Lab8 Report

8-1.audio-data parallel-to-serial module

●Design Specification

Input:

clk,(接上板子的原本的震盪頻率 W5)

rst_n, (控制整個功能的開關)

[15:0] audio_l(控制左邊音量)

[15:0] audio_r(控制右邊的聲音)

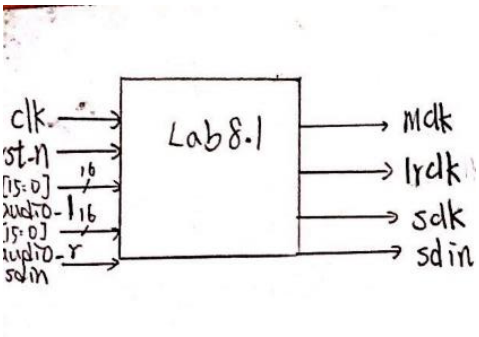
Output:

Mclk,(master clock，是最快的 clk 讓其他 clk 校準, 25MHz)

lrclk,(left-right clock，是要給左聲道和右聲道訊息的, 25MHz/128)

Sck,(serial clk,讓一個個訊號進去的頻率，比 lrclk 的頻率快 32 倍, 25Mhz/4)

Sdin(1 bit serial audio data output)



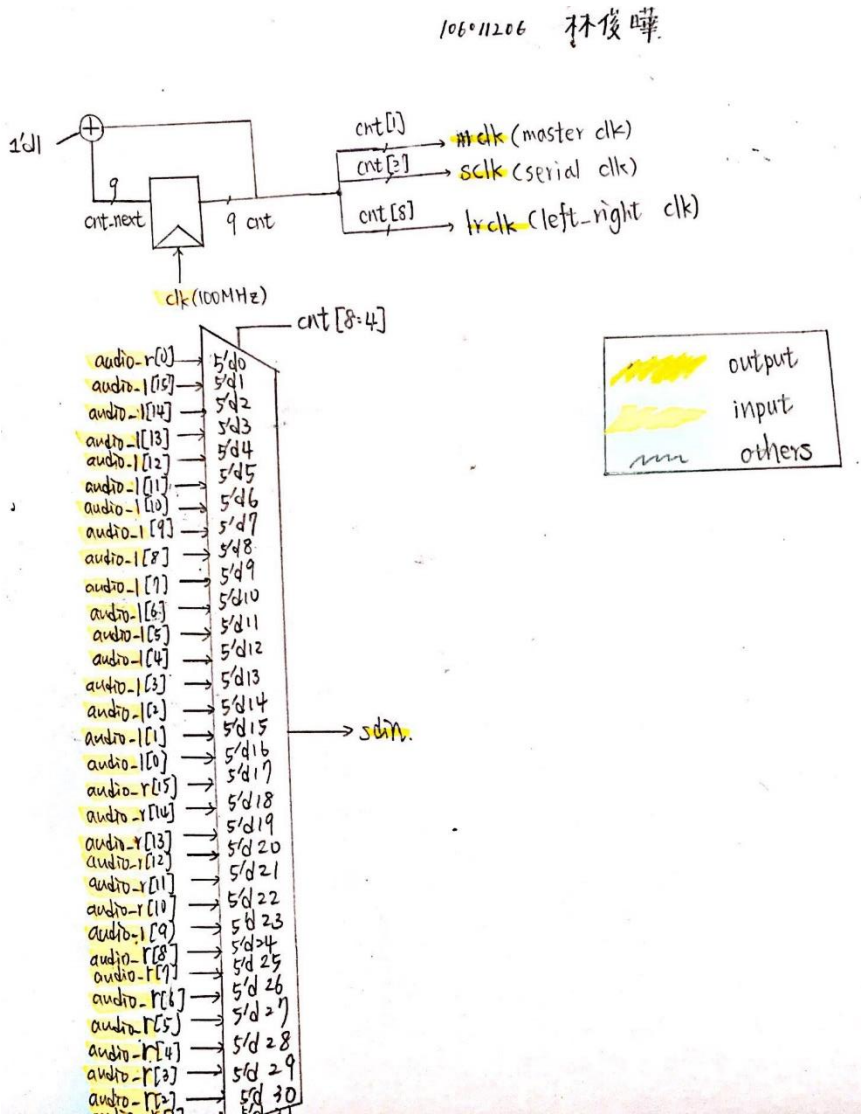
●Design Implementation

1.Outline

