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

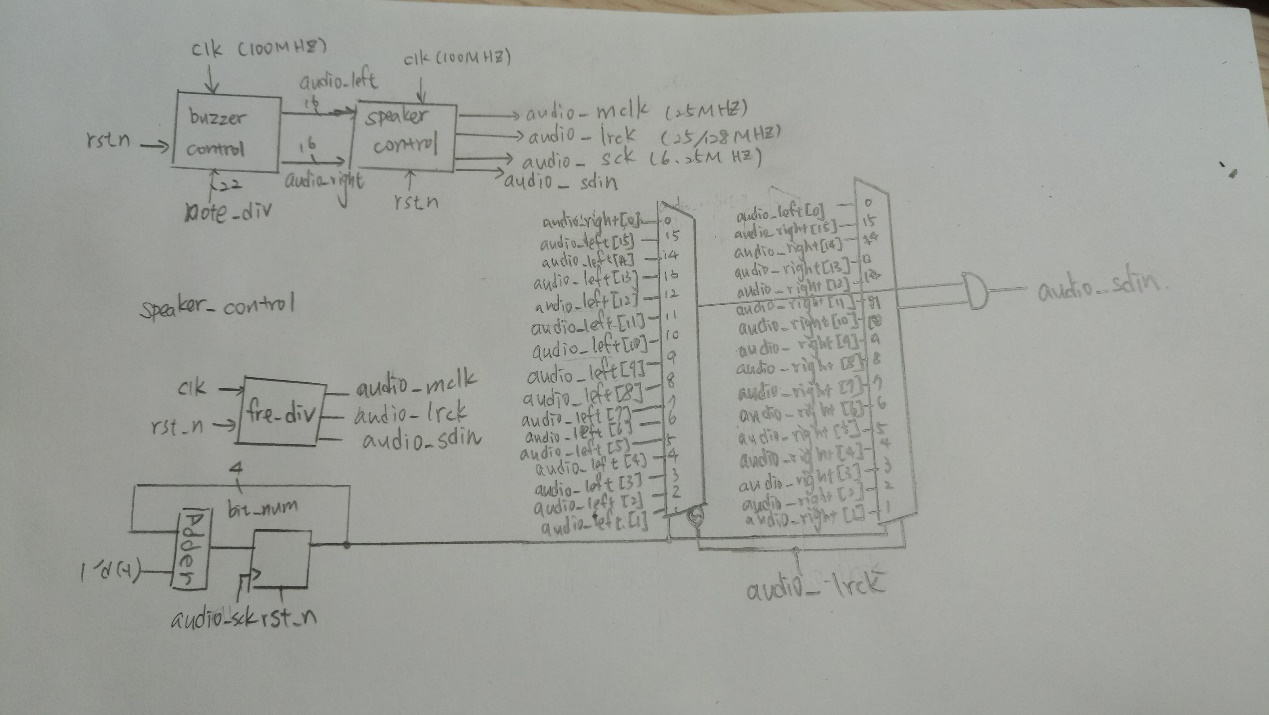
A14 - audio\_mclk

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。

1. 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:

V17- rst\_n

A14 - audio\_mclk

A16 - audio\_lrck

B15 - audio\_sck

B16 - audio\_sdin

W7 - D\_ssd[7]

W6 - D\_ssd[6]

U8 - D\_ssd[5]

V8 - D\_ssd[4]

U5 - D\_ssd[3]

V5 - D\_ssd[2]

U7 - D\_ssd[1]

V7 - D\_ssd[0]

