15_13 , 8'b00000000 , led4_0};
```

[66-70] volume 變數儲存 1~5 的數值，表音量。以 >= 的方式賦值 Led4-0。

[71] octave 變數儲存 1~3 的數值，表頻率。

[72] 串接 Led4-0。若 mute==1，Led 全暗。

[73] 串接 Led15-0。

Debounce、Onepulse

```
100     debounce d1( _volUP_de , _volUP , clk);
101     onepulse o1( _volUP_de , clk , _volUP_one);
102
103     debounce d2( _volDOWN_de , _volDOWN , clk);
104     onepulse o2( _volDOWN_de , clk , _volDOWN_one);
105
106     debounce d3( _higherOCT_de , _higherOCT , clk);
107     onepulse o3( _higherOCT_de , clk , _higherOCT_one);
108
109     debounce d4( _lowerOCT_de , _lowerOCT , clk);
110     onepulse o4( _lowerOCT_de , clk , _lowerOCT_one);
```

[100-110] 按鈕(控制音量、頻率 level)需要 debounce、onepulse。

控制音量、頻率 level

```
112     always @(posedge clk or posedge rst) begin
113         if(rst == 1'b1) begin
114             volume <= 3'd3;
115             octave <= 3'd2;
116         end else begin
117             if(_volUP_one == 1'b1) begin
118                 if(volume < 3'd5) volume <= volume + 3'd1;
119                 else volume <= volume;
120             end else if(_volDOWN_one == 1'b1) begin
121                 if(volume > 3'd1) volume <= volume - 3'd1;
122                 else volume <= volume;
123             end
124
125             if(_higherOCT_one == 1'b1) begin
126                 if(octave < 3'd3) octave <= octave + 3'd1;
127                 else octave <= octave;
128             end else if(_lowerOCT_one == 1'b1) begin
129                 if(octave > 3'd1) octave <= octave - 3'd1;
130                 else octave <= octave;
131             end
132         end
133     end
```

[113-115] rst 後，volume=3、octave=2。

[117-119] volume up(最高 5)。

[120-123] volume down(最低 1)。

[125-127] octave up(最高 3)。

[128-131] octave down(最低 1)。

7-segment

```
140     always@(posedge clk) begin
141         frequency_cnt <= frequency_cnt + 20'd1;
142     end
143     assign frequency = frequency_cnt[19:18];
```

[143] frequency 是 2 位數(00->01->10->11)，讓 7-segment 視覺暫留的工具。

```
146     always @(*) begin
147         case (frequency)
148             2'b00: begin
149                 value <= BCD0;
150                 DIGIT <= 4'b1110;
151             end
152             2'b01: begin
153                 value <= 4'd11;
154                 DIGIT <= 4'b1101;
155             end
156             2'b10: begin
157                 value <= 4'd11;
158                 DIGIT <= 4'b1011;
159             end
160             2'b11: begin
161                 value <= 4'd11;
162                 DIGIT <= 4'b0111;
163             end
164             default: begin
165                 value <= BCD0;
166                 DIGIT <= 4'b1110;
167             end
168         endcase
169     end
```

[150、154、158、162、166] 輸出 DIGIT，0 的那位是 7-segment 亮的地方。

[149] BCD0 會根據當前音符設定。

[153、157、161] 因為我選的曲子沒有升降記號，所以這 3 位都是 dash(4'd11)。

```
170     always @* begin
171         case (value)
172             4'd0: DISPLAY = 7'b1000110; // C
173             4'd1: DISPLAY = 7'b0100001; // D
174             4'd2: DISPLAY = 7'b0000110; // E
175             4'd3: DISPLAY = 7'b0001110; // F
176             4'd4: DISPLAY = 7'b0010000; // G
177             4'd5: DISPLAY = 7'b0001000; // A
178             4'd6: DISPLAY = 7'b0000011; // B
179             4'd11: DISPLAY = 7'b0111111; // dash
180             default: DISPLAY = 7'b1111111;
181         endcase
182     end
```

[172-178] CDEFGAB 分別代表 DO RE MI FA SO LA SI。

```
184     always @* begin
185         if(freqR == `hdo || freqR == `do) BCD0 = 4'd0;
186         else if(freqR == `hre || freqR == `re) BCD0 = 4'd1;
187         else if(freqR == `hmi || freqR == `mi) BCD0 = 4'd2;
188         else if(freqR == `hfa || freqR == `fa) BCD0 = 4'd3;
189         else if(freqR == `hso || freqR == `so) BCD0 = 4'd4;
190         else if(freqR == `la) BCD0 = 4'd5;
191         else if(freqR == `si) BCD0 = 4'd6;
192         else BCD0 = 4'd11;
193     end
```

[185-191] 根據主旋律(freqR)當前的音符，設置 BCD0，方可顯示於 7-segment。

鍵盤