這個實驗主要是要為了之後做準備，之後所有會用到聲音的成品都需要用到這個 module，這也是我把這個 module 的名子設成 speaker 的原因。

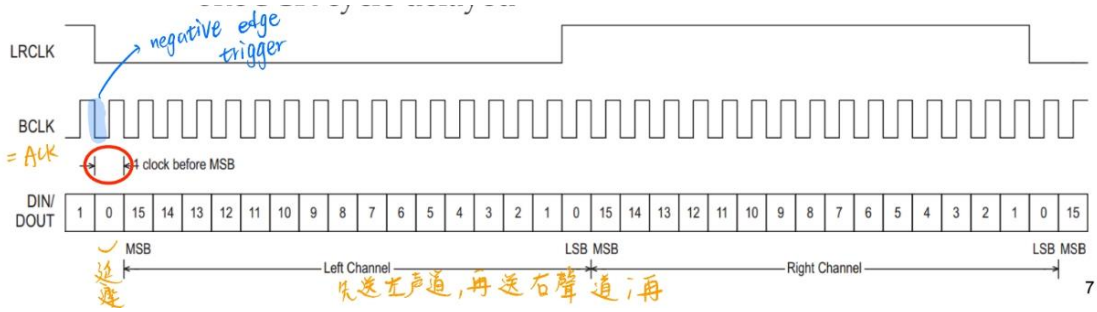
我一開始的想法是把所有有關於 frequency 的功能都放在 frequency divider 裡面，但是仔細思考後覺得應該要分開，不然每次都還是要把那些必要的 clk 接線再接到 speaker 的 module 裡面，input/output 那麼多反而複雜，不如把相關的功能寫再一起感覺看起來比較清爽。

2.Logic Diagram



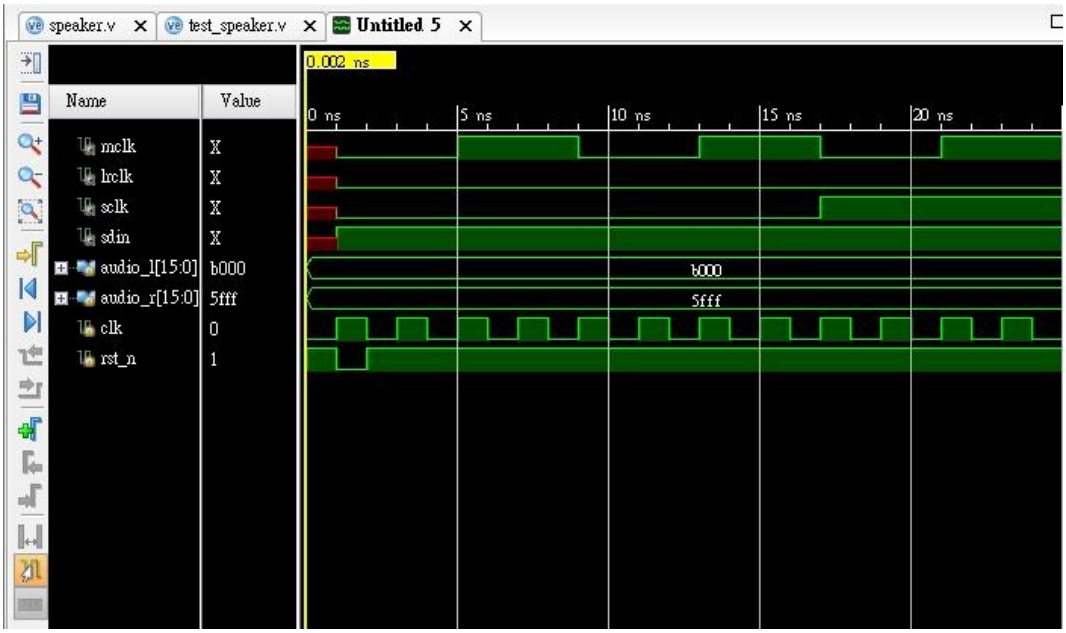
← 上半部的部份是要先做出 speaker 會用到的 clk，分別是 mclk、sclk 還有 lrclk。利用一個 D flip flop 不斷用 clk 向上加 1，因為這三個 clk 彼此差的倍數都是 2 的整數次方倍，也就是說不需要像是之前在找 clk_1Hz 的時候還要另外再接上 XNOR 等來調整，而是直接把該特殊的 clk 設在適當的 cnt 位數即可，mclk、sclk、lrclk 分別是 cnt 的[1]、[3]。