W5 - clk

U18- rst

B17 - PS2\_DATA

C17 - PS2\_CLK

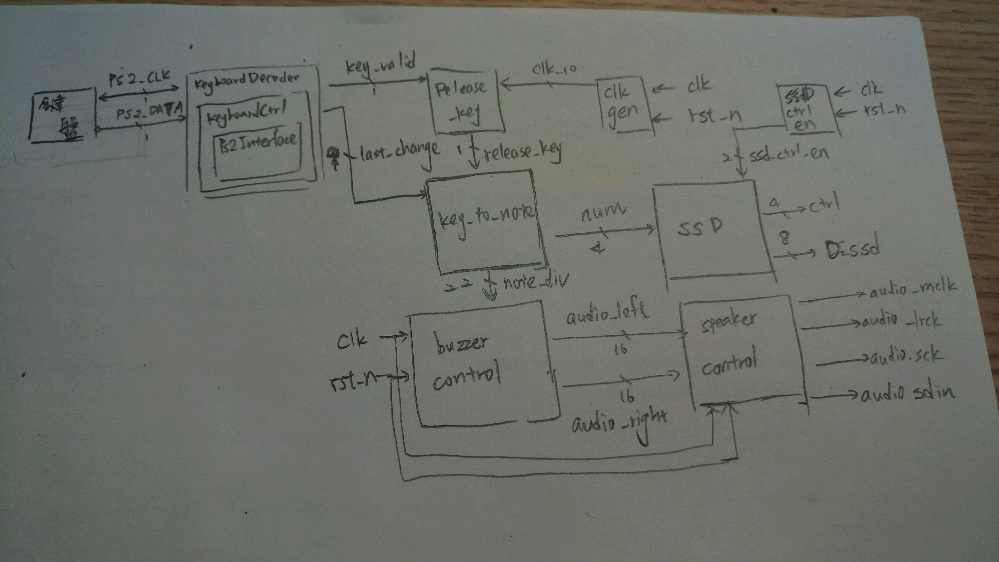
W4 - ctrl[3]

