Lab 8

學號: 109062202 姓名: 陳禹辰

1. 實作過程

這次因為是第一次用 audio peripheral 的東西，所以一開始先花了一點時間搞懂 template 是怎麼操作與進行的，之後就是根據 template 下去改。

我首先把要處理的東西分成幾個部分去做，分別是音調與聲音大小的控制、7-segment 的顯示、頻率的控制(決定要輸出哪個音調的聲音)、鍵盤的控制(在 piano mode 下按下那些鍵會有聲音)。

```
/*volume octave control*/ ···
/*variable*/ ···
/*7-segment*/ ···
/*module*/ ···
/*keyboard*/ ···
/*frequency*/ ···
endmodule
```

鍵盤的部分就是根據按了什麼鍵來決定音頻是哪個，然後 assign 到 k_freqL 和 k_freqR 上。

```
always @(posedge clk or posedge rst) begin
    if(rst)begin
        k_freqR <= 32'd50000000;
    end else begin
        k_freqR <= 32'd50000000;
        if(key_down[last_change]) begin
            case(last_change)
                KEY_CODES[0]: k_freqR <= 32'd262;
                KEY_CODES[1]: k_freqR <= 32'd277;
                KEY_CODES[2]: k_freqR <= 32'd294;
                KEY_CODES[3]: k_freqR <= 32'd311;
                KEY_CODES[4]: k_freqR <= 32'd330;
                KEY_CODES[5]: k_freqR <= 32'd349;
                KEY_CODES[6]: k_freqR <= 32'd369;
                KEY_CODES[7]: k_freqR <= 32'd392;
                KEY_CODES[8]: k_freqR <= 32'd415;
                KEY_CODES[9]: k_freqR <= 32'd440;
                KEY_CODES[10]: k_freqR <= 32'd466;
                KEY_CODES[11]: k_freqR <= 32'd494;
                default : k_freqR <= 32'd50000000;
            endcase
        end
    end
end
```

```
always @(posedge clk or posedge rst) begin
    if(rst)begin
        k_freqL <= 32'd50000000;
    end else begin
        k_freqL <= 32'd50000000;
        if(key_down[last_change]) begin
            case(last_change)
                KEY_CODES[0]: k_freqL <= 32'd262;
                KEY_CODES[1]: k_freqL <= 32'd277;
                KEY_CODES[2]: k_freqL <= 32'd294;
                KEY_CODES[3]: k_freqL <= 32'd311;
                KEY_CODES[4]: k_freqL <= 32'd330;
                KEY_CODES[5]: k_freqL <= 32'd349;
                KEY_CODES[6]: k_freqL <= 32'd369;
                KEY_CODES[7]: k_freqL <= 32'd392;
                KEY_CODES[8]: k_freqL <= 32'd415;
                KEY_CODES[9]: k_freqL <= 32'd440;
                KEY_CODES[10]: k_freqL <= 32'd466;
                KEY_CODES[11]: k_freqL <= 32'd494;
                default : k_freqL <= 32'd50000000;
            endcase
        end
    end
end
```

音量大小跟音頻控制，我設了兩個變數分別是 volume 代表音量是 level 幾(1, 2, 3, 4, 5)、octave 代表音高高是 level 幾(1, 2, 3)，以及相對應 led 要顯示什麼。然後如果超過上限就不能再增加，反之低於下限也不能再減少。

```
always @(posedge clk_16 or posedge rst) begin
    if(rst)begin
        volume <= 4'd3;
    end else begin
        volume <= volume;
        if(one_volup) begin
            if(volume < 4'd5) volume <= volume + 4'd1;
        end else if (one_voldown) begin
            if(volume > 4'd1) volume <= volume - 4'd1;
        end
    end
end
always @(*) begin
    case(volume)
        4'd1: _led[4:0] = 5'b00001;
        4'd2: _led[4:0] = 5'b00011;
        4'd3: _led[4:0] = 5'b00111;
        4'd4: _led[4:0] = 5'b01111;
        4'd5: _led[4:0] = 5'b11111;
    endcase
end
```

```
always @(posedge clk_16 or posedge rst) begin
    if(rst) begin
        octave <= 4'd2;
    end else begin
        octave <= octave;
        if(one_hioct) begin
            if(octave < 4'd3) octave <= octave + 4'd1;
        end else if (one_lowoct) begin
            if(octave > 4'd1) octave <= octave - 4'd1;
        end
    end
end
always @(*) begin
    case(octave)
        4'd1: _led[15:13] = 3'b100;
        4'd2: _led[15:13] = 3'b010;
        4'd3: _led[15:13] = 3'b001;
    endcase
end
```

