Lab07 (109061256 陳立萍)

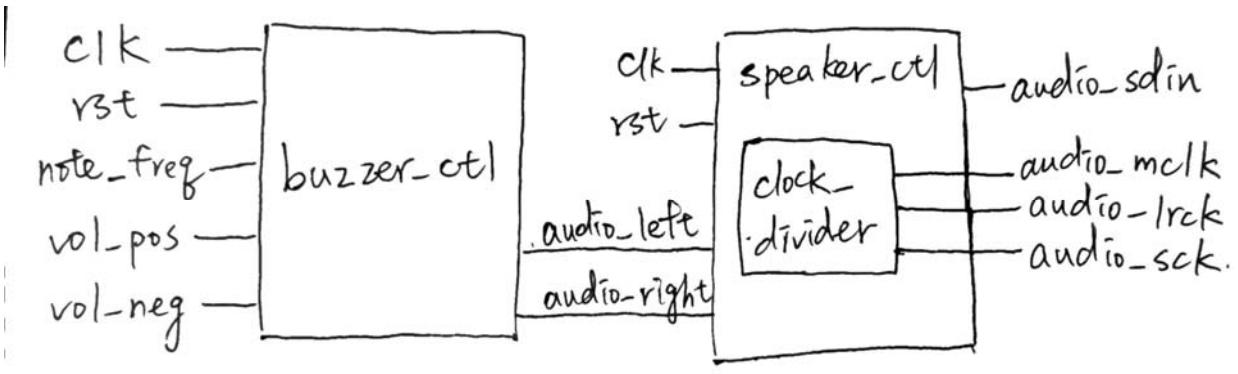
Lab7_1: Please design an audio-data parallel-to-serial module to generate the speaker control signal with 100MHz system clock, 25 MHz master clock, (25/128) MHz Left-Right clock(Fs), and 6.25 MHz (32Fs) sampling clock.:

1.1 Design a general frequency divider to generate the required frequencies for speaker clock.
1.2 Design a stereo signal parallel-to-serial processor to generate the speaker control signals. Please use Verilog simulation waveform to verify your control signal.

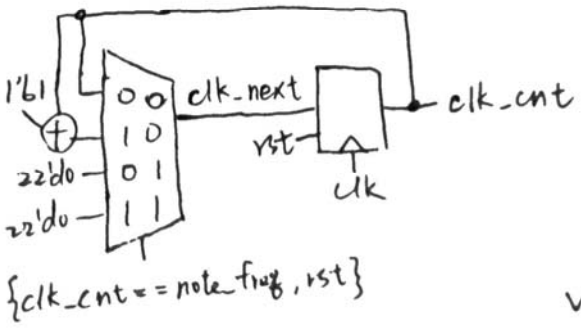
Design Specification

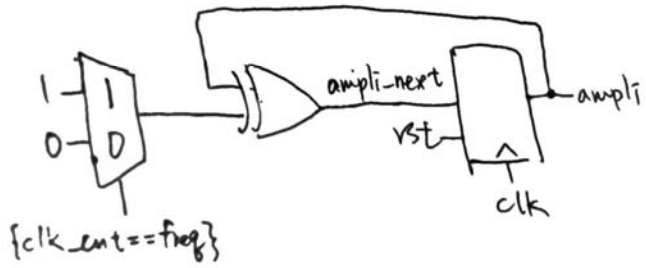
input : clk(100Mhz), rst(reset)
output : audio_mclk(25MHz), audio_lrck(25/128MHz),
audio_sck(6.25MHz), audio_sdin(serial output),
[15:0]audio_left(左聲道), [15:0]audio_right(右聲道)

