**Lab09 (109061256 陳立萍)**

**Lab9\_1:** **Play the sounds with 14 tones repeatedly based on the sound table. Every sound is played for one second..**

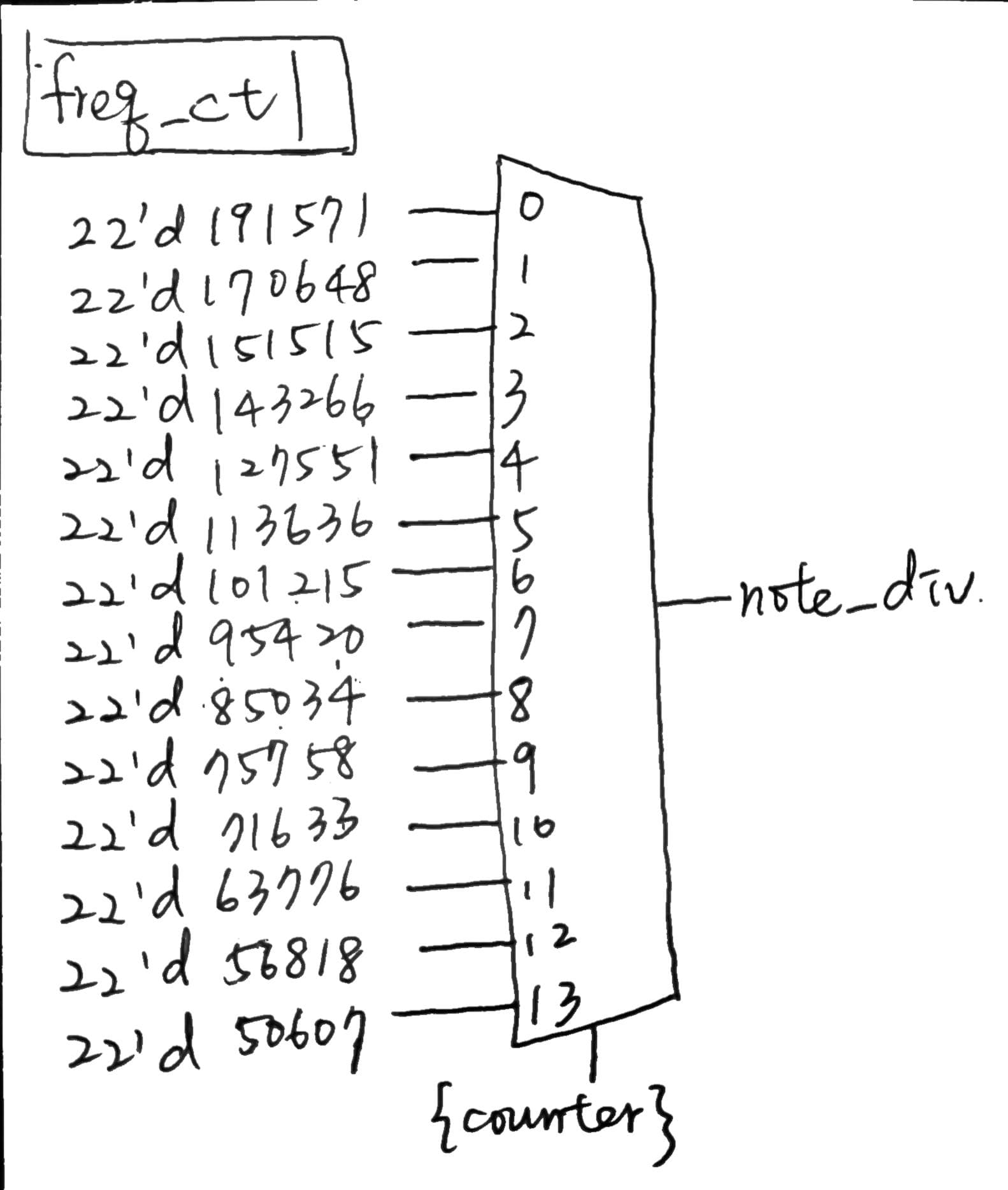
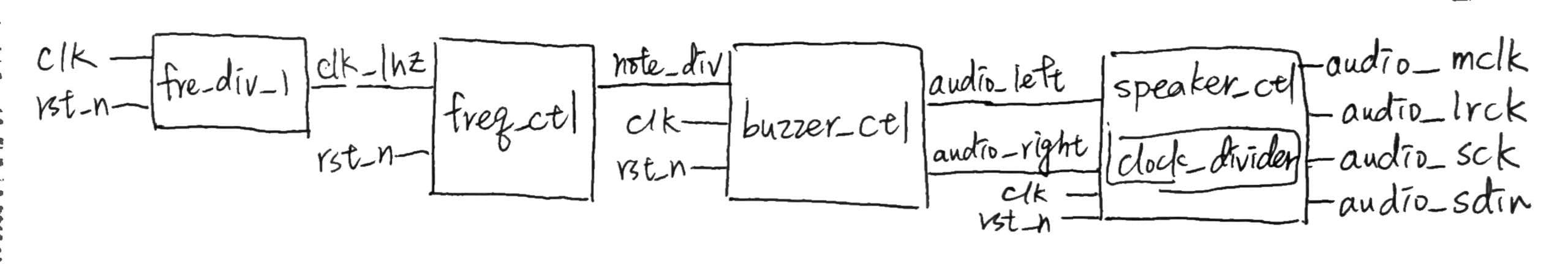
**Design Specification**