↑ 而下半部的這個巨大 mux 則是透過這張圖寫出來的，利用頻率差，可以讓 sdin 收到 32 個數值，跟以前寫的[1:0]clk_ctl 有異曲同工之妙。

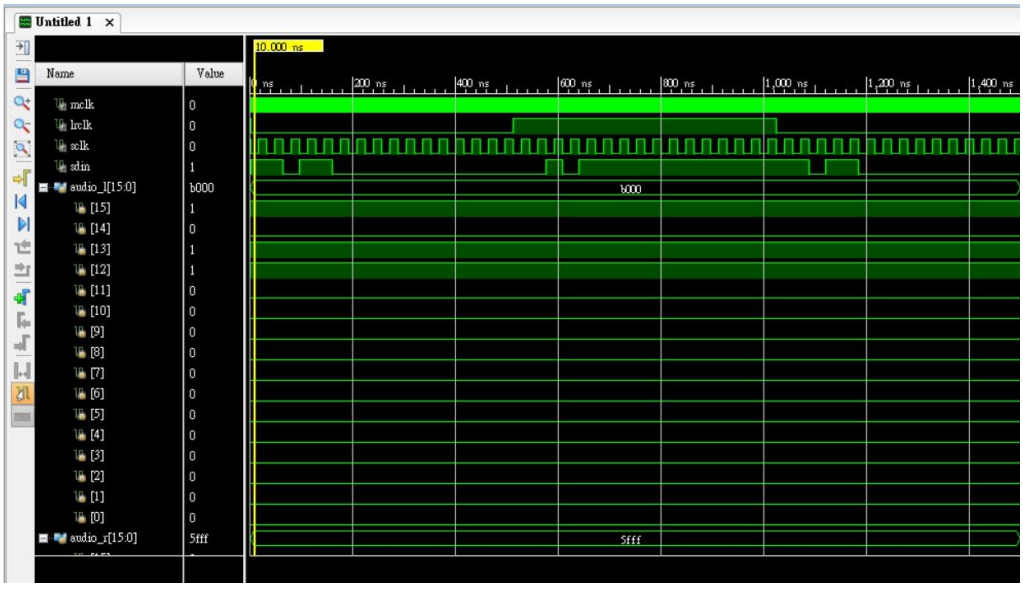


↑ 基本上是把講義上的圖用 verilog code 表現出來。
另外為了簡潔，rst_n 沒有寫在圖上

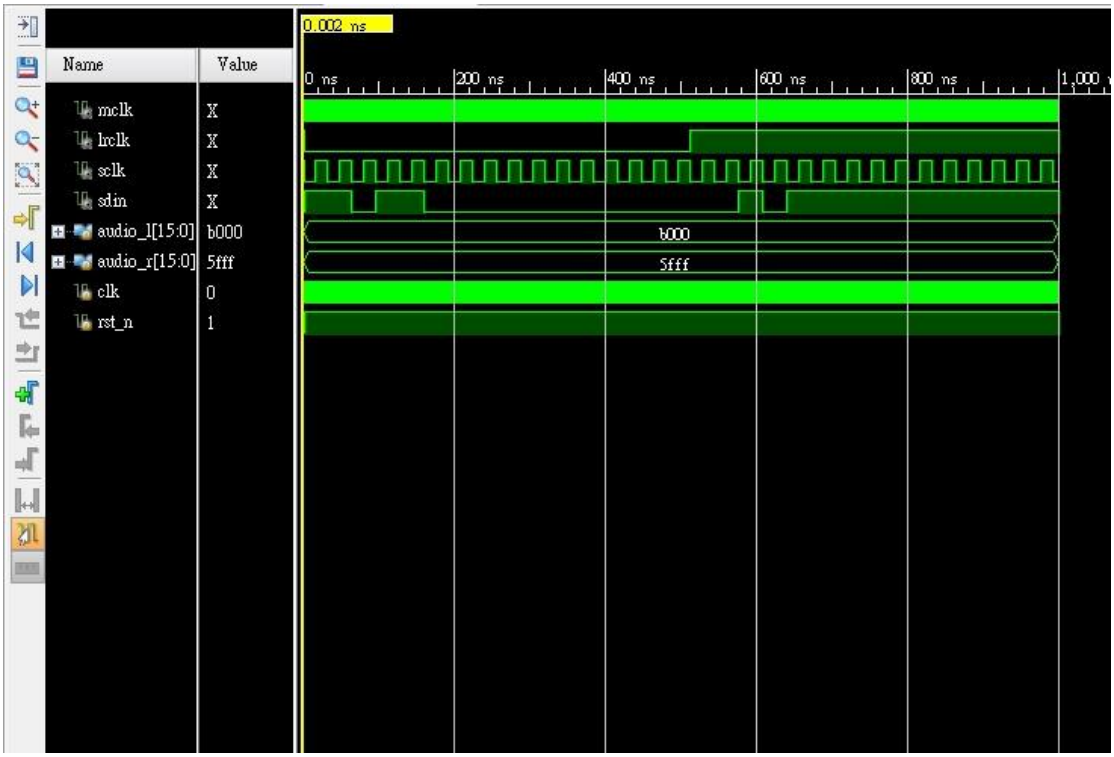
●Stimulation



↑ 由上圖可以看出 mclk 的頻率的確是 clk 的 0.25 倍，也就是 25Mhz;sclk 的頻率是 mclk 的 1/4 倍，也就是 25/4Mhz。另外因為這個題目沒有 audio_l &audio_r 的 input，所以我在 testbench 中就直接訂了 B000 還有 5FFF，是以 2’s complement 的形式表示的，用來控制振幅



↑ 由上圖可以看出整體的 sdin，原本 verilog 預設的長度是 1000ns，只能顯示到第 31 位數，所以我調整到 1500ns，可樣就可以看出整體的 sdin 分布。