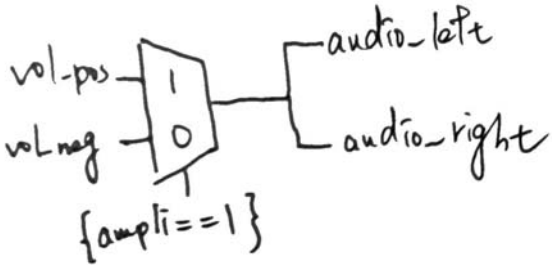
logic diagram



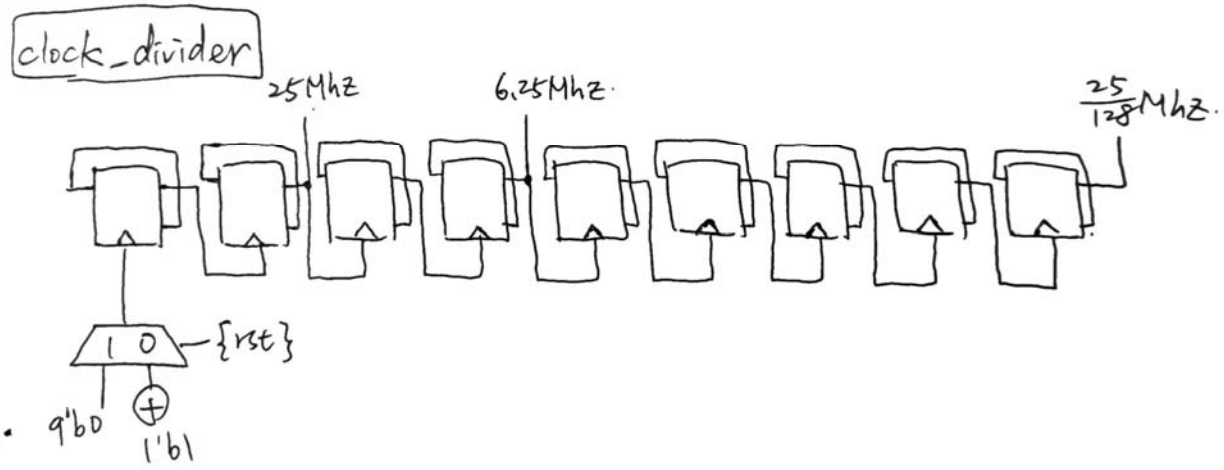
buzzer_ctl

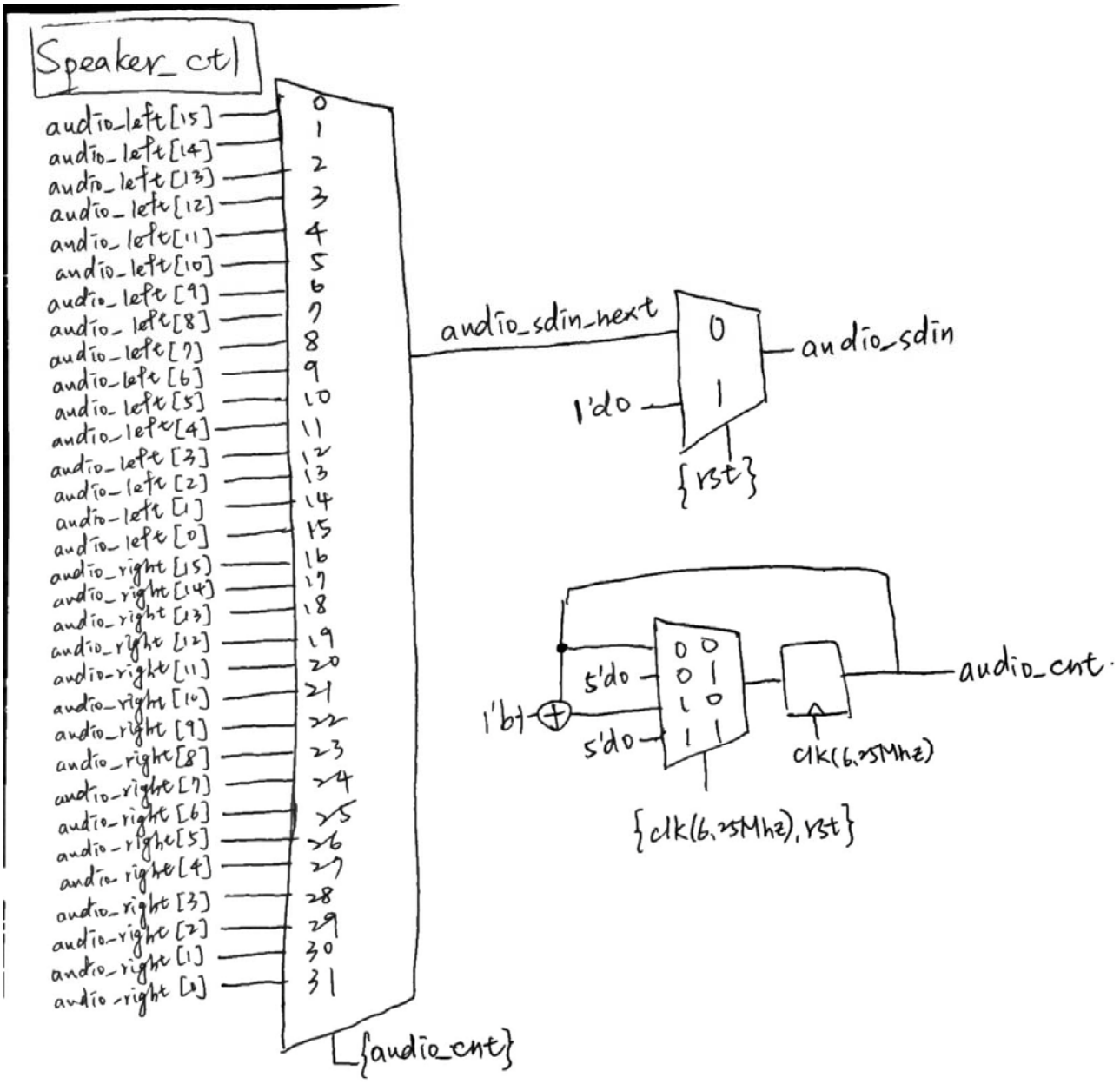






clock_divider





Design Implementation

設計方法：

＊buzzer_ctl.v :

用來製造特定的頻率，此頻率的除數會由其他地方輸入進來這裡，在此先隨意定義輸入進來的除數 note_freq=151515（mid Mi 音之除數），而除頻原理與之前的除頻做法相同。不同的是除完頻後的頻率是用來控制左右耳輸出的振幅最高點與最低點，當此頻率(ampli)為 1 的時候輸出最大值（似波峰），頻率(ampli)為 0 的時候為輸出最小值（似波谷），在這裡給的振幅最大值定為 16'h7FFFF，最小值定為 16'h8000，兩者互為 2's complement。

＊clock_divider.v :

用來除出三種頻率，分別為 25MHz(mclk), 25/128MHz(lrck), 6.25MHz(sck/sampling clock)。因為這三種 clock 剛好都是 100MHz/2^k 的形式，所以可以用 k 個 flipflop 組成的 counter 來做處理，

拉出第 k 個 filpflop 的輸出即為我們要的頻率。
(25MHz：100/2^2 ,　25/128MHz：100/2^9,　6.25MHz：100/2^4)

＊speaker_ctl.v:

用來將 audio_left 和 audio_right 轉成 1bit 輸出（做 parallel to serial 的處理），利用 clock_divider.v 做出在這裡需要的 6.25MHz（audio_lrck 的 f*32 = sck 的 f，因為 audio_left + audio_right 共有 32 個 bit）。 在這裡設定了一個 5bits 的 counter，這個 counter 的 clock 即為 6.25MHz，每個 clock 來的時候會將 counter+1，counter 暫存的每個不同的數字（0~31）會對應到 audio_left 或 audio_right 的某一個 bit，再將此 bit 給 audio_sdin 作輸出。

