Logic Design 106061212 賴傳堯

Lab10

(1) Play 16 Notes Repeatedly (1 sec each)

I/O:

Inout: rst_n // low active reset

Inout: clk // 100MHz

Output: audio_mclk // master clock

Output: audio_lrck // left-right clock

Output: audio_sck // serial clock

Output: audio_sdin // serial audio data input

Pin:

W5 - clk

V17- rst_n

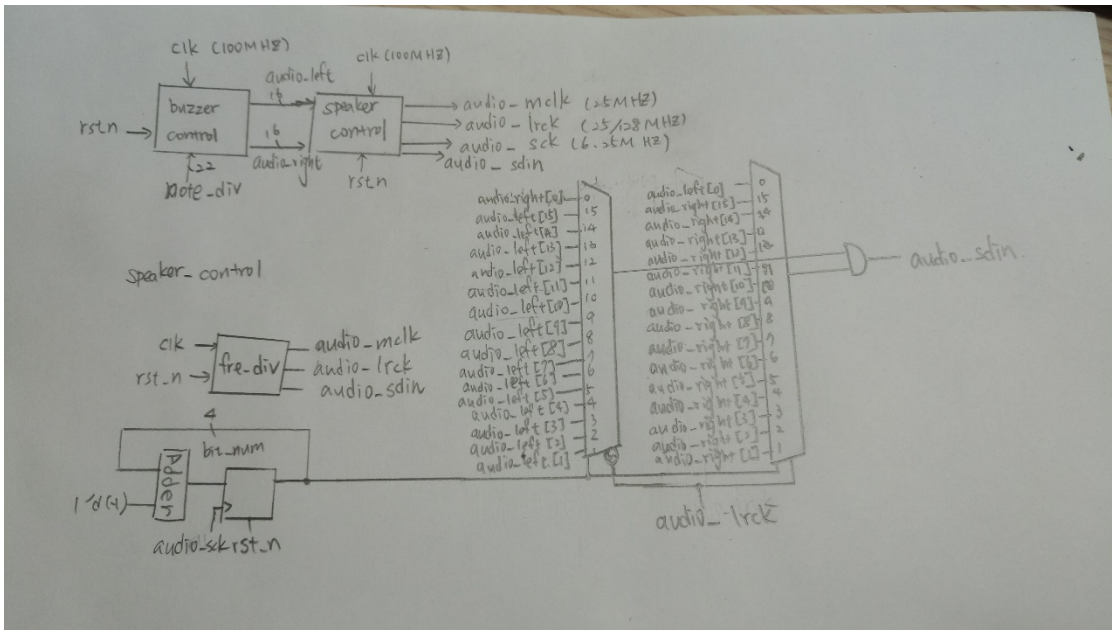
A14 - audio_mclk

A16 - audio_lrck

B15 - audio_sck

B16 - audio_sdin

Block diagram:



Discussion:

題解&思路&作法解釋:

這題要循環撥放 16 個音符、每個持續一秒。首先做除頻器獲得頻率為 1Hz 的 clk，接著許 Flip-Fiop，讓 4bit 的 note_cnt 每秒+1，利用其剛好可以數 16 個數字(0~15)並且不斷循環的特性，透過 case 判斷，讓每個數對應到一個

note_div 的數字(note_div 計算方式:100M/2/所求頻率)。最後將 note_div 傳進 buzzer_control 就能產生對應頻率的音階(原理同 8-1)。

Conclusion:

此題主要想法與 8-1 相同，只是多了循環播放的功能。而寫法也很直覺就是做頻率為 1 的 Flip-Flop 達到類似 ring counter 的功能，就能做得出來了，是簡單的延伸，大致上沒什麼問題。

(2) Keyboard-Controlled Note Player

I/O:

Inout: PS2_DATA

Inout: PS2_CLK

Inout: rst // high active reset

Input: clk // clock from the crystal

Input: rst_n // active low reset

Output: audio_mclk // master clock

Output: audio_lrck // left-right clock

Output: audio_sck // serial clock

Output: audio_sdin // serial audio data input

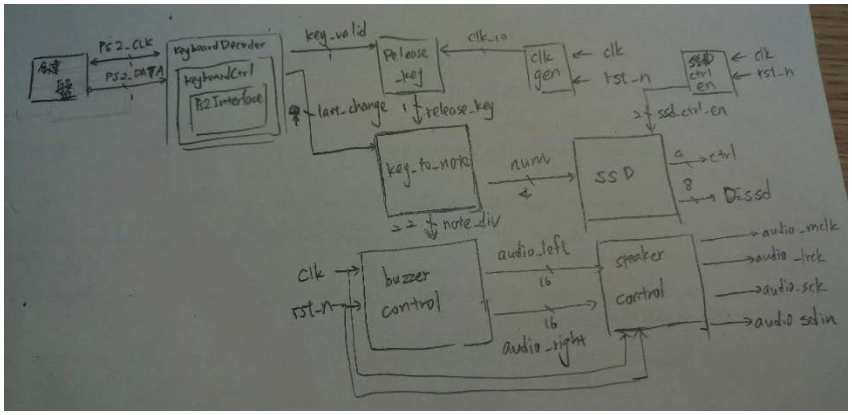
Output: [7:0]D_ssd

Output: [3:0]ctrl

Pin:

W5 - clk	W7 - D_ssd[7]	V17- rst_n
U18- rst	W6 - D_ssd[6]	A14 - audio_mclk
B17 - PS2_DATA	U8 - D_ssd[5]	A16 - audio_lrck
C17 - PS2_CLK	V8 - D_ssd[4]	B15 - audio_sck
W4 - ctrl[3]	U5 - D_ssd[3]	B16 - audio_sdin
V4 - ctrl[2]	V5 - D_ssd[2]	
U4 - ctrl[1]	U7 - D_ssd[1]	
U2 - ctrl[0]	V7 - D_ssd[0]	

Block diagram:



Discussion:

題解&思路&作法解釋:

這題要做鍵盤鋼琴，按 A、0~9、B~F 會播放相對應的音階，並在七段顯示器上顯示所播放的音。要判斷播放哪個音，就要先用判斷 last_change 判斷按下哪個鍵，用 case 把相對應頻率所要的值的 assign 給 note_div，然後依樣傳給 buzzer_control 就行了。另外，判斷 last_change 的同時，也給定 num 一個相對應的編號，傳給 SSD，再在 SSD 中用 case 判斷，讓 D_ssd 顯示相對應的符號。

```
`define D 8'b00000011
`define o 8'b11000101
`define R 8'b00010001
`define e 8'b00100001
`define M 8'b00010011
`define i 8'b11011111
`define F 8'b01110001
`define a 8'b00000101
`define S 8'b01001001
`define L 8'b11100011
```

Do Re Mi Fa Sol La Si

Conclusion:

這題是實驗 8、9 的結合，是兩個不同功能模組的合併使用。雖然說這兩個實驗的功能、pin 腳皆不同，但基本上變數間的連接、互相利用並不複雜，就只是將前兩個實驗的基礎模型合併，再稍微修改已符合此題需求就行了，並不困難，沒有大問題。

(3) Single Tune / Double Tune Note Player

I/O:

Inout: PS2_DATA

Inout: PS2_CLK

Inout: rst // high active reset

Input: clk // clock from the crystal

Input: rst_n // active low reset

Input: double_tune

Output: audio_mclk // master clock

Output: audio_lrck // left-right clock

Output: audio_sck // serial clock

Output: audio_sdin // serial audio data input

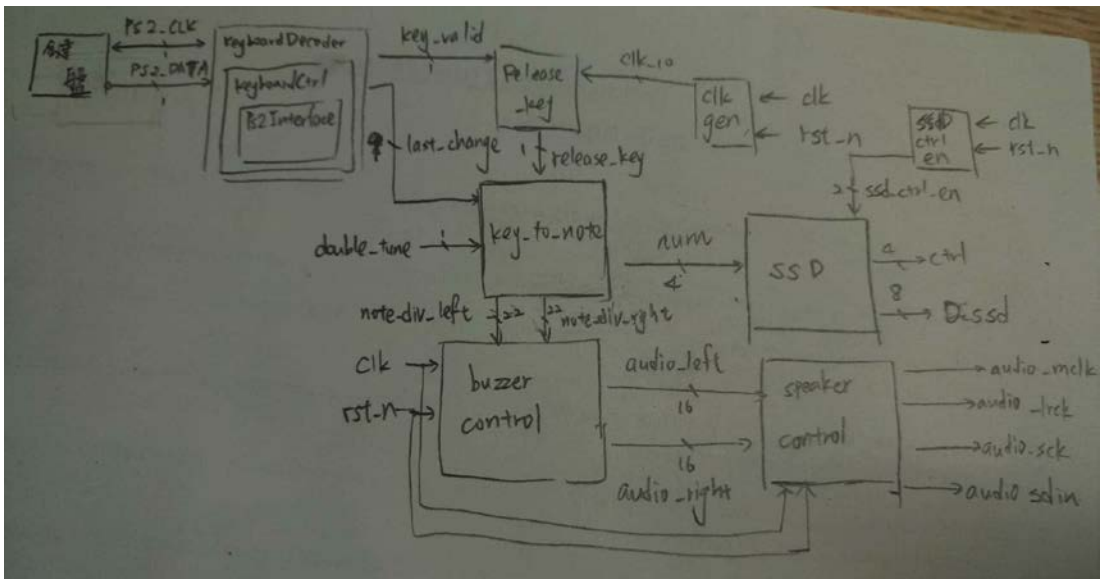
Output: [7:0]D_ssd

Output: [3:0]ctrl

Pin:

W5 - clk	W7 - D_ssd[7]	V16 - double_tune
U18- rst	W6 - D_ssd[6]	V17- rst_n
B17 - PS2_DATA	U8 - D_ssd[5]	A14 - audio_mclk
C17 - PS2_CLK	V8 - D_ssd[4]	A16 - audio_lrck
W4 - ctrl[3]	U5 - D_ssd[3]	B15 - audio_sck
V4 - ctrl[2]	V5 - D_ssd[2]	B16 - audio_sdin
U4 - ctrl[1]	U7 - D_ssd[1]	
U2 - ctrl[0]	V7 - D_ssd[0]	

Block Diagram:



Discussion:

題解&思路&作法解釋:

此提筆上提多了一個 DIP switch，功能為開啟時，播放雙聲道(左耳 Do 則誘耳 Mi，左耳 Re 則右耳 Fa，以此類推)，關閉時則是兩耳同音。基本上就直接沿用前一題的模組，並在 key_to_note 模組中導入 DIP switch 的訊號 (double_tune)，當 double_tune=0，note_div_left 和 note_div_right 有相同的值，double_tune=1 實則右耳恆比左耳高兩個音。而後把 note_div_left 和 note_div_right 傳給 buzzer_control，分別產生各自的頻率 l_clk 和 r_clk，取代原本的 b_clk，就完成了。

實驗困難與問題:

因為之前做過實驗 8，當時對老師給的這些模組，除了理解之外，多少也已經有些熟悉，知道哪部分城市是甚麼功能，因此要更改以符合此題要求並不算太難。

Conclusion:

完成實驗 10 的感想，莫過於感謝前兩個實驗的努力，才有辦法讓這次實驗這麼輕鬆!不禁讓人希望要是這學期實驗都如此順利該有多好!(雖然如果現在回去寫之前的實驗，應該也會比剛開學時要順利的多)

Reference:主要參考前兩個實驗(Lab08、Lab09)的做法。