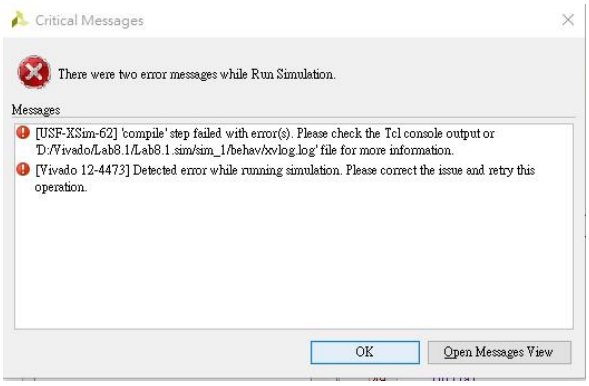


↑ 由這張放到比較小的圖可以看出 lrclk 的頻率是 sclk 的 32 倍，也就是 25Mhz/128。

●Discussion

在打這個實驗的時候遇到以下幾個問題:

1. 在跑 simulation 的時候一直跑出這個訊息，但是遲遲找不到問題是出在哪裡。



<sol:>我原本把 left-right clock 的變數設成 l_rclk，結果我最後改成 lrclk 就可以了，但是有點納悶的是不知道為什麼以前可以用“_“現在則不行……

8-2. Speaker control

●Design Specification

Input:

clk,(接上板子的原本的震盪頻率 W5)

rst_n, (控制整個功能的開關)

pb_up(可以讓音量提高的按鈕)

pb_down(可以讓音量下降的按鈕)

pb_Do(可以發出 Do 聲音的按鈕)

pb_Re(可以發出 Re 聲音的按鈕)

pb_Mi(可以發出 Mi 聲音的按鈕)

Output:

Mclk,(master clock，是最快的 clk 讓其他 clk 校準, 25MHz)

