Lab 7

學號: 109000205 姓名: 蕭皓隆

1. 實作過程

此次 Lab 有分為兩個模式，分別為 PLAY 跟 DEMOSTRATE，我利用 assign freqL、assign freqR 時判斷_mode 來決定要輸出何種模式的聲音頻率，bonus 的部分則是在 DEMONSTRATE MODE 時判斷_music 來賦予不同的 frequency。

```
wire [31:0] freqL, freqR;
assign freqL = (_mode) ? ((_music) ? freqL_DEMO : freqL_DEMO2) : freqL_play;
assign freqR = (_mode) ? ((_music) ? freqR_DEMO : freqR_DEMO2) : freqR_play;
```

PLAY:

在 PLAY mode 裡面有鍵盤彈奏、調音量、靜音、調音高以及 7-segment 顯示等功能，將針對以上功能做說明：

(1) 鍵盤彈奏：

利用 PLAY_NOTE 來記錄鍵盤按下的按鍵為何(數字為自行定義的各個按鍵)，其中一開始的 if(!mode)就可限制鍵盤彈奏的功能在 PLAY mode 下才能執行。

在接下來的 always block 裡面利用 case 判斷 PLAY_NOTE 為何，並依不同的 case 給予不同的聲音頻率以達到相對應的音。

```
reg [3:0] PLAY_NOTE;
always @(posedge clk or posedge rst) begin
    if(rst) begin
        PLAY_NOTE <= 0;
    end
    else begin
        if(!_mode) begin
            if(key_down[last_change]==1'b1) begin
                if(key_num == 4'b0000) begin
                    PLAY_NOTE <= 1;
                end
                else if(key_num == 4'b0001) begin
                    PLAY_NOTE <= 2;
                end
                else if(key_num == 4'b0010) begin
                    PLAY_NOTE <= 3;
                end
                else if(key_num == 4'b0011) begin
                    PLAY_NOTE <= 4;
                end
                else if(key_num == 4'b0100) begin
                    PLAY_NOTE <= 5;
                end
                else if(key_num == 4'b0101) begin
                    PLAY_NOTE <= 6;
                end
                else if(key_num == 4'b0110) begin
                    PLAY_NOTE <= 7;
                end
                else begin
                    PLAY_NOTE <= 0;
                end
            end
            else begin
                PLAY_NOTE <= 0;
            end
        end
        else begin
            PLAY_NOTE <= 0;
        end
    end
end
```

```
reg [31:0] freqL_play, freqR_play;
always @(posedge clkDiv13 or posedge rst) begin
    if(rst) begin
        freqL_play <= `silence;
        freqR_play <= `silence;
    end
    else begin
        case(PLAY_NOTE)
            0: begin
                freqL_play <= `silence;
                freqR_play <= `silence;
            end
            1: begin
                freqL_play <= `A;
                freqR_play <= `A;
            end
            2: begin
                freqL_play <= `S;
                freqR_play <= `S;
            end
            3: begin
                freqL_play <= `D;
                freqR_play <= `D;
            end
            4: begin
                freqL_play <= `F;
                freqR_play <= `F;
            end
            5: begin
                freqL_play <= `G;
                freqR_play <= `G;
            end
            6: begin
                freqL_play <= `H;
                freqR_play <= `H;
            end
            7: begin
                freqL_play <= `J;
                freqR_play <= `J;
            end
            default: begin
                freqL_play <= `silence;
                freqR_play <= `silence;
            end
        endcase
    end
end
```

(2) 調音量、靜音：

將_volUP、_volDOWN 訊號做 debounce 和 onepulse 處理，並利用此二按鈕控制變數 vol，再將此變數傳入 module note_gen 裡面用來判斷需要如何調整聲音輸出的振幅。

要達成靜音的效果則是直接宣告 wire VOL 當作傳進去 module note_gen 裡面的 input，並利用 assign VOL 時判斷_mute 是否等於 1，並決定該把 vol 的值給 VOL 或是讓 VOL=0。

```
//volume control
reg [2:0] vol, vol_next;
always @(posedge clkDiv13 or posedge rst) begin
    if(rst) begin
        vol <= 3;
    end
    else begin
        vol <= vol_next;
    end
end
always @* begin
    if(_VOLUP) begin
        vol_next = (vol==5) ? 5 : vol+1;
    end
    else if(_VOLDOWN) begin
        vol_next = (vol==1) ? 1 : vol-1;
    end
    else begin
        vol_next = vol;
    end
end
```

```
wire [2:0] VOL;
assign VOL = _mute ? 0 : vol;
```

```
// Assign the amplitude of the note
// Volume is controlled here
assign audio_left = (note_div_left == 22'd1 || vol==0) ? 16'h0000 :
                    (b_clk == 1'b0) ? (vol==1) ? 16'hE000 :
                                      (vol==2) ? 16'hD000 :
                                      (vol==3) ? 16'hC000 :
                                      (vol==4) ? 16'hB000 : 16'hA000 : (vol==1) ? 16'h2000 :
                                                                       (vol==2) ? 16'h3000 :
                                                                       (vol==3) ? 16'h4000 :
                                                                       (vol==4) ? 16'h5000 : 16'h6000;
assign audio_right = (note_div_right == 22'd1 || vol==0) ? 16'h0000 :
                     (c_clk == 1'b0) ? (vol==1) ? 16'hE000 :
                                       (vol==2) ? 16'hD000 :
                                       (vol==3) ? 16'hC000 :
                                       (vol==4) ? 16'hB000 : 16'hA000 : (vol==1) ? 16'h2000 :
                                                                        (vol==2) ? 16'h3000 :
                                                                        (vol==3) ? 16'h4000 :
                                                                        (vol==4) ? 16'h5000 : 16'h6000;
```