input : clk(100Mhz), rst\_n(reset)

output : audio\_mclk(25MHz), audio\_lrck(25/128MHz),

audio\_sck(6.25MHz), audio\_sdin(serial output)

**logic diagram**



**I/O pin assignment:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| audio\_mclk | audio\_lrck | audio\_sck | audio\_sdin | clk | rst\_n |
| A14 | A16 | B15 | B16 | W5 | R2 |

**Design Implementation**

**設計方法：**

這題用到共5個module，其中buzzer\_ctl.v：用來製造特定的頻率。speaker\_ctl.v：用來將audio\_left和audio\_right轉成1bit輸出 （做parallel to serial的處理）。clock\_divider.v：來除出三種頻率，分別為25MHz(mclk),25/128MHz(lrck),6.25MHz(sck/sampling clock)。上述在lab7已經介紹過，可以直接include進來使用，另外還會用到1hz除頻的module因此也將fre\_div\_1.v include進來。

這次新增的module是

＊freq\_ctl.v：主要是用來循環播放14個音階。首先先把1hz拉線進來當作clock，再用一個4bit的counter不斷循環從0數到13而這個counter會接到多工器作為選擇的判斷，0的時候note\_div輸出中音Do，1輸出中音Re……13輸出高音Si，接著歸零到0又開始播放中音Do，之後note\_div會輸出給buzzer\_ctl.v做特定頻率輸出，達到題目的循環播放要求。

最後用lab9\_1\_top.v將所有module連接在一起。

**Lab9\_2:** **Electronic Organ**

**2.1 Integrate the PC keyboard as the keyboard of the electronic organ. Keys c, d, e, f, g, a, b, C, D, E, F, G, A, B (two octaves from mid-Do) represent the sounds from low to high frequencies.**