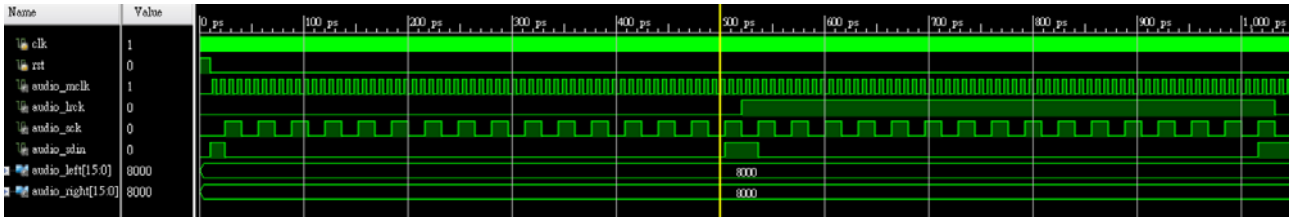
＊lab7_1_top.v:

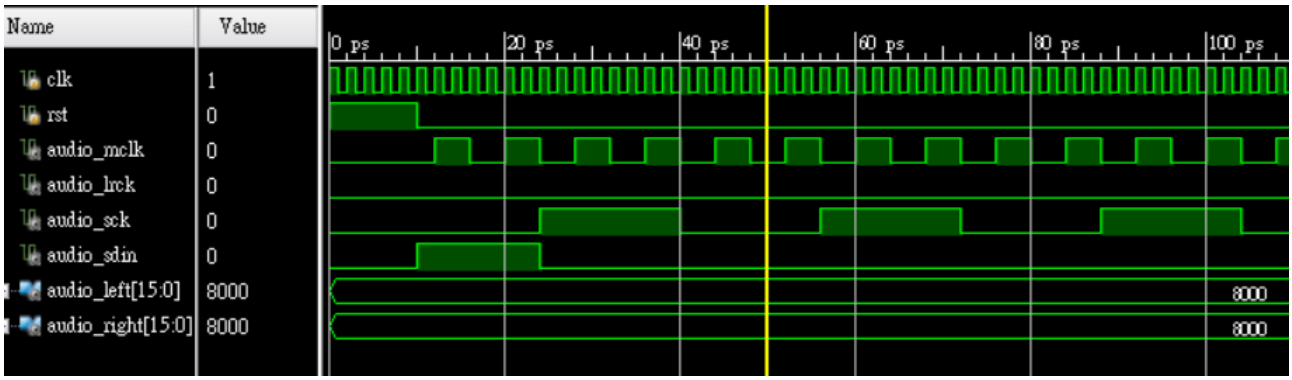
連結 buzzer_ctl.v 和 speaker_ctl.v 並指定音量最大值 7FFF 與最小值 8000，以及 note_freq 為 151515。

＊tesetbench.v:

在此先讓 clk 初始值為 1，rst 初始值為 1，接著讓 rst 變為 0 後保持不變，並讓 clk 不斷 0101 交替。

wave diagram





波形圖分析：

從波形圖可以看出，rst 從 1 變回 0 後，各個 output 開始變化。其中 clk（100MHz）的頻率是 audio_mclk（25/4MHz）的 4 倍，而 audio_mclk 的頻率是 audio_sck（6.25MHz）的 4 倍/audio_lrck（25/128MHz）的 128 倍。audio_sck 的頻率是 audio_lrck 的 32 倍，剛好符合 parallel to serial 要有的頻率關係。audio_sdin 為 1 處則表示 speake_ctl.v 的 counter 數到 1，輸出 audio_left[14]。而從 audio_lrck 可以看到是左聲道先出來，因為 audio_lrck 從 0 開始，接著 audio_lrck 變成 1，換右聲道出來。

Lab7_2: Speaker control:

2.1 Please produce the buzzer sounds of Do, Re, and Mi by pressing buttons (Left, Center, Right) respectively. When you press down the button, the speaker produces corresponding frequency sound. When you release the switch, the speaker stops the sound.
2.2 Please control the volume of the sound by pressing button (Up) as increase and (Down) and decrease the volume. Please also quantize the audio dynamic range as 16 levels and show the current sound level in the 7-segment display.

