EECS 2070 02 Digital Design Labs 2019

Lab 8

學號：107062107　姓名：王領崧

1.前言

此次 lab 外接聲音模組，搭配板子上的按鈕、開關、LED 燈與七段顯示器，實作出類似 MP3 的小型撥放器，並要手刻兩小段旋律作為切換的歌曲。

2.實作過程

這次 lab 是以助教給的 Sample Code 下去修改完成的，以下會分成有進行修改的 module 個別條列解釋。

note_gen

note_gen module 主要是調整左右聲道的音量大小，有點類似震幅的概念，給定上下的臨界值，並隨著 clk 去驅動。這邊直接利用 case 判斷式來判斷 input volume，從 0 到 5 給予不同的兩補數大小，音量越小的部分的差距越大，反之則越小，當音量為 0 時，數值直接給 0。

```
always @(*) begin
    case (volume)
        3'd0 : begin
            audio_left = 16'h0000;
            audio_right = 16'h0000;
        end
        3'd1: begin
            audio_left = (b_clk == 1'b0) ? 16'hF000 : 16'h1000;
            audio_right = (c_clk == 1'b0) ? 16'hF000 : 16'h1000;
        end
        3'd2 : begin
            audio_left = (b_clk == 1'b0) ? 16'hE000 : 16'h2000;
            audio_right = (c_clk == 1'b0) ? 16'hE000 : 16'h2000;
        end
        3'd3 : begin
            audio_left = (b_clk == 1'b0) ? 16'hB000 : 16'h4000;
            audio_right = (c_clk == 1'b0) ? 16'hB000 : 16'h4000;
        end
        3'd4 : begin
            audio_left = (b_clk == 1'b0) ? 16'hA000 : 16'h6000;
            audio_right = (c_clk == 1'b0) ? 16'hA000 : 16'h6000;
        end
        3'd5 : begin
            audio_left = (b_clk == 1'b0) ? 16'h9000 : 16'h7000;
            audio_right = (c_clk == 1'b0) ? 16'h9000 : 16'h7000;
        end
        default : begin
            audio_left = 16'h0000;
            audio_right = 16'h0000;
        end
    endcase
end
```

player_control

player_control module 為使用者主要控制的部分，使用了 3 個 always block 來實作切歌、暫停、循環的功能。

第一個 always block 為控制樂譜的進行，像個 counter 一樣，在播放模式下，將 ibeat 不斷累加，如果加到歌曲的長度的話，則判斷是否要循環撥放，如果是的話，就將 ibeat 歸零，反之將 ibeat 維持在最後一個音(silence)，如果是在暫停模式下，則將 ibeat 維持住，固定現在音樂的位置。

第二個 always block 用來當切歌發生時，將 ibeat 從頭開始計數。我利用加入 check 值來判斷切歌的發生，當 music 開關為開時，check 為 1，否則為 0，因此，當 music 與 check 不同步時，就可以判斷出此開關剛剛被切換，因此將 ibeat 歸零，check 設為相同的值，如果 music 與 check 同步時，代表沒有切換歌曲的情況發生，ibeat 等於第一個 always block 所產生的 next_ibeat。

第三個 always block 為給音樂長度值，根據 music 開關的開與關，給予相對應的歌曲長度。

```
always @* begin
    if (_play) begin
        next_ibeat = (ibeat + 1 < LEN) ? (ibeat + 1) : ((_repeat) ? 15'd0 : ibeat);
    end
    else begin
        next_ibeat = ibeat;
    end
end
```

```
always @(*) begin
    if (_music) begin
        LEN = 15'd321;
    end
    else begin
        LEN = 15'd513;
    end
end
```

```
always @(posedge clk, posedge reset) begin
    if (reset) begin
        check = 1'b0;
        ibeat = 0;
    end
    else begin
        ibeat = next_ibeat;
        if (_music) begin
            check = check;
            if (check == 1'b0) begin
                check = 1'b1;
                ibeat = 0;
            end
        end
        else begin
            check = check;
            if (check == 1'b1) begin
                check = 1'b0;
                ibeat = 0;
            end
        end
    end
end
```

music_example

music_example module 為製作樂譜的地方，利用判斷 ibeatNum，給予左右聲道對應的頻率，我在這邊加入了暫停與切歌的功能，如果是暫停的話就無需判斷 ibeatNum，直接輸出 silence，反之先判斷為第幾首歌後，再依據 ibeatNum 給予那首歌對應的頻率。

speaker

speaker module 為程式的 top module，我在這邊加入了七段顯示、LED 燈顯示與音量按鈕三個部分。

七段顯示器需要顯示當前音樂的音符，利用從 music_example module 得到的 freqR 來進行判斷，只要是 Do，不管是高音、低音還是升降，都顯示 C，其他音符也是以此類推，如果是 silence 狀態的話，則顯示-(dash)，接著再將判斷的結果傳入 seven segment 中執行輸出。