V4 - ctrl[2]

U4 - ctrl[1]

U2 - ctrl[0]

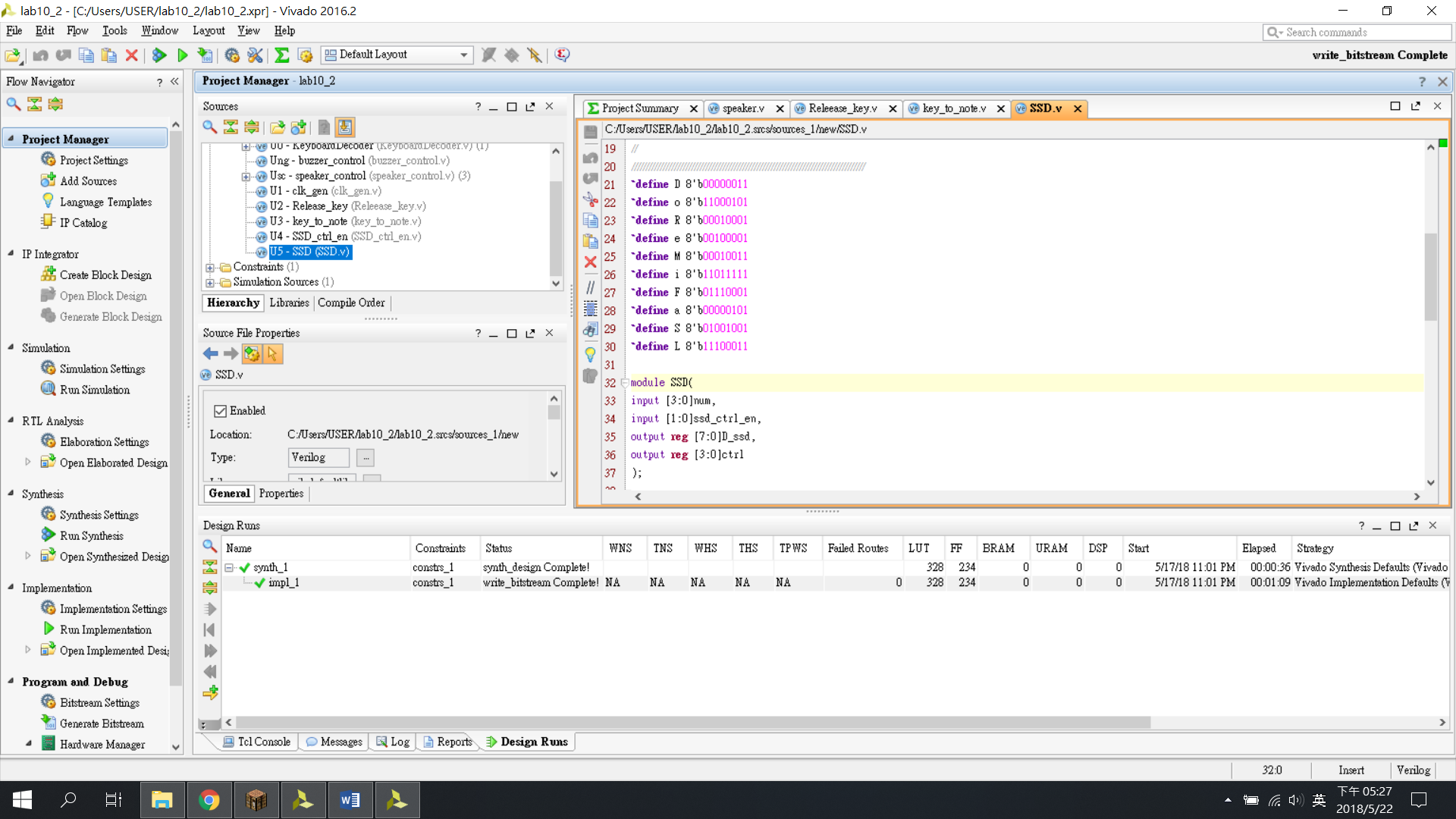
Block diagram:

