**Lab10\_1**

**Design Specification**

* For a speaker:

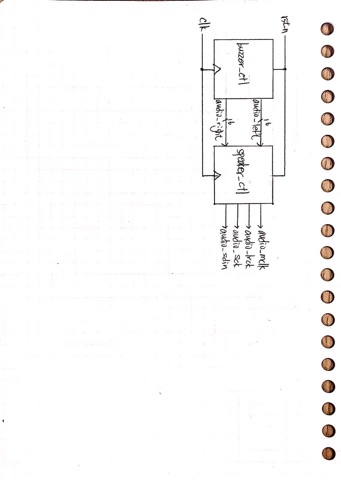
Input:

clk, rst\_n

Output:

audio\_mclk, audio\_lrck, audio\_sck, audio\_sdin

* Draw the block diagram of the design.



**Design Implementation**

* 本題由buzzer\_control與 speaker\_control組成。
* 本題與lab8概念相同，差別在於是自動發出聲音。因此我在buzzer\_ctl中加了一個頻率為一秒的clock，用以控制每秒輸出的聲音頻率。每過一秒後counter會加一，並將note\_div輸入下一個音調的頻率。
* I/O pin

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| I/O | audio\_mclk | audio\_lrck | audio\_sck | audio\_sdin | clk | rst\_n |
| VOC | A14 | A16 | B15 | B16 | W5 | V17 |

**Lab10\_2**

**Design Specification**

* For a keyboard speaker:

Input:

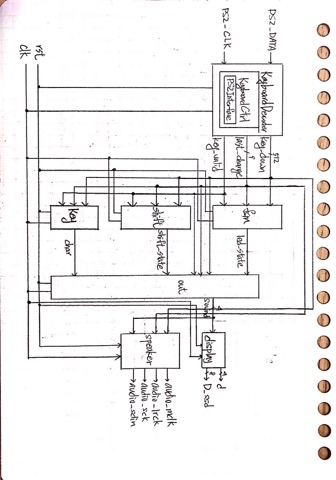
clk, rst

Inout: PS2\_CLK, PS2\_DATA

Output:

[3:0]d, [7:0]D\_ssd, audio\_mclk, audio\_lrck, audio\_sck, audio\_sdin

* Draw the block diagram of the design.



**Design Implementation**

* 本題由keyboardDecoder、fsm、shift、key、out、display、speaker組成
* 其中keyboardDecoder、fsm、shift、key、out皆與lab9\_4設計原理相近，在此不再贅述。
* Out為一個decoder，將key輸出的5 bits訊號轉成4 bits，目的是為了方便判定輸出14個音中的哪一個音。並透過接收fsm與shift的state，判斷是大寫還是小寫(高音或低音)。
* speaker接收keyboard輸出的key\_down與last\_change，以確定鍵盤被按下。其頻率的控制為透過out 輸出的sound判斷。
* display接收sound，為一個decoder，顯示14個音的唱名。在我的設計中，我將低音用7-segment的點表示，高音則沒有。
* I/O pin

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I/O | audio\_mclk | audio\_lrck | audio\_sck | audio\_sdin | clk | rst | PS2\_CLK | PS2\_DATA |
| VOC | A14 | A16 | B15 | B16 | W5 | U18 | C17 | B17 |
| I/O | D\_ssd[7] | D\_ssd[6] | D\_ssd[5] | D\_ssd[4] | D\_ssd[3] | D\_ssd[2] | D\_ssd[1] | D\_ssd[0] |
| LOC | W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |
| I/O | d[3] | d[2] | d[1] | d[0] |  |  |  |  |
| LOC | W4 | V4 | U4 | U2 |  |  |  |  |

**Lab10\_3**

**Design Specification**

* For a two-digit decimal adder/subtractor/multiplier:
* For a key board display:

Input:

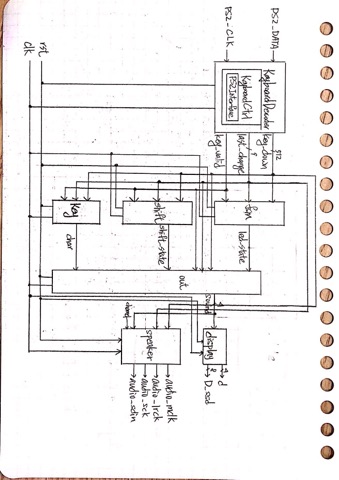
clk, rst, channel

Inout: PS2\_CLK, PS2\_DATA

Output:

[3:0]d, [7:0]D\_ssd, audio\_mclk, audio\_lrck, audio\_sck, audio\_sdin

* Draw the block diagram of the design.



**Design Implementation**

* 本題由keyboardDecoder、fsm、shift、key、out、display、speaker組成
* 本題與第二題相似，差別在於第二channel為和弦，因此多了一個channel的input，並在buzzer\_ctl中加了另一個note\_div。
* 為了製造左右耳輸出不同聲音，因此我在buzzer\_ctl中加了一組新的控制note\_div的功能，將其作為左耳的輸出。在我的設計中，是先判斷channel為0或1，0則是第二題的模式，左耳聲音頻率等於右耳；1則是和弦模式，左耳聲音頻率高右耳兩個音。
* I/O pin

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| I/O | audio\_mclk | audio\_lrck | audio\_sck | audio\_sdin | clk | rst | PS2\_CLK | PS2\_DATA |
| VOC | A14 | A16 | B15 | B16 | W5 | U18 | C17 | B17 |
| I/O | D\_ssd[7] | D\_ssd[6] | D\_ssd[5] | D\_ssd[4] | D\_ssd[3] | D\_ssd[2] | D\_ssd[1] | D\_ssd[0] |
| LOC | W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |
| I/O | d[3] | d[2] | d[1] | d[0] | channel |  |  |  |
| LOC | W4 | V4 | U4 | U2 | V17 |  |  |  |

**Discussion**

　　在第一題中，主要的問題是要怎麼控制每秒換一次聲音頻率。我用的方法是用陣列，設變數i讓他每隔一秒加一，並在一開始設定陣列內容的值。之後發現其實可以用counter去做，其實意思是一樣的。

　　第二題與第三題與lab8\_2很類似，差別只在於控制音頻的輸入是用keyboard。主要是將lab9\_4與lab8\_2結合。之所以會用很多個decoder是為了在表示上更為清楚。另外須注意輸入的鍵號與聲音的輸出須符合正確要求。

**Conclusion**

　　這次利用鍵盤作為輸入，輸出聲音，為前兩個lab的結合，沒什麼太大的麻煩，主要是電腦問題，因此一開始無法跑老師給的keyboardDecoder程式。