Design Specification

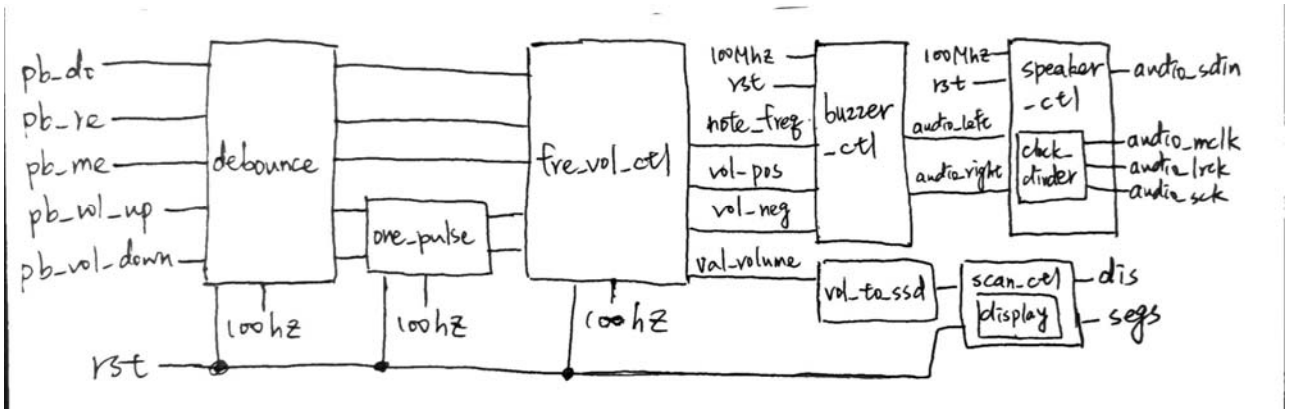
input : clk_100mhz (100Mhz), rst,(reset), pb_do(Do), pb_re (Re), pb_mi (Mi),

pb_vol_up(音量加 1), pb_vol_down (音量減 1)

output :audio_mclk(25MHz),audio_lrck(25/128MHz),
audio_sck(6.25MHz),audio_sdin(serial output),