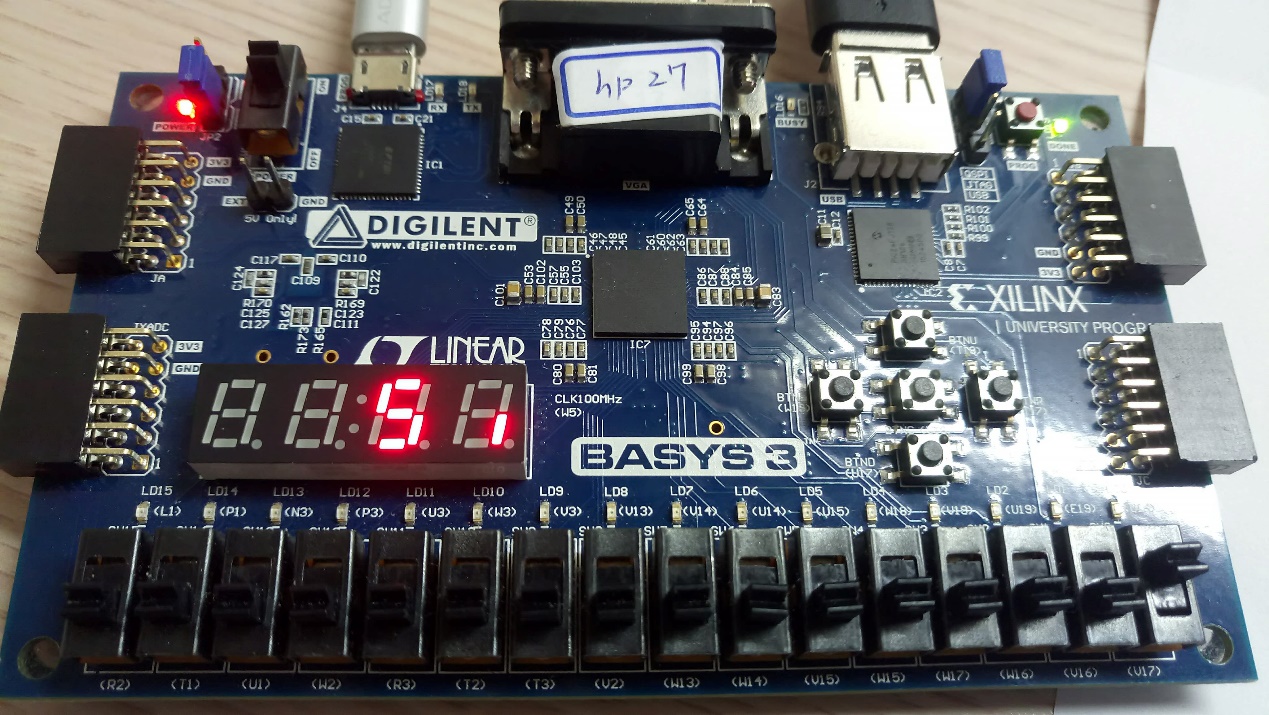
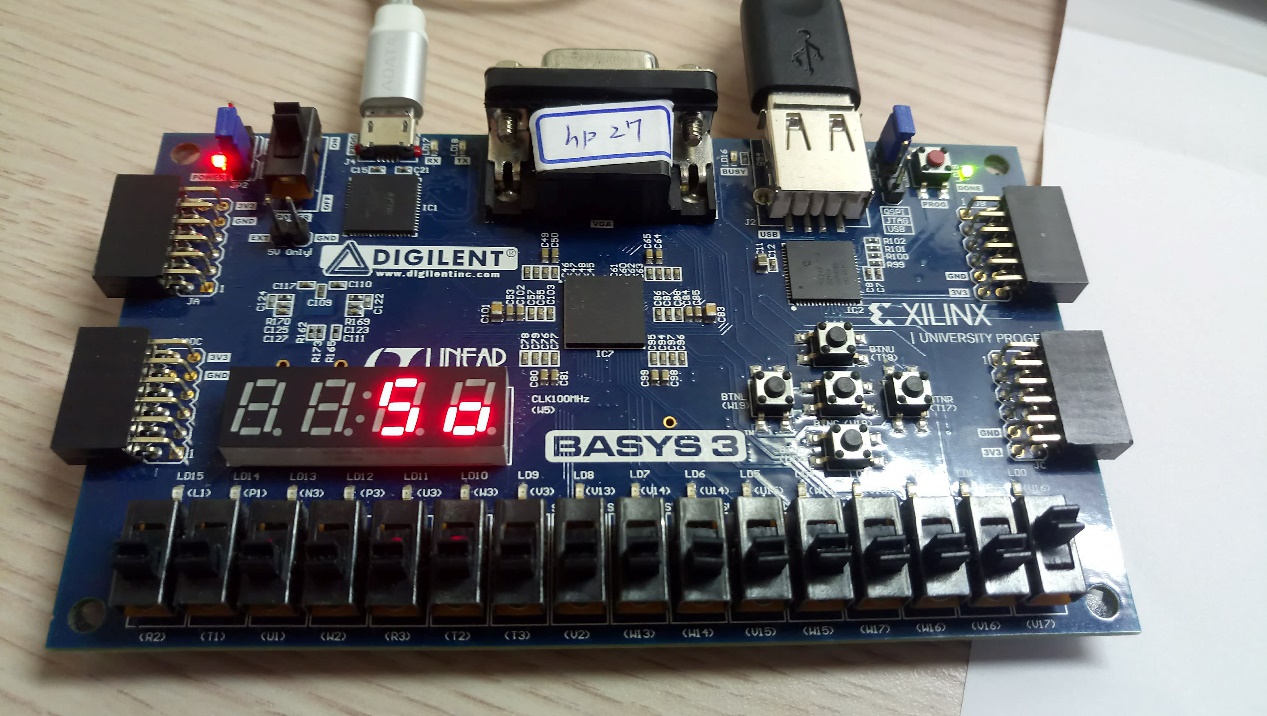