然後再把這兩個變數丟掉需要用到的地方。控制音量大小的是 note gen，然後我就依照給的 template 下去改。根據傳進來的 volume 來判斷輸出的 audio_right audio_left 應該為多少。

```
always @(*) begin
    if(note_div_left == 22'd1) audio_left = 16'h0000;
    else begin
        case(volume)
            4'd1: audio_left = (b_clk == 1'b0) ? 16'h0E00 : 16'h0200;
            4'd2: audio_left = (b_clk == 1'b0) ? 16'h1C00 : 16'h0400;
            4'd3: audio_left = (b_clk == 1'b0) ? 16'h3800 : 16'h0800;
            4'd4: audio_left = (b_clk == 1'b0) ? 16'h7000 : 16'h1000;
            4'd5: audio_left = (b_clk == 1'b0) ? 16'hE000 : 16'h2000;
        endcase
    end
end
always @(*) begin
    if(note_div_right == 22'd1) audio_right = 16'h0000;
    else begin
        case(volume)
            4'd1: audio_right = (c_clk == 1'b0) ? 16'h0E00 : 16'h0200;
            4'd2: audio_right = (c_clk == 1'b0) ? 16'h1C00 : 16'h0400;
            4'd3: audio_right = (c_clk == 1'b0) ? 16'h3800 : 16'h0800;
            4'd4: audio_right = (c_clk == 1'b0) ? 16'h7000 : 16'h1000;
            4'd5: audio_right = (c_clk == 1'b0) ? 16'hE000 : 16'h2000;
        endcase
    end
end
```

然後 octave 則是丟到控制 frequency 的地方。因為會有兩種音樂可以撥放，所以我用了兩種變數來存一個是 freqL1, freqR1 跟 freqL0, freqR0 代表的是直接從 music example 內出來的 frequency，然後根據_music 來決定要用哪個，以及根據_mode 來決定要用從鍵盤來的還是從歌曲來的，然後根據 octave 來決定 frequency 要維持、乘兩倍還是除兩倍。

```
/*frequency*/
    // freq_outL, freq_outR
    assign freqL = (_music == 1) ? freqL1 : freqL0;
    assign freqR = (_music == 1) ? freqR1 : freqR0;

    assign r_freqL = (_mode == 1) ? freqL : k_freqL;
    assign r_freqR = (_mode == 1) ? freqR : k_freqR;

    assign new_freqL = (_mute == 1) ? 32'd50000000 : (octave == 2) ? r_freqL : (octave == 3) ? (r_freqL*2) : (r_freqL/2);
    assign new_freqR = (_mute == 1) ? 32'd50000000 : (octave == 2) ? r_freqR : (octave == 3) ? (r_freqR*2) : (r_freqR/2);
    // Note gen makes no sound, if freq_out = 50000000 / `silence = 1
    assign freq_outL = 50000000 / new_freqL;
    assign freq_outR = 50000000 / new_freqR;
```

7-segment 則是根據 r_freqR 來判斷所要顯示的是哪個音。然後再傳到 seven_segment 內去顯示。

```
always @(*) begin
    case(r_freqR)
        32'd131: note_1 = 4'd0; //oc
        32'd138: note_1 = 4'd0; //#oc2
        32'd147: note_1 = 4'd1; //od
        32'd155: note_1 = 4'd1; //#od2
        32'd165: note_1 = 4'd2; //oe
        32'd174: note_1 = 4'd3; //of
        32'd185: note_1 = 4'd3; //#of2
        32'd196: note_1 = 4'd4; //og
        32'd207: note_1 = 4'd4; //#og2
        32'd220: note_1 = 4'd5; //oa
```

```
always @* begin
    case (display_number)  // cdefgab
        4'd0: display    = 7'b0100_111;     //c
        4'd1: display    = 7'b0100_001;     //d
        4'd2: display    = 7'b0000_110;     //E
        4'd3: display    = 7'b0001_110;     //F
        4'd4: display    = 7'b1000_010;     //G
        4'd5: display    = 7'b0100_000;     //a
        4'd6: display    = 7'b0000_011;     //b
        4'd7: display    = 7'b0111_111;     //-
        4'd8: display    = 7'b0011_100;     //# (shap notation)
        4'd9: display    = 7'b0000_011;     //b (flat notation)
        default: display = 7'b0111_111; // ERROR
    endcase
end
```