**2.2 Display your playing sounds (Do, Re, Mi, Fa, So, La, Si) in the 7-segment LED**

**Design Specification**

inout : PS2\_DATA, PS2\_CLK

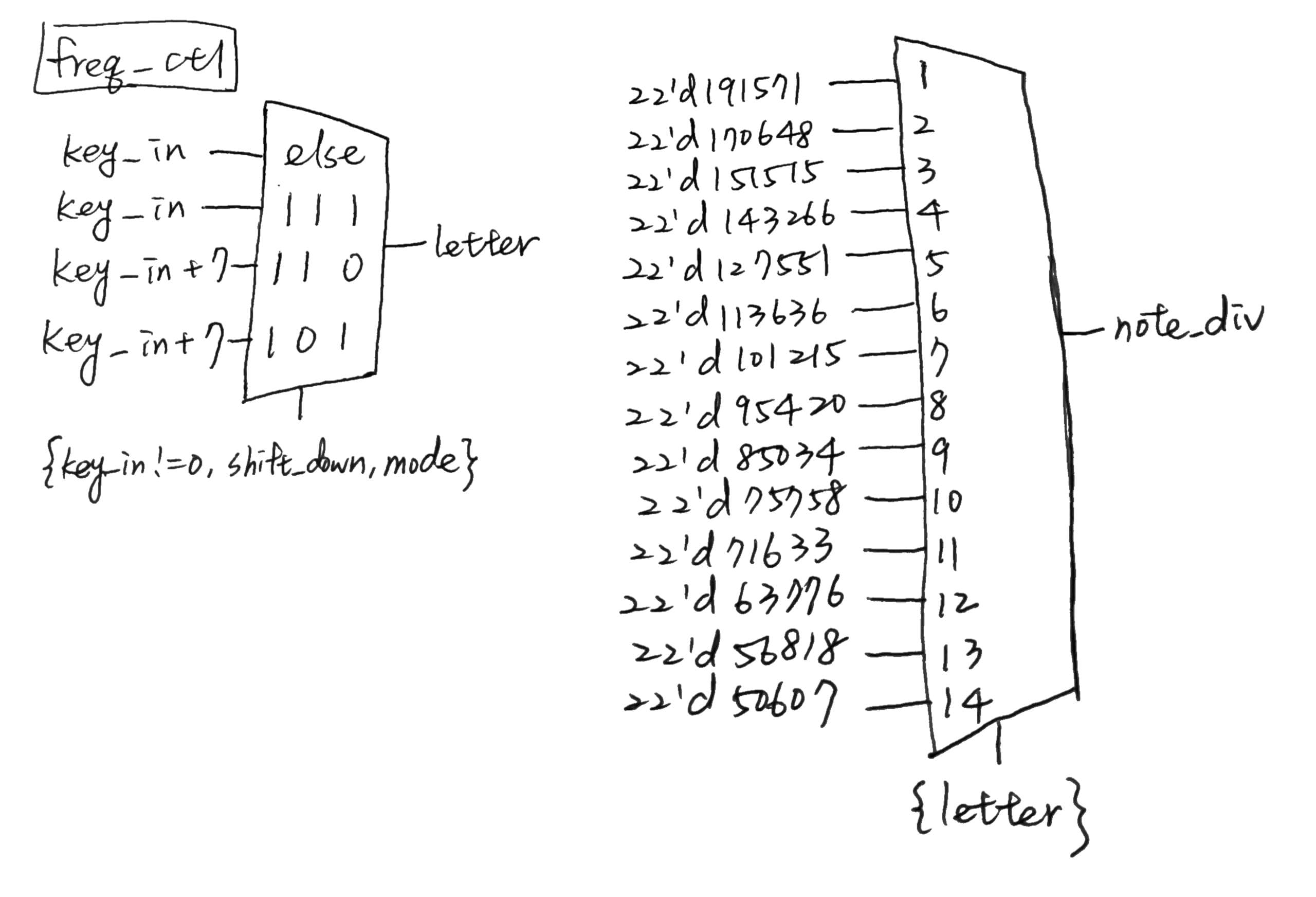