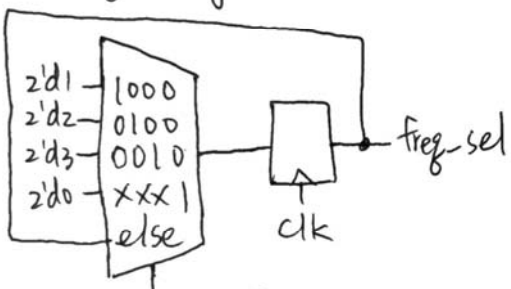
[7:0]segs(七段段顯示器圖形), [3:0]dis(四個七段顯示器),

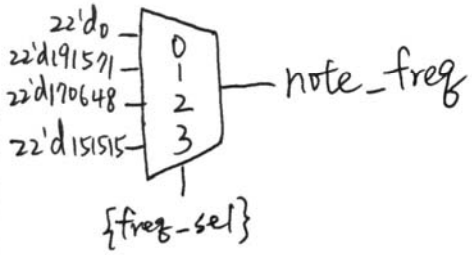
logic diagram



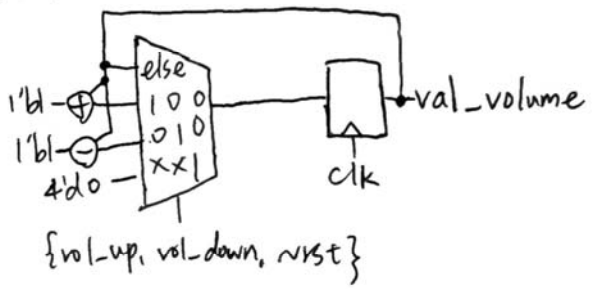
fre_vol_ctl

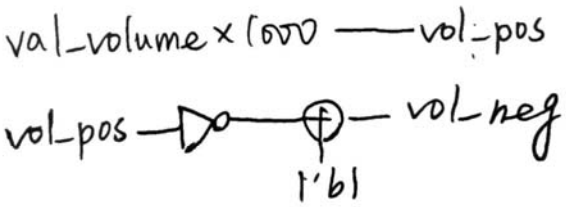