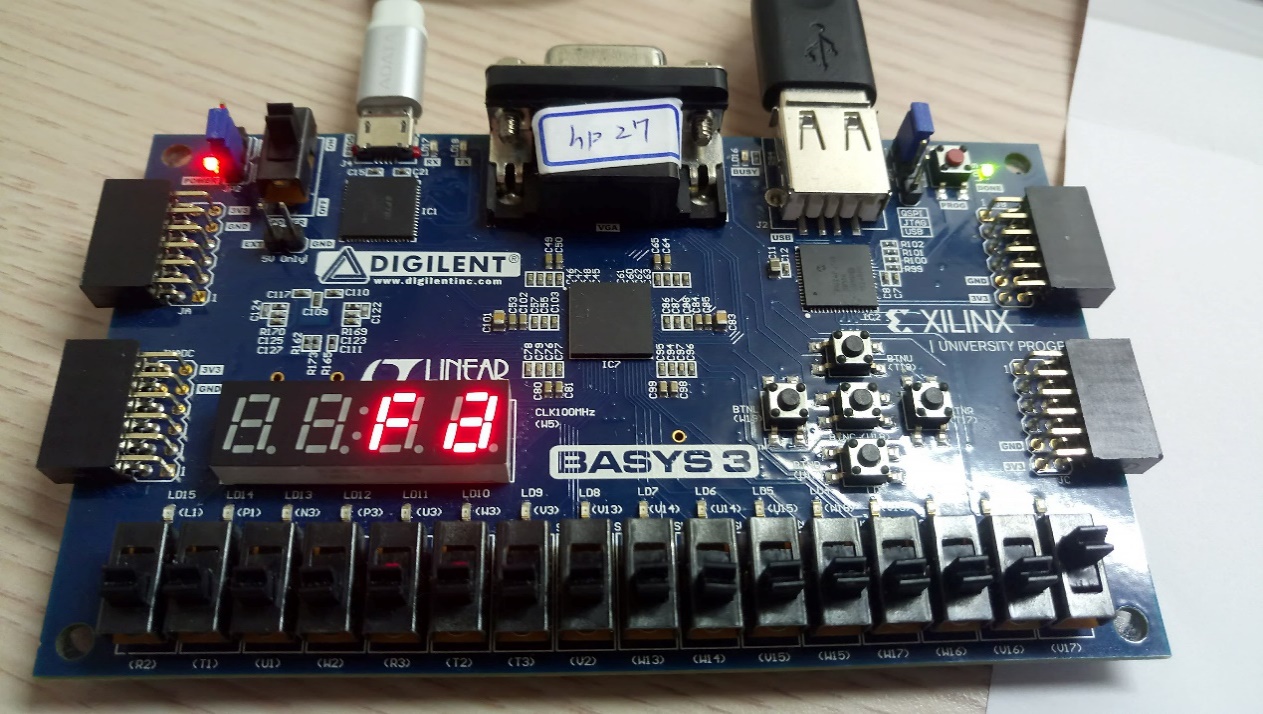