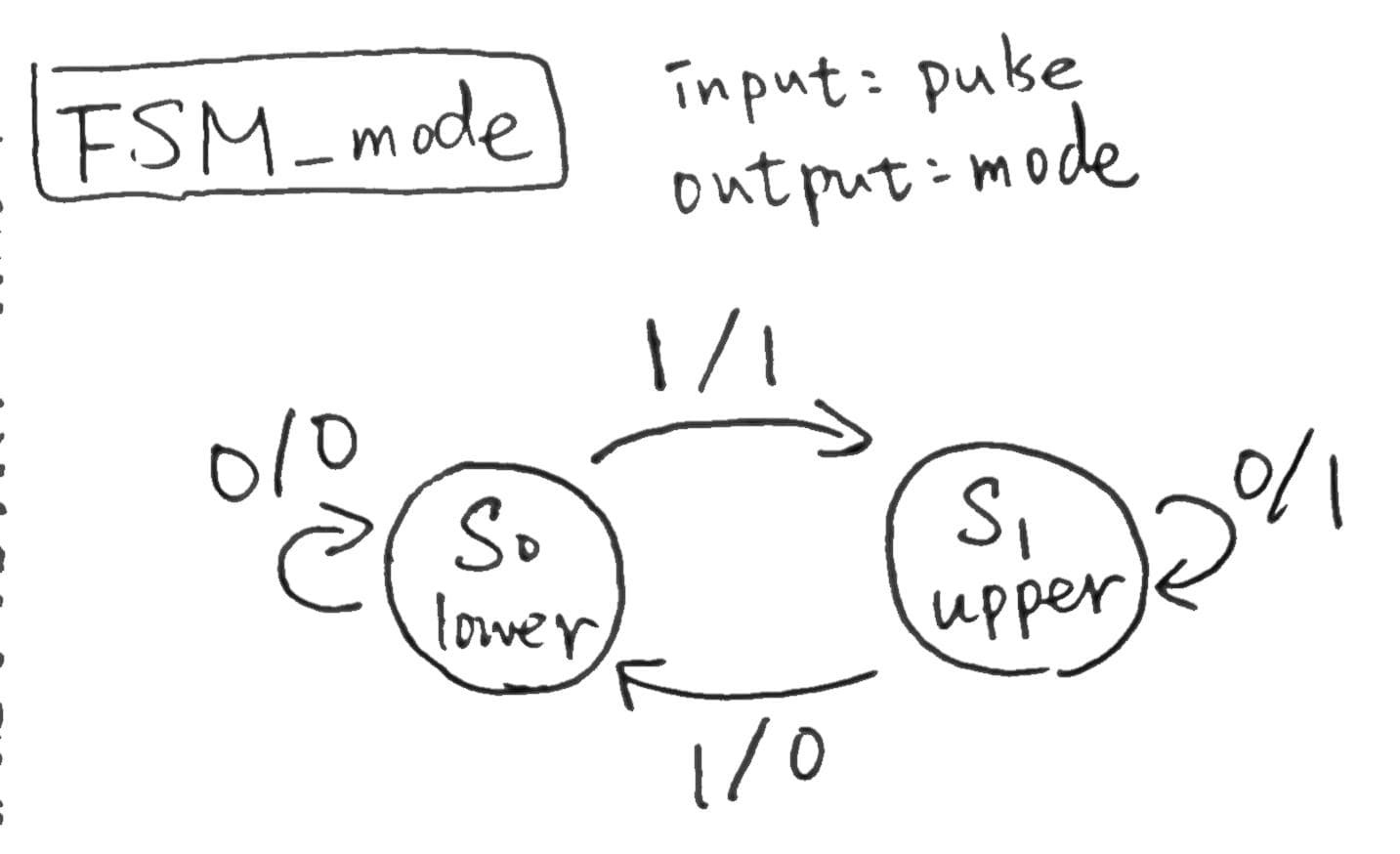
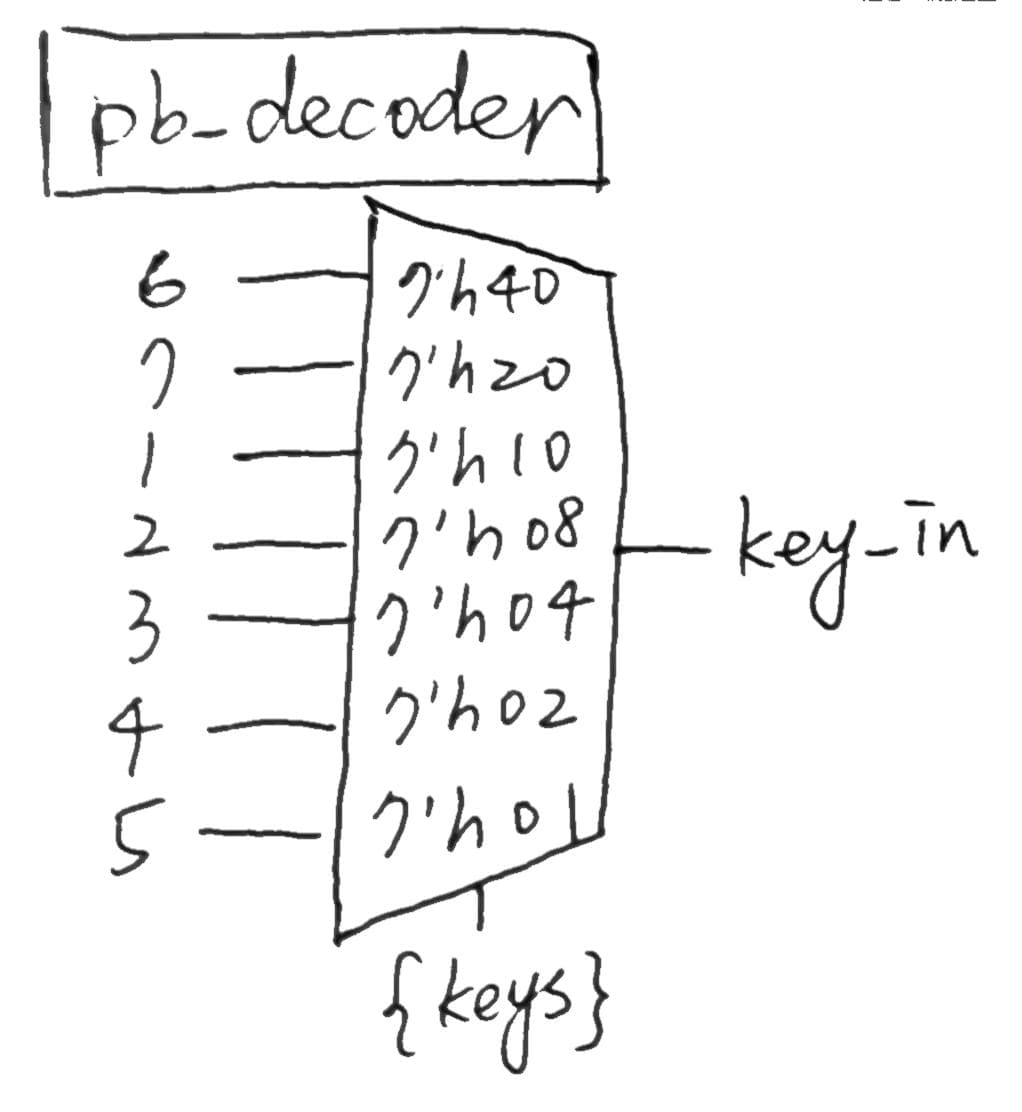
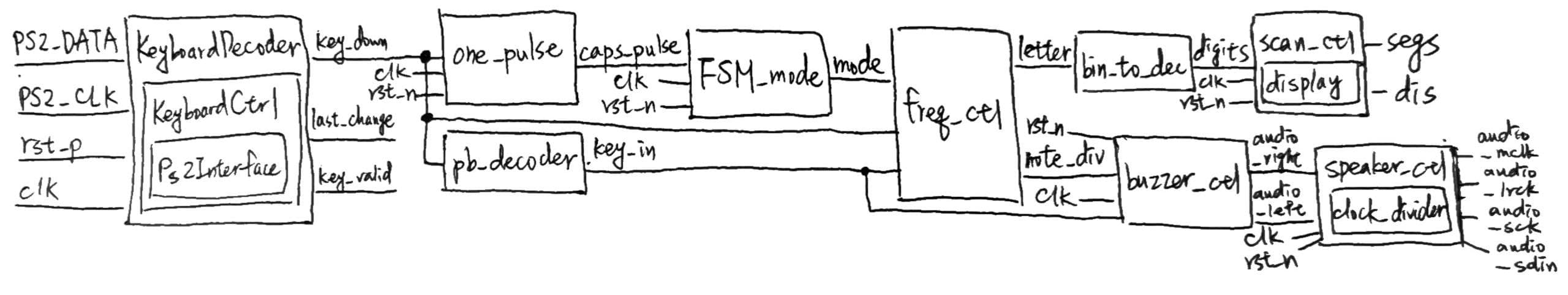
input : clk(100Mhz), rst\_n(reset)