LED 燈顯示與音量大小是同步的，當 volUP 被按下時，如果音量還沒到 5(最大聲)，volume 則+1，LED 進行左移，並且在最右邊補上一個 1，當 volDown 被按下時，如果音量還沒到 0(最小聲)，volume

則-1，LED 進行右移，並且在最左邊補上一個 0，當靜音開關被開啟時，LED 和 volume 直接為 0。因為回到有聲音時，音量要與靜音前相同，因此 next 另外接了兩個 reg 出來，一種是真正輸出 module 的，會隨著靜音調整，另一個為記憶型的，不會隨靜音調整，用來記憶音量大小的，另外如果是 reset 時，LED 調整為 3 個燈，volume 大小為 3。

```
always @(*) begin
    if (freqR == `a || freqR == `ha || freqR == `ra || freqR == `rha || freqR == `da) tn = `AA;
    else if (freqR == `b || freqR == `lb || freqR == `hb || freqR == `db) tn = `BB;
    else if (freqR == `c || freqR == `hc || freqR == `rc || freqR == `rhc || freqR == `hhc) tn = `CC;
    else if (freqR == `d || freqR == `hd || freqR == `rd || freqR == `rhd
    || freqR == `ld || freqR == `lhd) tn = `DD;
    else if (freqR == `e || freqR == `he || freqR == `le || freqR == `lhe) tn = `EE;
    else if (freqR == `f || freqR == `hf || freqR == `rf || freqR == `rhf) tn = `FF;
    else if (freqR == `g || freqR == `hg || freqR == `rg || freqR == `rhg
    || freqR == `lg || freqR == `lhg) tn = `GG;
    else if (freqR == `silence) tn = 4'd10;
    else tn = 4'd10;
end
```

```
always @(posedge clk, posedge rst) begin
    if (rst) begin
        _led_vol <= 5'b0_0111;
        led_vol <= 5'b0_0111;
    end
    else begin
        _led_vol <= (_mute) ? 5'b0_0000 : next_led_vol;
        led_vol <= next_led_vol;
    end
end

always @* begin
    next_led_vol = led_vol;
    if (_volUP_onepulse) begin
        next_led_vol = (led_vol == 5'b1_1111) ? 5'b1_1111 : {led_vol[3:0], 1'b1};
    end
    else if (_volDOWN_onepulse) begin
        next_led_vol = (led_vol == 5'b0_0001) ? 5'b0_0001 : {1'b0, led_vol[4:1]};
    end
end
```

```
always @(posedge clk, posedge rst) begin
    if (rst) begin
        volume <= 3'd3;
        volume1 <= 3'd3;
    end
    else begin
        volume <= (_mute) ? 3'd0 : next_volume;
        volume1 <= next_volume;
    end
end

always @(*) begin
    next_volume = volume1;
    if (_volUP_onepulse) begin
        next_volume = (volume1 == 3'd5) ? 3'd5 : volume1 + 3'd1;
    end
    else if (_volDOWN_onepulse) begin
        next_volume = (volume1 == 3'b001) ? 3'b001 : volume1 - 3'd1;
    end
end
```

3.學習到的東西與困難

沒想到轉眼間，來到本學期最後一個 lab 了，回想在這 8 次的 lab 中，從最基礎的語法開始複習，至利用 FPGA 板各種外接的資源，學習到不少的技能，也累積許多可貴的經驗和教訓，希望這些種種能帶領我順利地完成 final project。

此次 lab 中在如此完整、架構清楚的 sample code 的帶領下，添加大小聲控制鍵、循環鍵等等功能，比起前幾次的 lab，其實不太困難，我認為最有挑戰性的是手刻樂譜的部分，身為接近音樂白癡的我，事前還先學習了怎麼看樂譜、節拍，另外雖然有用 C++程式協助我產生程式碼，但在輸入音符和節拍時，還是必須非常專注且小心，我就在過程中發生了 2 3 次小失誤，耗費了不少時間去做修改與調整，不過完成後，還是蠻感動的，雖然不會使用樂器演奏，但能利用程式譜出一段音樂，也十分不錯~