**Lab 8: Speaker**

**106061218 李丞恩**

**1. Audio-data parallel-to-serial module**

**Design Specification**

(1) Input:

Clk: 100MHz global clock。

rst: 開關，接在一個dip switvh上。

(2) Output:

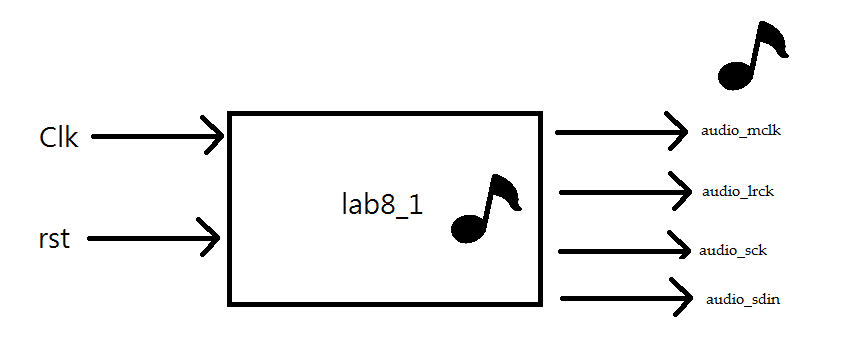
audio\_mclk: Master clock。

audio\_lrck: Lert\_right clock。

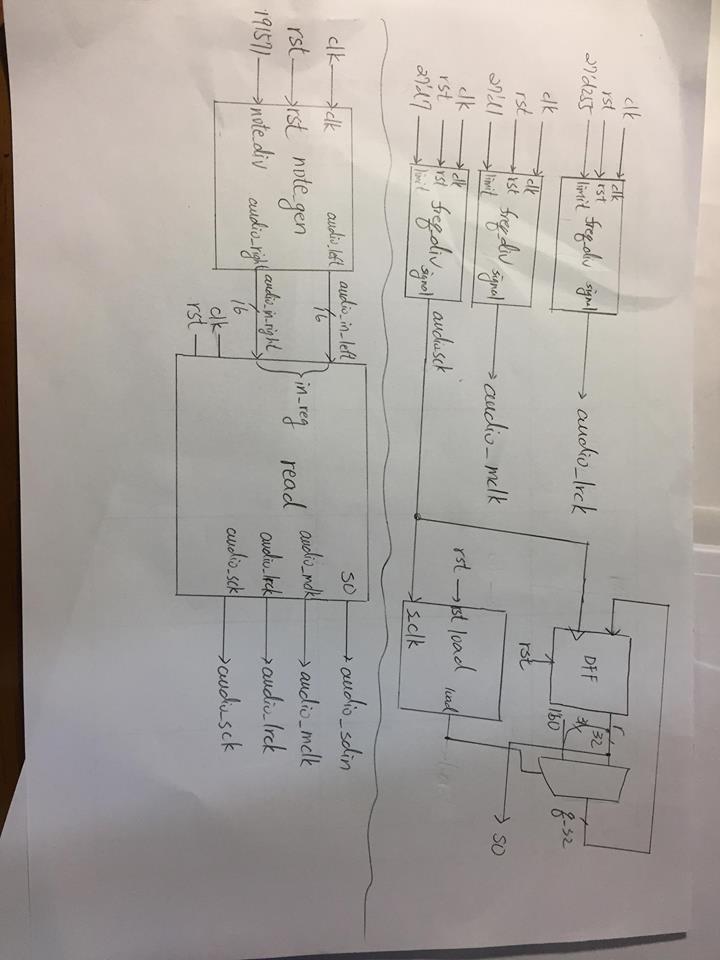
audio\_sck: serial clock。

audio\_sdin: serial data input。

(3) Block diagram:

****

**Design Implementation**



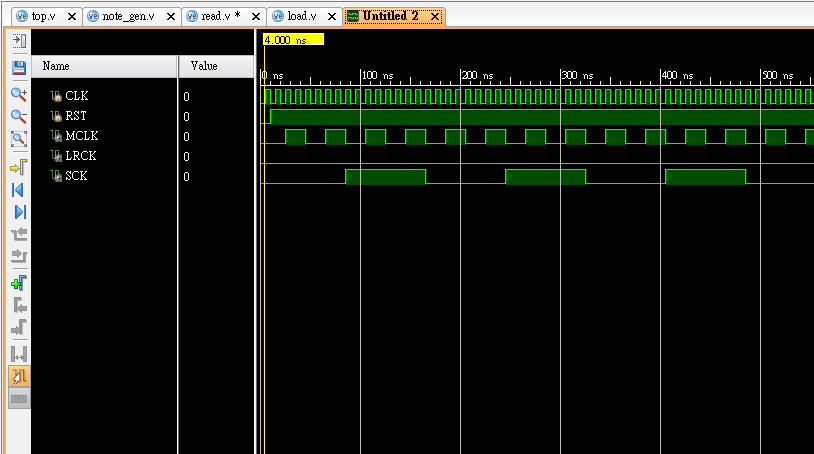
圖片的下半部是整個lab8\_1的接線圖，上半部是read的接線圖。其中每個module的功能如下：

Note\_gen：產生audio\_right跟audio\_left，照講義上面打就差不多了。

Read：負責產出三大clock與serial input。藉由同一個除頻器的module但餵給不同的上數極限limit，即可產出三大clock。再把sampling clock接到一個DFF即可產出serial output。

**Discussion**

波形模擬的結果如下：

****

****

(博君一笑，轉自FB粉專「彥斌的程式小教室」)

**2. Speaker control**

**Design Specification**

(1) Input:

clk: Global 100MHz clock。

rst: 開關，接在dip switch上。

Do: 接在push button上，按下去時產生Do。

Re: 接在push button上，按下去時產生Re。

Mi 接在push button上，按下去時產生Mi。

up\_in: 接在push button上，負責調高音量

down\_in: 接在push button上，負責調低音量。

(2) Output:

