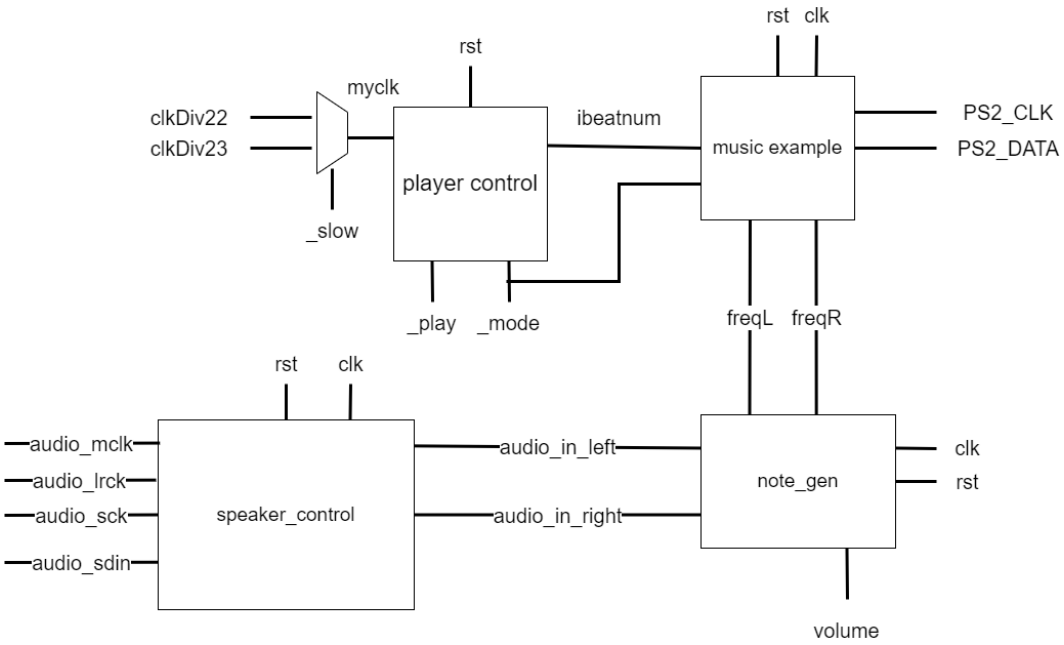
Lab 8

學號: 109062318 姓名: 簡弘哲

1. 實作過程



(我在 music_example 裡處理鍵盤的部分)

這次的 lab 只要能搞懂上課講義裡的東西，修改 template 就快上許多也算容易。

7segment:

```
//7-segment control, freqR = main melody
reg [3:0] num;
always @(*) begin
    if(!_mode || (_mode && _play)) begin   //
        if(freqR == `a || freqR == `ha)
            num = 4'd5;
        else if(freqR == `b || freqR == `hb)
            num = 4'd6;
        else if(freqR == `c || freqR == `hc)
            num = 4'd0;
        else if(freqR == `d || freqR == `hd)
            num = 4'd1;
        else if(freqR == `e || freqR == `he)
            num = 4'd2;
        else if(freqR == `f || freqR == `hf)
            num = 4'd3;
        else if(freqR == `g || freqR == `hg)
            num = 4'd4;
        else if(`sil)
            num = 4'd7;
        else
            num = 4'd7;
    end else begin
        num = 4'd7;
    end
end
```

在 play mode(mode==0) 按下鍵盤(不管其他的 switch 是開或關)，或是在 demo mode 且播放的情況下(mode && play)，都要在 7segment 上顯示英文字母；else 的部分就例如 demo mode 且 pause 的情況。

調整 octave、volume:

```
if(_higherOCT_1p) begin
    if(octave==3) begin
        octave_next=3;
    end else begin
        octave_next=octave+1;
    end
end

if(_lowerOCT_1p) begin
    if(octave==1) begin
        octave_next=1;
    end else begin
        octave_next=octave-1;
    end
end
```

```
//adjust volume,octave
always @(*) begin
    volume_next=volume;
    octave_next=octave;
    if(_volUP_1p) begin
        if(volume==5) begin
            volume_next=5;
        end else begin
            volume_next=volume+1;
        end
    end

    if(_volDOWN_1p) begin
        if(volume==1) begin
            volume_next=1;
        end else begin
            volume_next=volume-1;
        end
    end
```

僅須注意 boundary case(octave:1~3、volume:1~5)就好，其他都是正常加減運算。

Ibeat:

```
always @* begin
    next_ibeat=ibeat;
    if(!_play || !_mode) begin  //pause || user play mode
        next_ibeat = ibeat;
    end else begin
        next_ibeat = (ibeat + 1 < LEN) ? (ibeat + 1) : 0;
    end
end
```

如果從 demo mode 切到 play mode、或是在 demo mode 將 pause 扳起，則 ibeat 得維持原來的值，否則(在 demo mode && play)就讓歌曲重複循環，當 ibeat 跑到最後一個音符時就讓它從 0 開始，就可以達到重複播放的功能。