output : audio\_mclk(25MHz), audio\_lrck(25/128MHz),

audio\_sck(6.25MHz), audio\_sdin(serial output)

[7:0]segs(七段段顯示器圖形), [3:0]dis(四個七段顯示器)

**logic diagram**



**I/O pin assignment:**

|  |  |  |  |
| --- | --- | --- | --- |
| PS2\_DATA | PS2\_CLK | rst\_n | clk |
| B17 | C17 | R2 | W5 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| segs[7] | segs[6] | segs[5] | segs[4] | segs[3] | segs[2] | segs[1] | segs[0] |
| W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |

|  |  |  |  |
| --- | --- | --- | --- |
| dis[3] | dis[2] | dis[1] | dis[0] |
| W4 | V4 | U4 | U2 |

|  |  |  |  |
| --- | --- | --- | --- |
| audio\_mclk | audio\_lrck | audio\_sck | audio\_sdin |
| A14 | A16 | B15 | B16 |

**Design Implementation**

**設計方法：**

這個題目一共用到14個module。其中KeyboardDecoder.v：鍵盤處理，one\_pulse.v：判斷是否重新按下按鍵（caps\_lock），產生一個pulse，buzzer\_ctl.v：用來製造特定的頻率，speaker\_ctl.v：用來將audio\_left和audio\_right轉成1bit輸出 （做parallel to serial的處理），clock\_divider.v：來除出三種頻率，分別為25MHz(mclk), 25/128MHz(lrck), 6.25MHz(sck/sampling clock)，bin\_to\_dec.v：二進位轉十進位，scan\_ctl.v：製造視覺暫留，display.v：七段顯示器輸出，之前已介紹過，不再贅述，此次新增的module如下：

＊pb\_decoder.v：將鍵盤上abcdefg的key\_down組成7bit的組合訊號輸入進來，哪一個bit是1代表那個字母被按下，依序由多工器輸出key\_in編碼，c為1，d為2，e為3，f為4，g為5，a為6，b為7。

＊FSM\_mode：為一個FSM狀態機，用來判斷是否切換大小寫。狀態分為大寫（1）與小寫（0）狀態，由one\_pulse產生的大小寫按鍵pulse進來做狀態轉換的觸發。