audio\_mclk: Master clock。

audio\_lrck: Lert\_right clock。

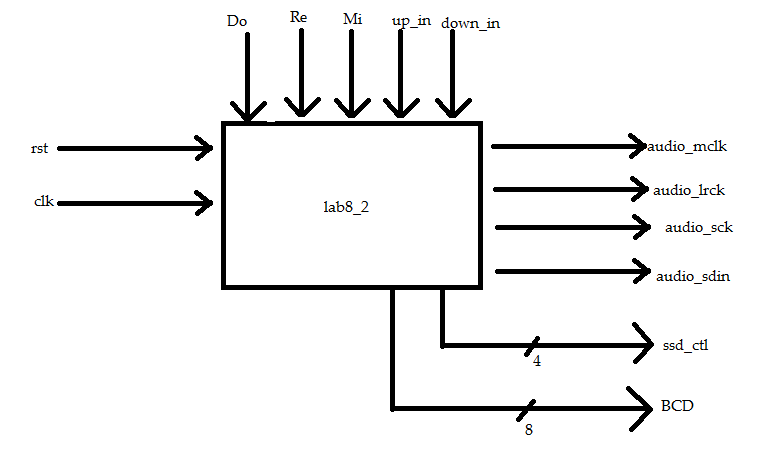
audio\_sck: serial clock。

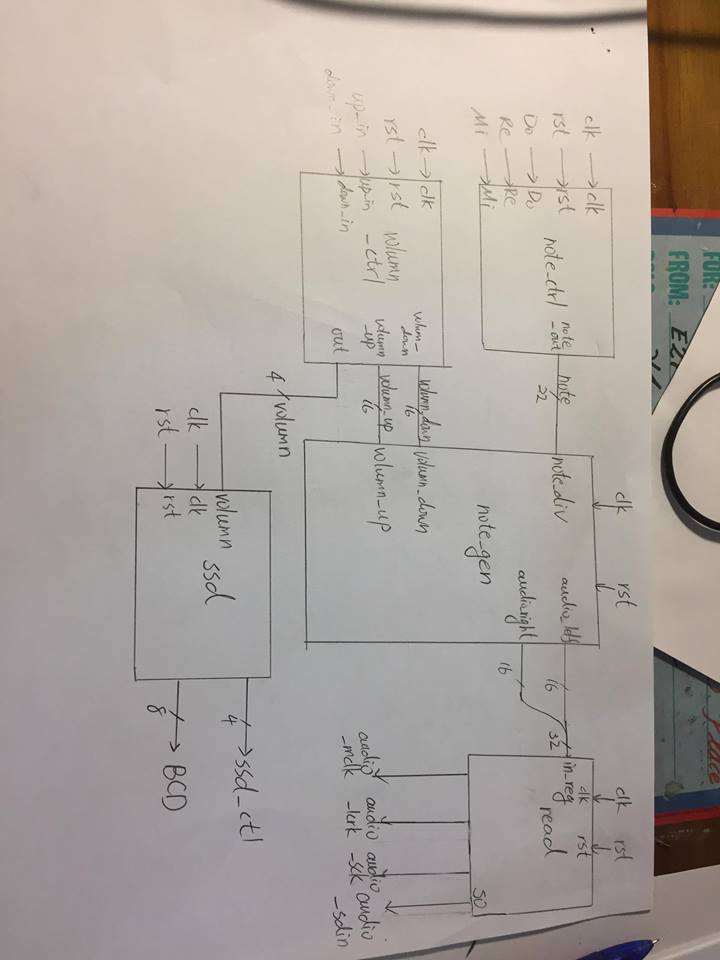
audio\_sdin: serial data input。

[3:0] ssd\_ctl: 控制七段顯示器。

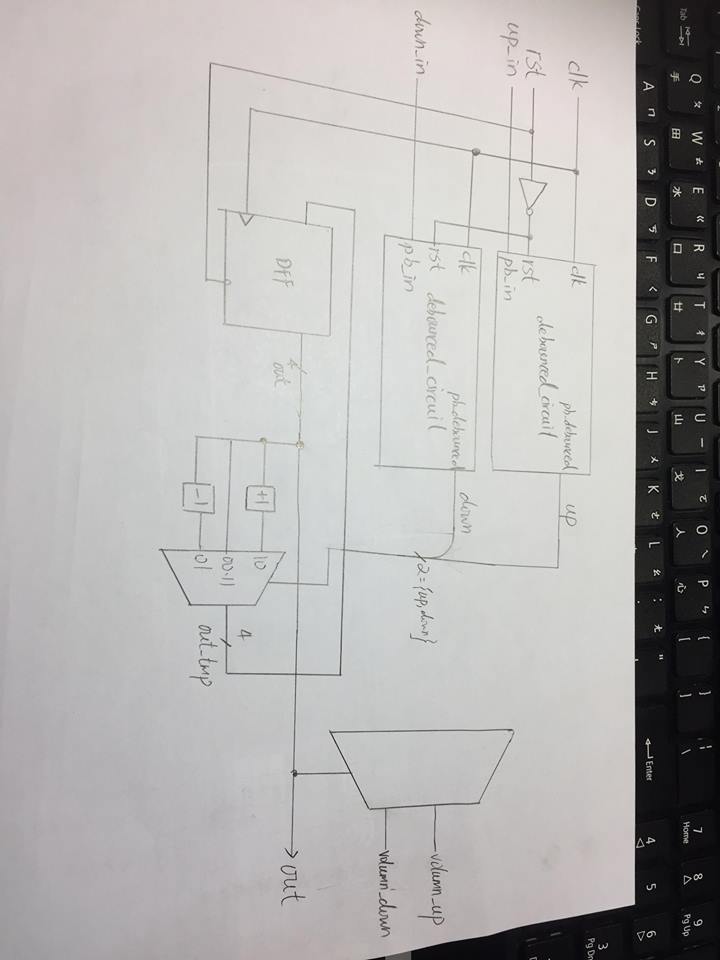
[7:0] BCD: 七段顯示器的各段。

(3) Block diagram:

****

**Design Implementation**

下圖是整個lab的block diagram：

下圖是volumn\_ctrl的block diagram：

各個module的功能如下：

Note\_ctrl: 產生Do、Re、Mi三個聲音的頻率。

Volumn\_ctrl: 輸入兩個控制音量的按鈕，控制播放的音量volumn\_up跟volumn\_down。

Note\_gen: 產生audio\_right跟audio\_left，照講義上面打就差不多了。

Read: 與第一題一模一樣，負責產出三大clock與serial input。藉由同一個除頻器的module但餵給不同的上數極限limit，即可產出三大clock。再把sampling clock接到一個DFF即可產出serial output。

Ssd: 顯示音量(1~16)。

**Discussion**

我自己對這一題有一些改進的想法。當我暗下控制音量的按鈕時，音量並不會一格一格往上升，而是隨機往上跳一個數字。放一個one\_pulse進去可能就可以了。

**Conclusion**

控制speaker的奧義就在read的那三個除頻器和名為load的shift register之中。其他的部分大體都是一些簡單的組合邏輯。



**References**

《上課的講義》，應該是馬席彬教授所著，應該是2018年

《黃元豪教授的講義》，應該是黃元豪教授所著，應該是2018年

《從漫畫了解FGO！》，リヨ著，Type-Moon，2015年，