調整頻率:

```
always @(*) begin
    if(!_mode || (_mode && _play)) begin //user play mode || (demonstrate && play)
        freq_outL = 50000000 / (_mute ? `silence : freqL);
        if(octave==1) begin
            freq_outL = 50000000 / (_mute ? `silence : freqL/2);
        end else if(octave==2) begin
            freq_outL = 50000000 / (_mute ? `silence : freqL);
        end else if(octave==3) begin
            freq_outL = 50000000 / (_mute ? `silence : freqL*2);
        end
    end else begin
        freq_outL = 50000000 / `silence;
    end
end
```

根據 octave 的值去處理從其他 module 得到的 freqL，當升高一個八度，freq 就乘 2。將 50000000/silence 就可以靜音，因為 freq_outL 會接上 note_gen module 的 note_div_left。在 note_gen 裡，當 freq_outL==1 時會將 audio 設為 0，藉此達到靜音的效果。

各按鍵所對應到的音高:

```
always @* begin
    if(en == 1) begin   //demonstrate mode
        case(ibeatNum)…
        endcase
    end else begin  //user play mode
        //toneR = `sil;
        if(key_down[last_change]) begin
            if(key_num!=3'b111) begin
                if(key_num==3'b000) begin
                    toneR = `c;
                end else if(key_num==3'b001) begin
                    toneR = `d;
                end else if(key_num==3'b010) begin
                    toneR = `e;
                end else if(key_num==3'b011) begin
                    toneR = `f;
                end else if(key_num==3'b100) begin
                    toneR = `g;
                end else if(key_num==3'b101) begin
                    toneR = `a;
                end else if(key_num==3'b110) begin
                    toneR = `b;
                end else begin
                    toneR = `sil;
                end
            end
        end else begin
            toneR = `sil;
        end
    end
```

我將_mode 接到 music_example 的 en 上，所以當 en=1 的時候(demo mode)就播放提前寫好的歌，反之(play mode)就要接收鍵盤的訊號，為了要達成 spec 中”按下有反應、放開則無”的條件，我加了一個 else 代表沒有按下按鍵的情況，就讓 tone=sil，這是我在 exam2 中發現的，把 key_valid 的判斷去掉再加上一個 else 處理放開按鍵的情況，就能達到 spec 的要求。

Led:

```
always @(*) begin
    _led_next = _led;
    if(_mute) begin
        _led_next[4:0] = 5'b00000;
    end else begin
        if(volume==1) begin
            _led_next[4:0] = 5'b00001;
        end else if(volume==2) begin
            _led_next[4:0] = 5'b00011;
        end else if(volume==3) begin
            _led_next[4:0] = 5'b00111;
        end else if(volume==4) begin
            _led_next[4:0] = 5'b01111;
        end else if(volume==5) begin
            _led_next[4:0] = 5'b11111;
        end else begin
            _led_next[4:0] = 5'b00000;
        end

        if(octave==1) begin
            _led_next[15:13] = 3'b100;
        end else if(octave==2) begin
            _led_next[15:13] = 3'b010;
        end else if(octave==3) begin
            _led_next[15:13] = 3'b001;
        end else begin
            _led_next[15:13] = 3'b000;
        end
    end
end
```

先看有沒有靜音，如果有就讓 led 全暗，沒有的話就根據 volume、octave 的值讓該亮的 led 都亮起來。

音量大小:

```
always @(*) begin
    if(note_div_left == 22'd1) begin
        audio_left = 16'h0000;
    end else begin
        if(volume==1) begin
            audio_left = (b_clk == 1'b0) ? 16'hF000 : 16'h1000;
        end else if(volume==2) begin
            audio_left = (b_clk == 1'b0) ? 16'hE000 : 16'h2000;
        end else if(volume==3) begin
            audio_left = (b_clk == 1'b0) ? 16'hC000 : 16'h4000;
        end else if(volume==4) begin
            audio_left = (b_clk == 1'b0) ? 16'hB000 : 16'h5000;
        end else if(volume==5) begin
            audio_left = (b_clk == 1'b0) ? 16'hA000 : 16'h6000;
        end else begin
            audio_left = 16'h0000;
        end
    end
end
```

仿造 template 的寫法，設定好振幅的極大與極小值就行了，像是 C000 = -2^14, 4000 = 2^14，兩個極值的絕對值需一樣。

2. 學到的東西與遇到的困難

   在處理 volume、octave 的 led 時，我將它們分開寫在兩個不同的 always block 裡，一個管 led[4:0]、另一個管 led[15:13]，但這樣導致我的 led 完全不會動。一開始我以為是 debounce、onepulse 的問題，但事實上不是。我又去檢查了一下變數 volume、octave 的值，將它們的值顯示在 7segment 上，是正常的。這困擾了我 1,2hr，明明邏輯是對的但 led 就是不會正常運作，直到我眼尖發現同一個變數 led_next 出現在不同 always block 裡(但它們取值範圍不一樣，一個是[4:0]、另一個是[15:13])，我將它們合併以後就正常了。vivado 當時的 warning 裡沒有警告我 multi net driven，直到我自己發現這個問題，但我不確定這算不算一種 multi net driven?

3. 想對老師或助教說的話

   問上題所提及的問題

   reg [15:0] led;

   always @(*) begin
   led_next[4:0] = …
   end
   always @(*) begin
   led_next[15:13] = …
   end

   雖然同一個變數出現在不同的 always block 裡，但它們 assign 的範圍不一樣，那這樣還會造成 multi net driven 的問題嗎?