＊freq\_ctl.v：這裡我多做了和lab8\_4一樣有shift的功能，用多工器來做letter暫存器的輸出選擇，多工器的選擇由pb\_decoder的輸出key\_in, shift\_down(key\_down[18]), FSM\_mode的輸出mode來判斷，分為四種狀況，1.key\_in不為0（有按鍵被按下）且有按下shift且mode是大寫，則輸出letter=key\_in（低音），2.key\_in不為0（有按鍵被按下）且有按下shift且mode是小寫，則輸出letter=key\_in+7（高音），3.有按下按鍵且mode=1則輸出letter=key\_in+7（高音），4.剩餘情況皆讓letter=key\_in（低音）。

之後這個letter暫存器會變成另一組多工器的note\_div輸出判斷，1是輸出中音Do……14輸出高音Si。

最後再用lab9\_2\_top.v將上述module連接。

**3.Lab9\_3(bonus): Playback double tones by separate left and right channels. If you turn one DIP switch off, the electronic organ playback single tone when you press keyboard (as in Prob. 2). If you turn DIP switch on, left (right) channels play Do(Mi), Re(Fa), Mi(So), Fa(La), So(Si) when you press the keyboard.**

**Design Specification**

inout : PS2\_DATA, PS2\_CLK

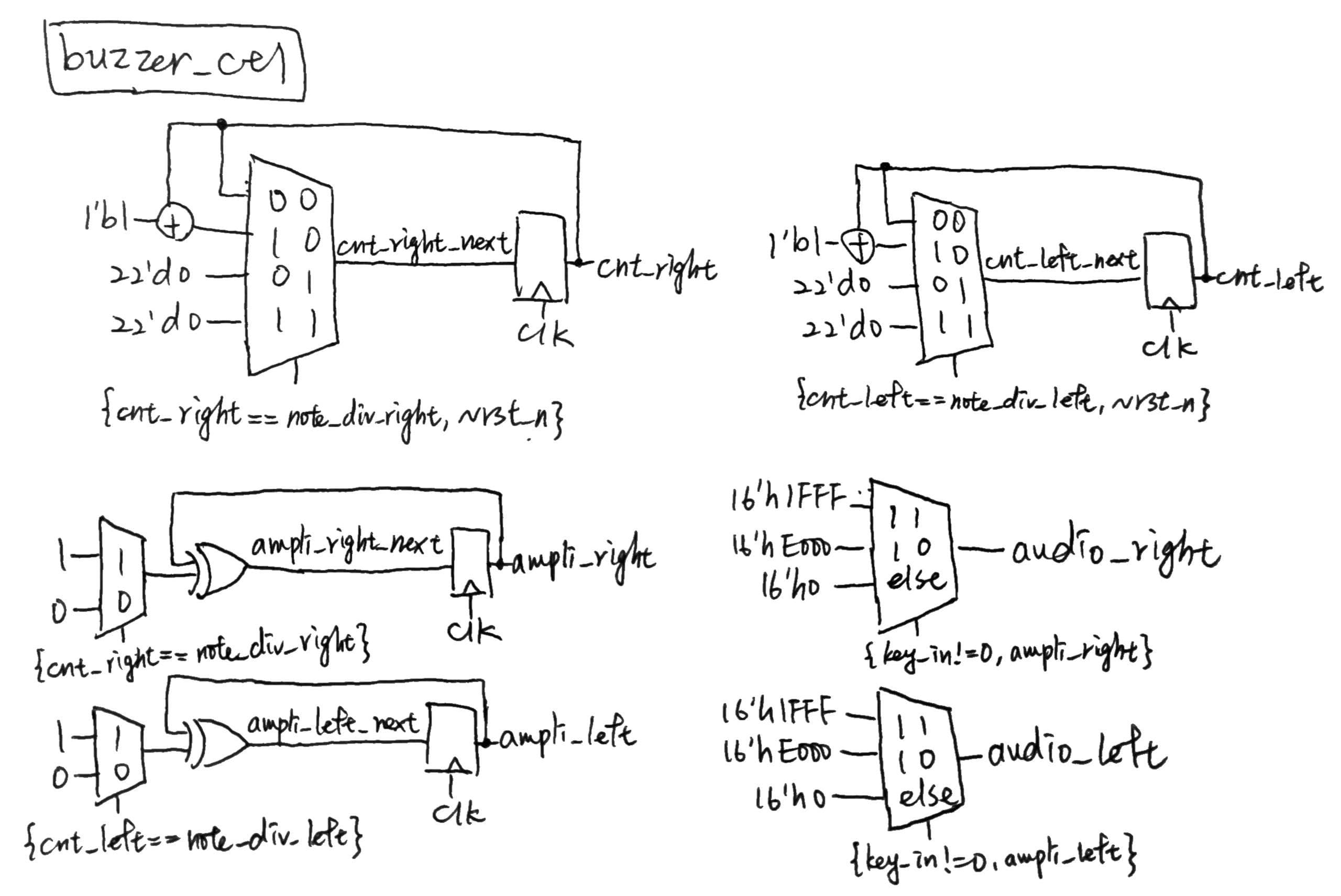
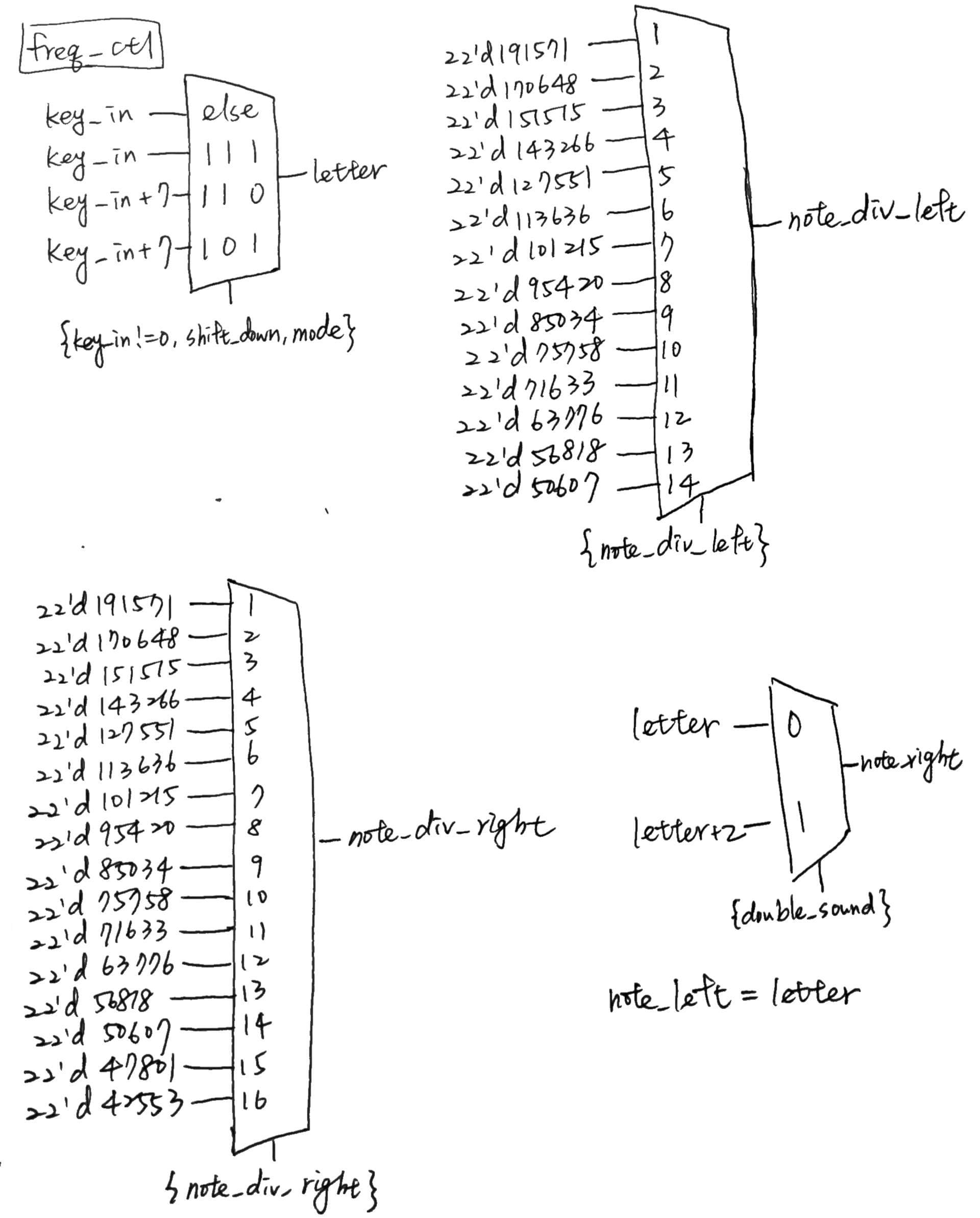
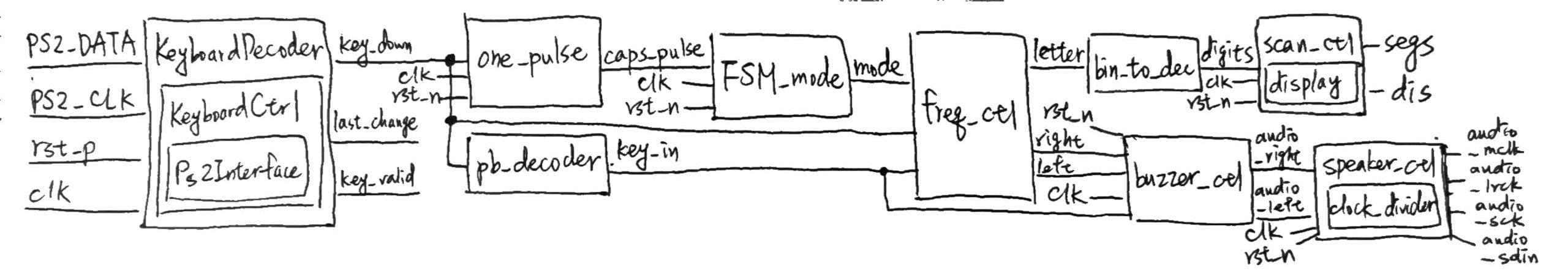
input : clk(100Mhz), rst\_n(reset) , double\_sound(select one/two sound)