(3) 調音高：

將_higherOCT、_lowerOCT 做 debounce 和 onepulse 處理，並利用此二按鈕控制變數 oct，利用 oct 來記錄目前音高的等級，並在 assign freq_out 時判斷 oct 為何，若 oct==1 則將頻率除以 2，若 oct==2 則不變，若 oct==3 則將頻率乘以 2。

```
// freq_outL, freq_outR
// Note gen makes no sound, if freq_out = 50000000 / `silence = 1
assign freq_outL = (oct==2)? 50000000 / freqL : (oct==1) ? 25000000 / freqL : 100000000 / freqL;
assign freq_outR = (oct==2)? 50000000 / freqR : (oct==1) ? 25000000 / freqR : 100000000 / freqR;
```

```
//octave control
reg [3:0] oct, oct_next;
always @(posedge clkDiv13 or posedge rst) begin
    if(rst) begin
        oct <= 2;
    end
    else begin
        oct <= oct_next;
    end
end
always @* begin
    if(_LOWEROCT) begin
        oct_next = (oct==1) ? 1 : oct-1;
    end
    else if(_HIGHEROCT) begin
        oct_next = (oct==3) ? 3 : oct+1;
    end
    else begin
        oct_next = oct;
    end
end
```

(4) 7-segment：

在 PLAY mode 下判斷為何 PLAY_NOTE 為何，並給予相對應的英文記號。

DEMONSTRATE：

在 DEMONSTRATE mode 裡面有自動演奏樂曲、放慢速度、調音量、靜音、調音高、切換音樂以及 7-segment 顯示等功能，將針對以上功能做說明：

(1) 自動演奏音樂、放慢速度：

依照樂譜延長 module music_example，bonus 曲目也是按照 music_example 去製作成另一首樂曲。要讓樂曲循環播放則是對 ibeat 做操作，若 ibeat + 1 <= LEN，則將 ibeat 變成 0，這樣 ibeat 就會無限循環數下去。放慢速度的部分則是新增一個變數(count)，當 count==2 時才將 ibeat + 1，並將 count 歸零重算，如此一來速度就會變成 0.5 倍。

```
always @(posedge clk, posedge reset) begin
    if (reset) begin
        ibeat <= 0;
        count <= 0;
        music_now <= 0;
    end else begin
        ibeat <= next_ibeat;
        count <= count_next;
        music_now <= _music;
    end
end
always @* begin
    if(_music != music_now) begin
        next_ibeat = 0;
        count_next = 0;
    end
    else if(_play && _mode && ((count==2 && _slow) || !_slow)) begin
        next_ibeat = (ibeat + 1 <= LEN) ? (ibeat+1) : 0;
        count_next = 0;
    end
    else begin
        next_ibeat = ibeat;
        if(_slow && _mode) begin
            count_next = count + 1;
        end
        else begin
            count_next = count;
        end
    end
end
```

(2) 調音量、靜音、調音高：

與 PLAY mode 一致。

(3) 切換音樂：

在_mode==1 時判斷_music 來決定要給哪一首曲子的頻率

```
wire [31:0] freqL, freqR;
assign freqL = (_mode) ? ((_music) ? freqL_DEMO : freqL_DEMO2) : freqL_play;
assign freqR = (_mode) ? ((_music) ? freqR_DEMO : freqR_DEMO2) : freqR_play;
```

```
// Music module
// [in]  beat number and en
// [out] left & right raw frequency
music_example music_00 (
    .ibeatNum(ibeatNum),
    .en(_play),
    .toneL(freqL_DEMO),
    .toneR(freqR_DEMO)
);

wire [31:0] freqL_DEMO2, freqR_DEMO2;
music_2 music_02 (
    .ibeatNum(ibeatNum),
    .en(_play),
    .toneL(freqL_DEMO2),
    .toneR(freqR_DEMO2)
);
```

(4) 7-segment：

判斷目前輸出的 frequency 來決定 7-segment 要顯示何種英文記號。

2. 學到的東西與遇到的困難

學到的東西：聲音訊號的處理，以及想達到各式各樣的聲音效果需要經過怎麼樣的處理才能符合預期，這次 Lab 結合蠻多之前學到的東西，把各式各樣東西的整合起來蠻有成就感。

遇到的困難：要把音符轉成可以演奏的曲子有點累人，雖然可以用程式來產生，但後來發現還是用手打比較好掌握，但缺點是真的很累人...

3. 想對老師或助教說的話