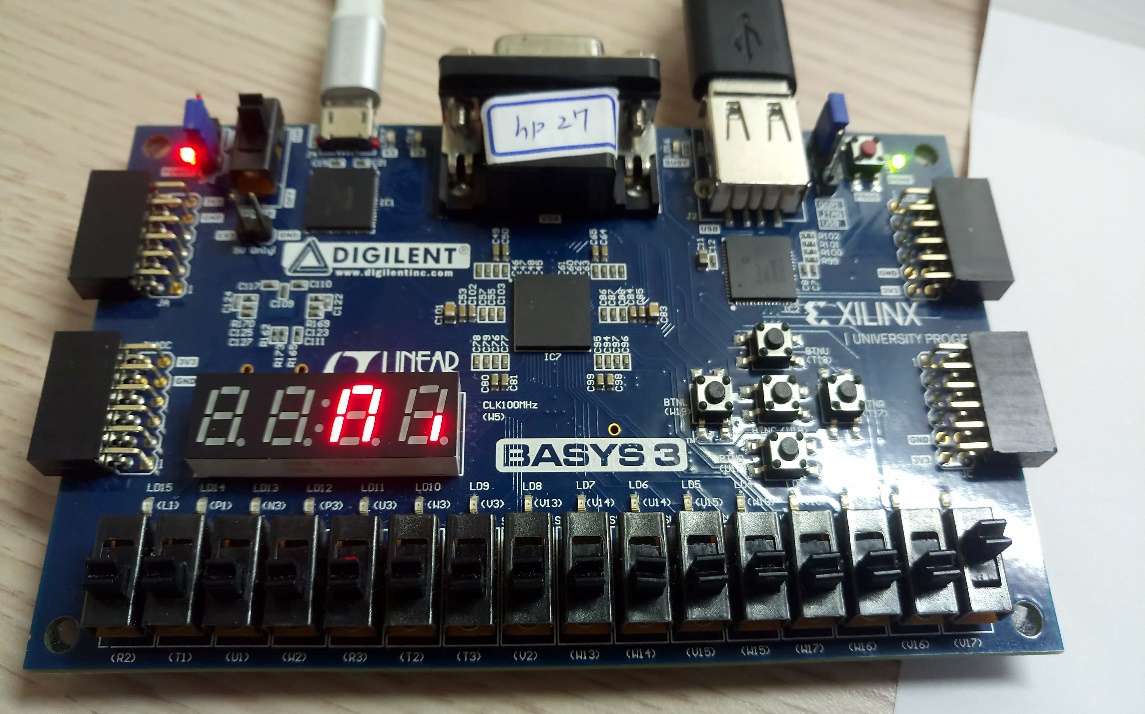
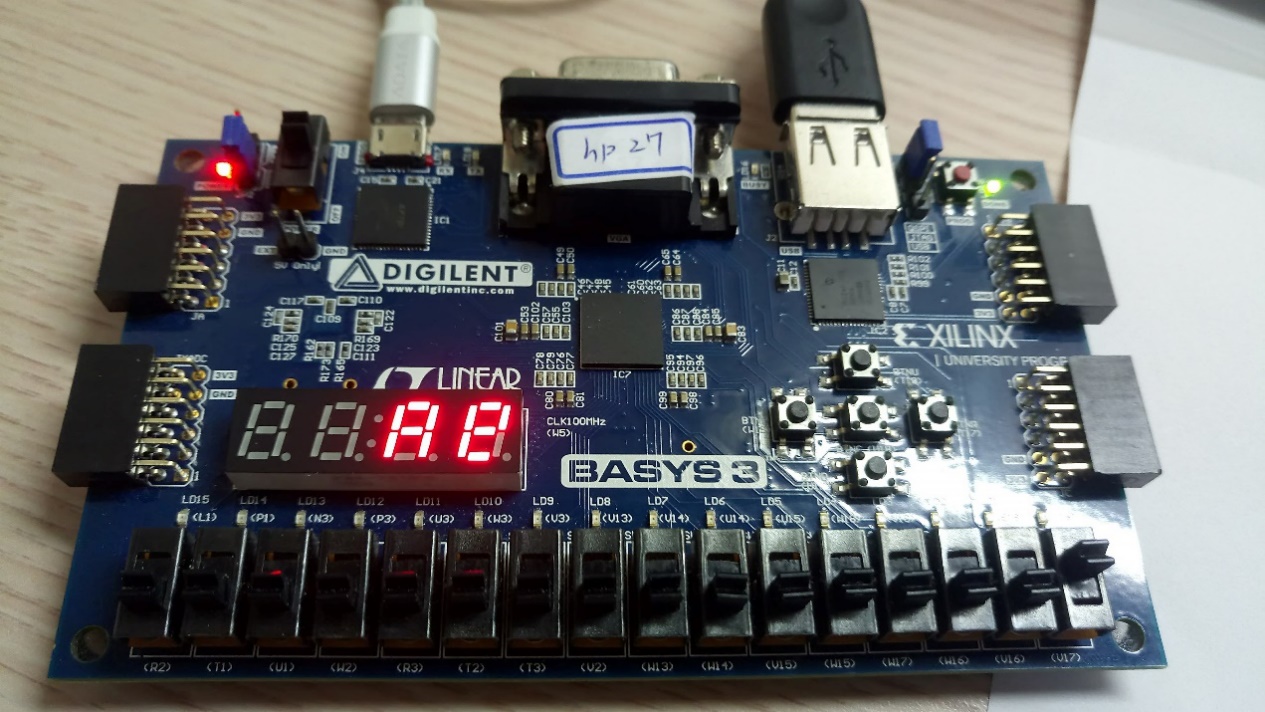