output : audio\_mclk(25MHz), audio\_lrck(25/128MHz),

audio\_sck(6.25MHz), audio\_sdin(serial output)

[7:0]segs(七段段顯示器圖形), [3:0]dis(四個七段顯示器)

**logic diagram**



**I/O pin assignment:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PS2\_DATA | PS2\_CLK | rst\_n | clk | double\_sound |
| B17 | C17 | R2 | W5 | V17 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| segs[7] | segs[6] | segs[5] | segs[4] | segs[3] | segs[2] | segs[1] | segs[0] |
| W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |

|  |  |  |  |
| --- | --- | --- | --- |
| dis[3] | dis[2] | dis[1] | dis[0] |
| W4 | V4 | U4 | U2 |

|  |  |  |  |
| --- | --- | --- | --- |
| audio\_mclk | audio\_lrck | audio\_sck | audio\_sdin |
| A14 | A16 | B15 | B16 |

**Design Implementation**

**設計方法：**

這題大致上和第二題一樣，不一樣的地方只有buzzer\_ctl.v與freq\_ctl.v。

＊freq\_ctl.v：這裡和第二題不一樣的地方在於多了一個input: double\_sound來控制一個多工器，當double\_sound為0的時候代表左右耳同聲，因此右耳暫存器(note\_right)保持letter給的值，當double\_sound為1時代表右耳高左耳兩個音階，因此右耳暫存器的值為此時的letter再+2，左耳暫存器(note\_left)則保持原本的letter值。之後note\_left和note\_right會各自控制一組多工器來分別輸出note\_div\_right和note\_div\_left，也就是左右耳個別的頻率除數，給buzzer\_ctl.v。

＊buzzer\_ctl.v：和之前的不一樣在於輸入的除數有兩個，一個是給右耳的note\_div\_right另一個是給左耳的note\_div\_left，分別用來除出各自的頻率給audio\_right和audio\_left去輸出給speaker\_ctl.v。

最後再用lab9\_3\_top.v 整合上述各個module。

**Discussion**

**Lab9\_1**

這題其實非常簡單，把lab7的東西拿進來用而已。不過一開始我還是遇到一個問題，就是每當在reset後開始循環播放，結果reset後的第一個Do, Re播放非常快，其他時候的Do, Re都正常，後來思考後覺得是因為positive edge 還是negative edge觸發的問題，因為我的1hz輸出，一開始是從0開始，所以如果freq\_ctl設posedge，那麼在reset後半個1hz週期就會遇到正緣，就會觸發讓Do指撥放半秒就變Re。

**Lab9\_2**

這題其實就是把lab7和lab8組合在一起而已，所以難度不大，不過我還是不小心接錯線，打錯字，更不小心的是宣告的時候忘記給予變數bit數目，導致聲音多工器的判斷key\_in只有1個bit，所以一開始用耳機聽的時候按下不同的字母，都只有Do(1)和沒聲音(0)。

**Lab9\_3**

這題基本上只有把第二題小小修改，再加上一個DIP switch input做判斷，因此沒有太大的問題與難度。

**Conclusion**

這次的lab做的好快，基本上都只是在貼上lab7與lab8的module到這個lab，再加以修改而已。另外因為做的很快，所以原本第2、3題只有要求大小寫的功能，我還自己另外做了shift的功能，如此一來在玩電子琴可以更方便的切換音高。學期初的時候根本沒有想到會做出這麼龐大的lab與各種功能的結合，這個lab真的讓我很有成就感，下一個VGA感覺會很難，希望自己可進步更多。

**References**

教授授課投影片：語法運用，符號運用，設計觀念。