```
201     parameter [8:0] KEY_CODES [0:6] = {
202         9'b0_0001_1100, // A=> 1C
203         9'b0_0001_1011, // S=> 1B
204         9'b0_0010_0011, // D => 23
205         9'b0_0010_1011, // F => 2B
206         9'b0_0011_0100, // G => 34
207         9'b0_0011_0011, // H => 33
208         9'b0_0011_1011  // J => 3B
209     };
```

[202-208] 在 Play mode，鍵盤按鍵 ASDFGHJ 分別代表 DO RE MI FA SO LA SI。

```
249     assign aa = ((key_down[KEY_CODES[00]] == 1'b1) ? 1'b1 : 1'b0);
250     assign ss = ((key_down[KEY_CODES[01]] == 1'b1) ? 1'b1 : 1'b0);
251     assign dd = ((key_down[KEY_CODES[02]] == 1'b1) ? 1'b1 : 1'b0);
252     assign ff = ((key_down[KEY_CODES[03]] == 1'b1) ? 1'b1 : 1'b0);
253     assign gg = ((key_down[KEY_CODES[04]] == 1'b1) ? 1'b1 : 1'b0);
254     assign hh = ((key_down[KEY_CODES[05]] == 1'b1) ? 1'b1 : 1'b0);
255     assign jj = ((key_down[KEY_CODES[06]] == 1'b1) ? 1'b1 : 1'b0);
```

[249-255] 判斷鍵盤按鍵 ASDFGHJ 是否被使用者長按著。

IbeatNum 的計算，也就是原本的 player_control()

```
259     always @(posedge clkDiv22 or posedge rst) begin
260         if(rst) begin
261             ibeatNum <= 0;
262             counter <= 0;
263         end else begin
264             counter <= counter + 1'b1;
265 
266             if(_mode == 1'b1) begin // 放音樂
267                 if(flag==1) begin
268                     ibeatNum = record;
269                     flag <= 0;
270                 end
271 
272                 if(music_1or2 != _music) begin
273                     music_1or2 <= _music;
274                     ibeatNum = 0;
275                     counter <= 2'd0;
276                 end else begin
277                     if(_play == 1'b1) begin
278                         if(_slow == 1'b1) begin
279                             if(counter == 2'd2) begin
280                                 counter <= 2'd0;
281                                 if(ibeatNum + 1 < LEN) ibeatNum = ibeatNum + 1;
282                                 else ibeatNum = 0;
283                             end else begin
284                                 ibeatNum = ibeatNum;
285                             end
286                         end else begin
287                             counter <= 2'd0;
288                             if(ibeatNum + 1 < LEN) ibeatNum = ibeatNum + 1;
289                             else ibeatNum = 0;
290                         end
291                     end else begin
292                         ibeatNum = ibeatNum;
293                     end
294                 end
295             end else begin // 鍵盤
296                 if(flag==0) begin
297                     record = ibeatNum;
298                     flag = 1;
299                 end
300 
301                 if(aa == 1) ibeatNum = 12'd0;
302                 else if(ss == 1) ibeatNum = 12'd1;
303                 else if(dd == 1) ibeatNum = 12'd2;
304                 else if(ff == 1) ibeatNum = 12'd3;
305                 else if(gg == 1) ibeatNum = 12'd4;
306                 else if(hh == 1) ibeatNum = 12'd5;
307                 else if(jj == 1) ibeatNum = 12'd6;
308                 else ibeatNum = 12'd7;
```

[260-262] rst 後，ibeatnum 和 conter 重置(counter 是當 slow==1 時，用來幫助減速的變數)。

[264] 每次 counter 都+1。

[266] mode==1，表當前是 demonstrate mode。

[267-270] flag==1，代表剛從 play mode 切回到 demonstrate mode，曲子要從剛剛切到 play mode 的斷點繼續播放(record 變數會儲存那個斷點)，flag 歸零。

[272-275] music_1or2 變數是 bonus part 用來判斷是否切換曲子 1 和曲子 2。若 music_1or2 != music，代表切換至另一首曲子，設置 music_1or2、ibeatnum 歸零從頭開始播放曲子、counter 歸零。

[277] 當前為 play 狀態，曲子逕行中。

[278-285] slow==1，代表要減速 2 倍(想成 counter 加兩次 ibeatnum 才往前+1)，這裡的 LEN 是 512，因為一小節 4 個 4 分音符，一個 4 分音符 16 個 ibeatnum，共 8 小節，8*4*16 = 512。

[286-290] normal speed，counter 一直都是歸零，ibeatnum 正常加。

[291-293] 當前為 pause 狀態，曲子暫停。

[295] 當前是 play mode。

[296-299] 從 demonstrate mode 切來 play mode，record 要記錄 demonstrate mode 的曲子播到哪裡了，並將 flag 設為 1 表示有來過 play mode。

[301-308] 根據使用者長按的按鍵，決定 ibeatnum 要播的音符，沒按按鍵就 silence。

決定 frequency_out