最後就是撥放歌曲時的一些控制了。要讓他在_mode == 1 && _play == 1 的時候 ibeat 才會往後數，然後_slow 的時候我是讓他每個兩個 clock 才會數一次，然後 == 0 的時候就是暫停，至於換歌的時候要歸從頭開始撥放歌曲(也就是 ibeat 要歸 0)，我是設一個變數 switch 來判斷是不是剛切換歌曲，如果現在 music 跟前一個 clock 的不同就代表剛切換就讓 switch = 1，然後把 music 改成現在的 music。

```
always @(posedge clk or posedge reset) begin
    if (reset) begin
        ibeat <= 0;
    end else begin
        ibeat <= ibeat;
        if (_mode == 1) begin
            if(switch == 0) begin
                if(_play == 1'b1) begin
                    if(_slow == 1'b1) begin
                        if(slow)begin
                            ibeat <= next_ibeat_d;
                            slow <= ~slow;
                        end else begin
                            ibeat <= ibeat;
                            slow <= ~slow;
                        end
                    end else begin
                        ibeat <= next_ibeat_d;
                    end
                end
            end else ibeat <= 0;
        end
    end
end
```

```
always @(posedge clk) begin
    if(_music)begin
        if(_music == origin)begin
            switch <= 0;
            origin <= _music;
        end else begin
            switch <= 1;
            origin <= _music;
        end
    end else begin
        if(_music == origin)begin
            switch <= 0;
            origin <= _music;
        end else begin
            switch <= 1;
            origin <= _music;
        end
    end
end
```

然後要刻歌的話我是直接寫 python 這樣只要輸入那些音需要幾個，就可以直接跑完了(雖然是 demo 的時候等得太無聊為了 final project 才寫的)。

```python
music = []
while 1:
    description = input().split(' ')
    if description[0] == 'stop': break
    note = description[0]
    num = description[1]
    music.append((note, int(num)))
j = 0
for i in range(len(music)) :
    for k in range(0, music[i][1], 2):
        print(f'12\'d{j}: toneL = `{music[i][0]};    12\'d{j+1}: toneL = `{music[i][0]};')
        j += 2
    print('')
```

```
a 8
c 8
f 16
e 32
stop 0
12'd0: toneL = `a;    12'd1: toneL = `a;
12'd2: toneL = `a;    12'd3: toneL = `a;
12'd4: toneL = `a;    12'd5: toneL = `a;
12'd6: toneL = `a;    12'd7: toneL = `a;

12'd8: toneL = `c;    12'd9: toneL = `c;
12'd10: toneL = `c;    12'd11: toneL = `c;
12'd12: toneL = `c;    12'd13: toneL = `c;
12'd14: toneL = `c;    12'd15: toneL = `c;

12'd16: toneL = `f;    12'd17: toneL = `f;
12'd18: toneL = `f;    12'd19: toneL = `f;
12'd20: toneL = `f;    12'd21: toneL = `f;
12'd22: toneL = `f;    12'd23: toneL = `f;
12'd24: toneL = `f;    12'd25: toneL = `f;
12'd26: toneL = `f;    12'd27: toneL = `f;
12'd28: toneL = `f;    12'd29: toneL = `f;
12'd30: toneL = `f;    12'd31: toneL = `f;

12'd32: toneL = `e;    12'd33: toneL = `e;
12'd34: toneL = `e;    12'd35: toneL = `e;
12'd36: toneL = `e;    12'd37: toneL = `e;
12'd38: toneL = `e;    12'd39: toneL = `e;
```

2. 學到的東西與遇到的困難

這次因為給的 template 蠻多的，所以看懂之後就蠻好操作的，唯一的困難是我一開始把 keyboard 寫在 player_control 內，但是後來發現傳進去的 clock 是除頻過的會太慢所以行不通，剩下的就還好了。

3. 想對老師或助教說的話

聖誕快樂！