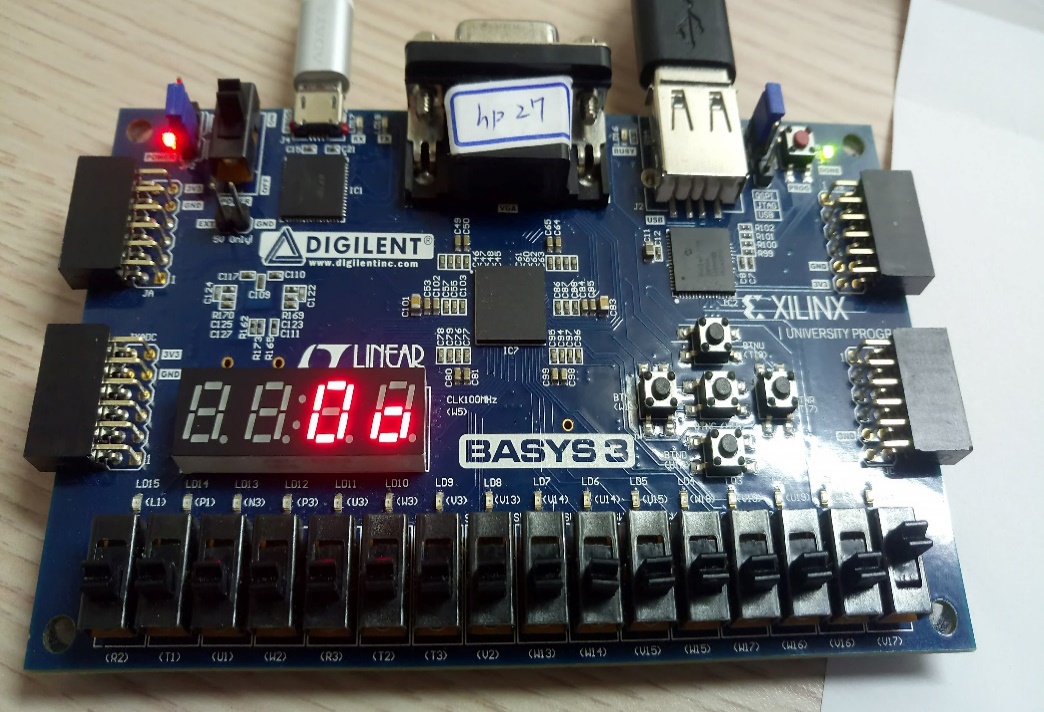