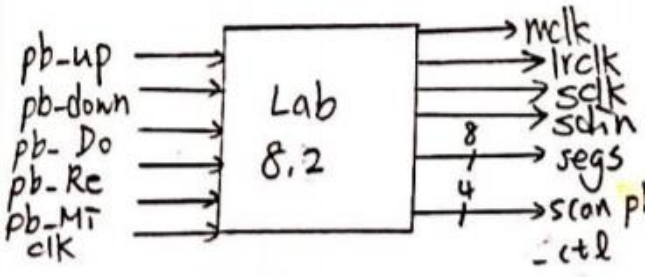
lrclk,(left-right clock，是要給左聲道和右聲道訊息的, 25MHz/128)

Sck,(serial clk,讓一個個訊號進去的頻率，比 lrclk 的頻率快 32 倍, 25Mhz/4)

Sdin(1 bit serial audio data output)

[3:0]Scan_ctl,(負責控制七段顯示器的顯示頻率)

[7:0]segs(負責七段顯示器的顯示)



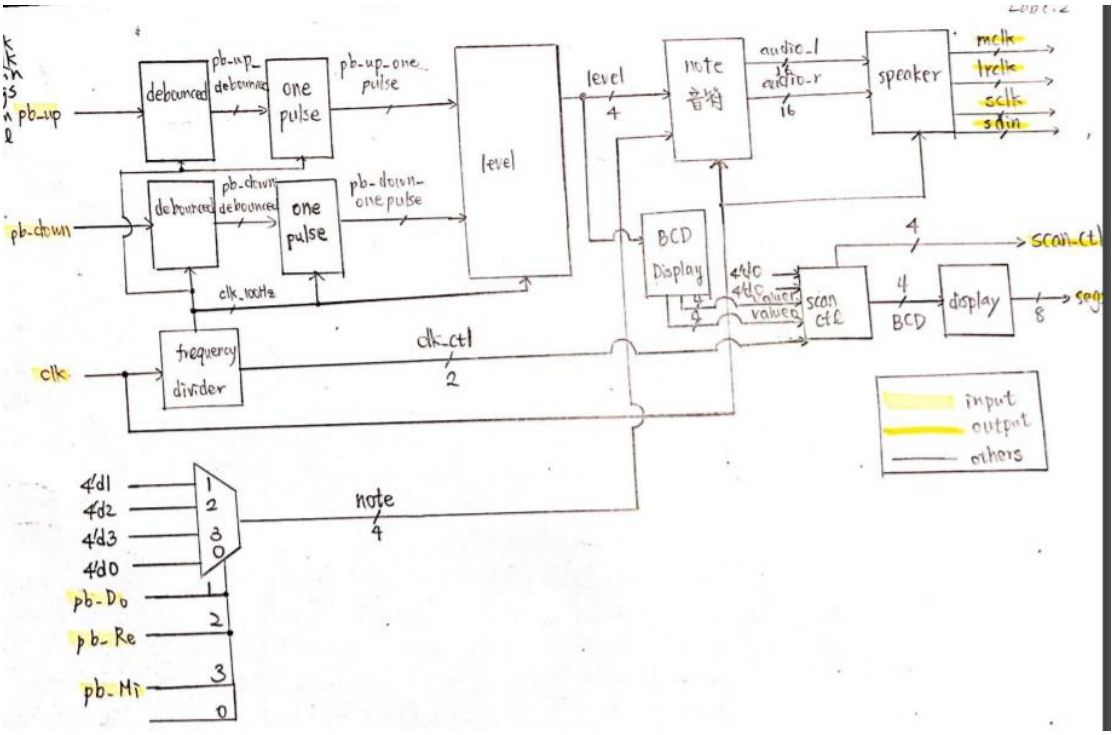
(input 少了 rst_n)