```
348     assign freq_outL = (octave == 3) ? (50000000 / (_mute ? `silence : freqL*2)) : ((octave == 1) ?
349     assign freq_outR = (octave == 3) ? (50000000 / (_mute ? `silence : freqR*2)) : ((octave == 1) ?
(50000000 / (_mute ? `silence : freqL/2)) : (50000000 / (_mute ? `silence : freqL)));
(50000000 / (_mute ? `silence : freqR/2)) : (50000000 / (_mute ? `silence : freqR)));
```

[348-349]決定輸出的頻率，如果 mute==1 就是 50000000/50000000；如果 mute==0 就是根據 octave 決定要*2(高 8 度)、不變、/2(低 8 度)。

決定音量，寫在 note_gen()裡面

```
446     always @* begin
447         if(note_div_left == 22'd1) begin
448             audio_left = 16'h0000;
449         end else begin
450             if(volume == 1) audio_left = (b_clk == 1'b0) ? 16'hFF38 : 16'hC8; //200
451             else if(volume == 2) audio_left = (b_clk == 1'b0) ? 16'hF380 : 16'hC80; //3200
452             else if(volume == 3) audio_left = (b_clk == 1'b0) ? 16'hE7C8 : 16'h1838; //6200
453             else if(volume == 4) audio_left = (b_clk == 1'b0) ? 16'hDC10 : 16'h23F0; //9200
454             else if(volume == 5) audio_left = (b_clk == 1'b0) ? 16'hD058 : 16'h2FA8; //12200
455             else audio_left = 16'h0000;
456         end
457     end
458
459     always @* begin
460         if(note_div_right == 22'd1) begin
461             audio_right = 16'h0000;
462         end else begin
463             if(volume == 1) audio_right = (c_clk == 1'b0) ? 16'hFF38 : 16'hC8;
464             else if(volume == 2) audio_right = (c_clk == 1'b0) ? 16'hF380 : 16'hC80;
465             else if(volume == 3) audio_right = (c_clk == 1'b0) ? 16'hE7C8 : 16'h1838;
466             else if(volume == 4)audio_right = (c_clk == 1'b0) ? 16'hDC10 : 16'h23F0;
467             else if(volume == 5) audio_right = (c_clk == 1'b0) ? 16'hD058 : 16'h2FA8;
468             else audio_right = 16'h0000;
469         end
470     end
```

[450-455] 根據 volume 的值，設置 audio。

Spec 裡面的提示有寫，

- Remember that the buzzer/speaker uses 2’s complement numbers for audio signals. When designing the volume level, choose the peak value to be some certain *val* and *-val*.

所以前面那個數是後面那個數的 2’s complement。

Music example

結構如下：

```
If(mode == 1) // 代表 demonstrate mode
    If(music == 0) // 代表曲子 1
        If(play == 1) // 代表曲子逕行中
            Case(ibeatnum)
                12’d0: toneR = `hso
                .
                .
                .
            Endcase
        Else // 代表曲子暫停
            toneR = `sil
    Else // 代表曲子 2
        If(play == 1) // 代表曲子逕行中
            Case(ibeatnum)
                12’d0: toneR = `hso
                .
                .
                .
            Endcase
        Else // 代表曲子暫停
            toneR = `sil
Else // 代表 play mode
    Case(ibeatnum)
        12’d0: toneR = `do
        .
        .
        .
    Endcase
```

...的部分就是曲子的每個音符了，2 分音符 32 個、4 分音符 16 個、8 分音符 32 個，以此類推。如果有連續相同的音，第一個音最後一個改成 silence。

2. 學到的東西與遇到的困難

(1). 鍵盤怪怪的，按下去都沒反應，後來參考 Lab7 shift 按鍵的作法順利解決。

(2). Demonstrate mode 切到 play mode，再切回 demonstrate mode 時，無法從剛剛的斷點繼續播放曲子，已經 record 了但不知為何每次都從頭開始放。後來把 non-blocking 的語法改成 blocking 的語法順利解決(想超久的...)。

3. 想對老師或助教說的話

覺得這個 Lab 是所有 Lab 裡面最有趣的 XD。
謝謝老師&助教。