Discussion:

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。





Si

La

Sol

Fa

Mi

Re

Do

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。

1. 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:

V16 - double\_tune

V17- rst\_n

A14 - audio\_mclk

A16 - audio\_lrck

B15 - audio\_sck

B16 - audio\_sdin

W7 - D\_ssd[7]

W6 - D\_ssd[6]

U8 - D\_ssd[5]

V8 - D\_ssd[4]

U5 - D\_ssd[3]

V5 - D\_ssd[2]

U7 - D\_ssd[1]

V7 - D\_ssd[0]

W5 - clk

U18- rst

B17 - PS2\_DATA

C17 - PS2\_CLK

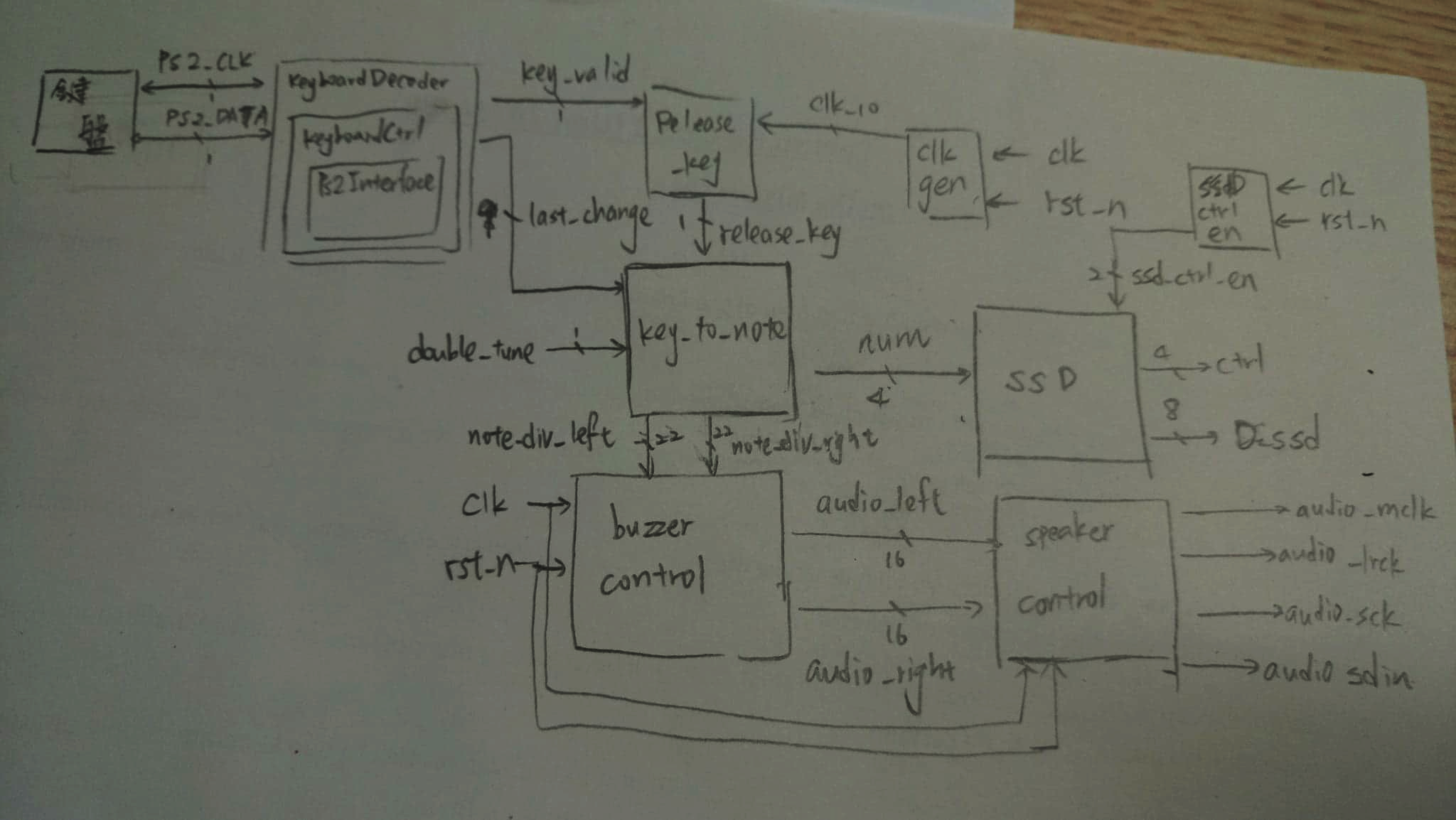
W4 - ctrl[3]

V4 - ctrl[2]

U4 - ctrl[1]

U2 - ctrl[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。