●Design Implementation

1.Outline

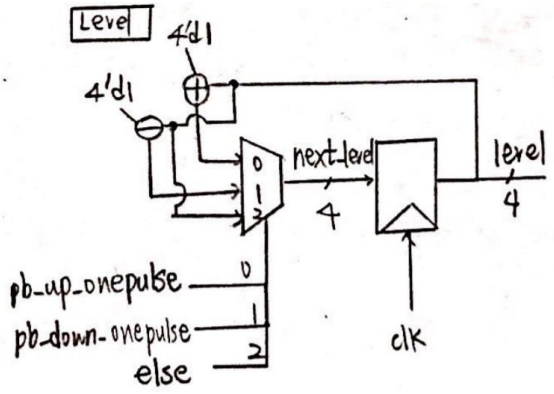
這題會用到的新 module 比較多，主要是要利 mux 來判斷輸入，進而決定不同數值，再傳入 note(音符)來 decode 變成相關頻率的 limit，當作 up counter toggle 的條件，即可發出題目音效 Do、Re、Mi。還需要利用 8.1 的 speaker 來創造 mclk、lrclk、sclk 還有 sdin 給 speaker 使用。另外還要重新做一個 level 可以調整音量，因為以前的 module 都是 up counter 或是 down counter，沒有同時有可以 up and down 的 counter，所以需要重新寫一個 level。整體來說算比較複雜，觀念較新的題目。

2.Logic Diagram



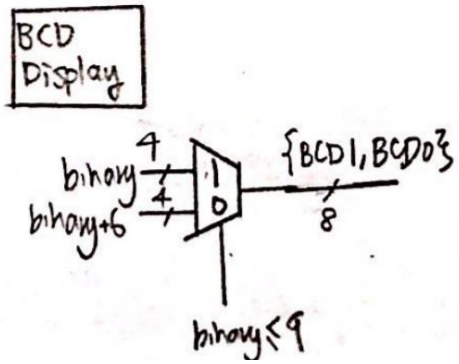
↑ 上圖是整個系統的邏輯圖，新的 module，像是 note、level、BCD display，會在下面詳細解釋。(speaker 已經再 8.1 解釋過了)

▲Level



這個 module 負責的功能是上下加減調整的 counter，在 D flip flop 外再接一個 mux 來控制要加減或是維持原本的數值。

▲BCD Display



之前沒有用這個 module 因為之前大多用 BCD 來處理需要顯示再七段顯示器的數據。但是這次 level 需要以 binary 的形式 input 進去 note module，所以要另外寫一個來轉換 LEVEL 成為 BCD 的形式。即是超過十的話，需要再加上 6 來調整結果。

▲Note

這個 module 負責產生不同頻率音量的聲音，不同 level 的時候，振幅會不圖，而 b_clk 為 1 的時候，代表波峰，為 0 的時候，代表波谷。至於不同頻率，則是用類似 frequency divider 的形式，屬到特定數字後，將 b_clk toggle。 而不同的頻率要屬到不同的數字。

↓以下是 I/O 接腳

I/O	segs [7]	segs [6]	segs [5]	segs [4]	segs [3]	segs [2]	segs [1]	segs [0]	Clk	Rst_n
Pin	V14	U14	U15	W18	V19	U19	E19	U16	W5	V17

I/O	ssd_ctl[3]	ssd_ctl[2]	ssd_ctl[1]	ssd_ctl[0]
Pin	W4	V4	U4	U2

I/O	Pb_up	pb_down	pb_Do	pb_Re	pb_Mi
Pin	T18	U17	W19	U18	T17

I/O	Mclk	Lrclk	Sclk	Sdin
Pin	A14	A16	B15	B16

●Discussion

這整個實驗運用到非常多新的觀念，還需要更多特定的頻率來讓SPEAKER運作，另外不同種的聲音也需要不同的limit來跑up counter來做成精準的發聲。這次的實驗我也是到處問同學還有教授才看懂講義上的code代表的意義，但是在了解之後，其實在實際操作上也沒有遇到什麼問題，希望之後的vga還有keyboard也能這樣順利…..

我一開始弄錯sdin的頻率，我把他跟 crytal clk同步，但實際上要跟慢16倍的sclk同步。

●Conclusion

這次的lab讓我學會如何處理parallel to serial的data，還有如何使用speaker。這讓我不禁想到之後的final project可以加入音樂的元素，但是如果光是三個頻率的聲音都這麼麻煩了，不知道如果想要放入一整首音樂，更何況音樂還有不同節奏或是合聲，會有多麼麻煩了……真是讓人不敢恭維。