※ frequency



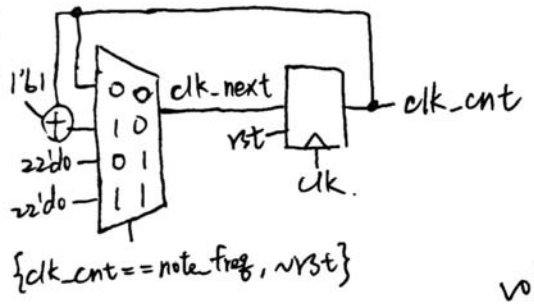


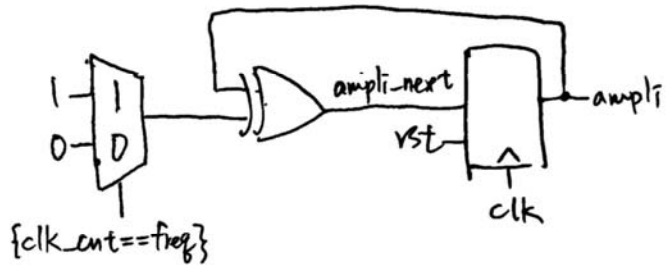
※ volume

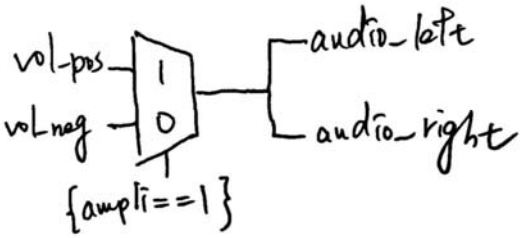


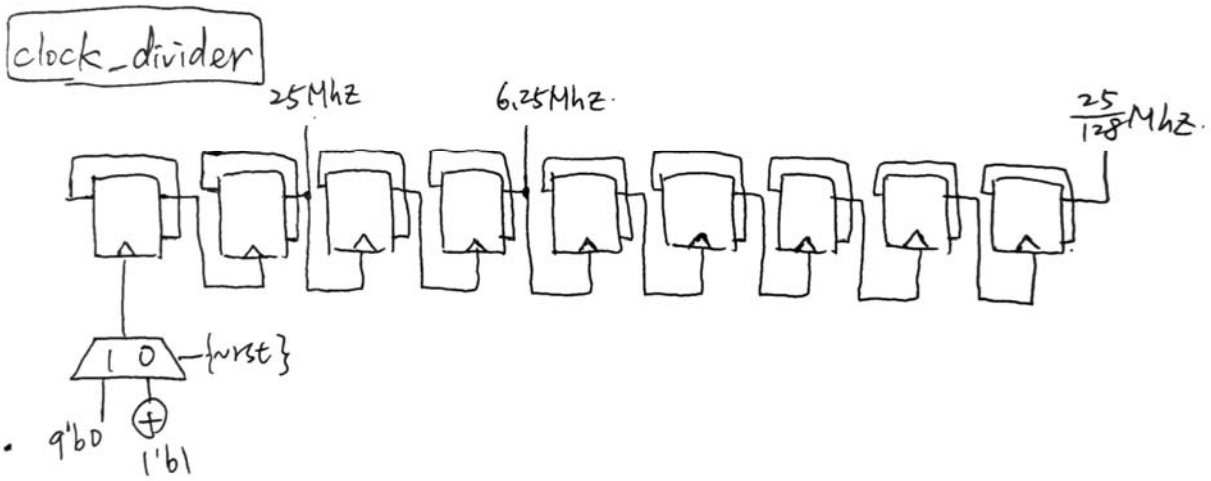


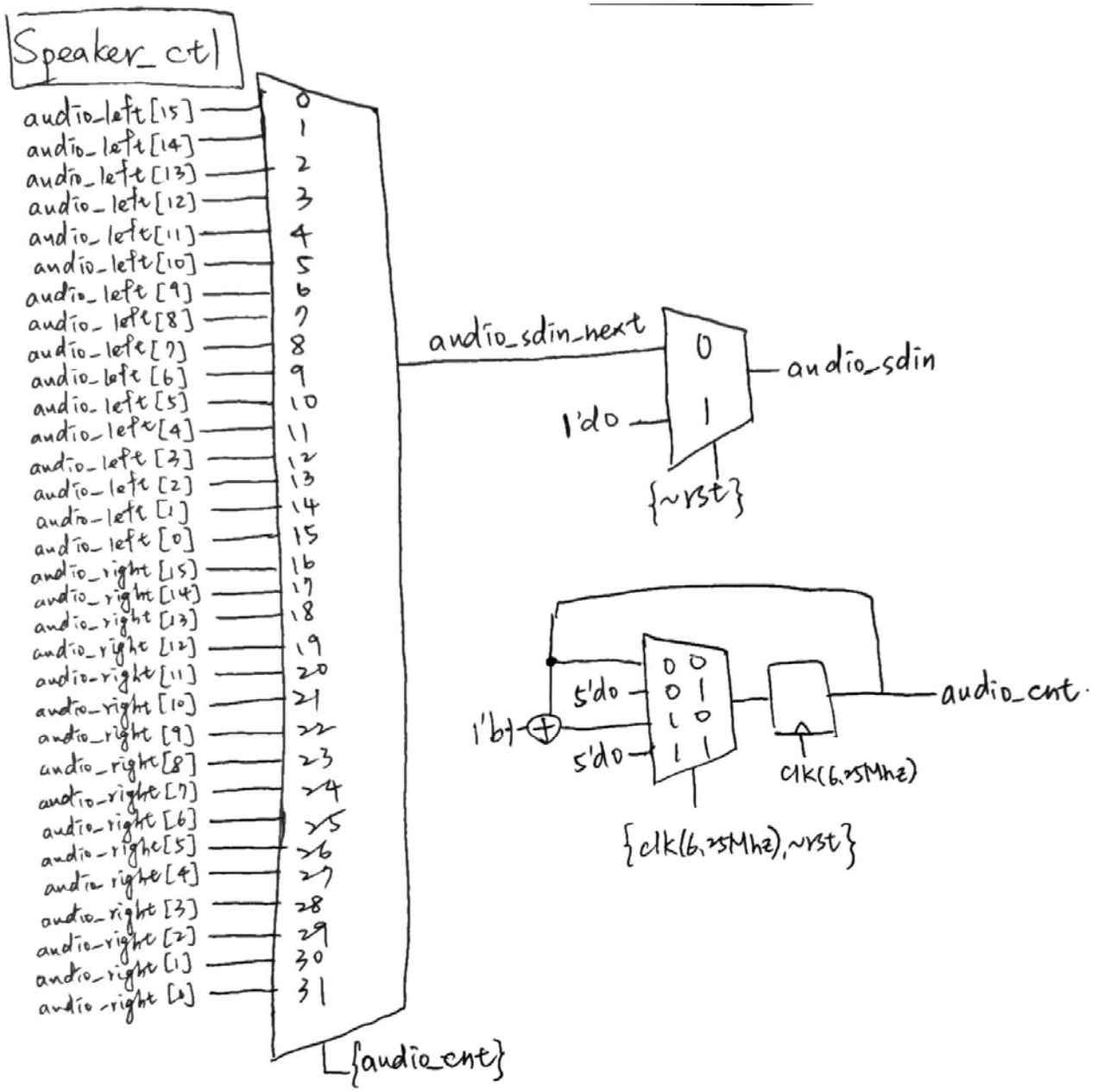
buzzer_ctl

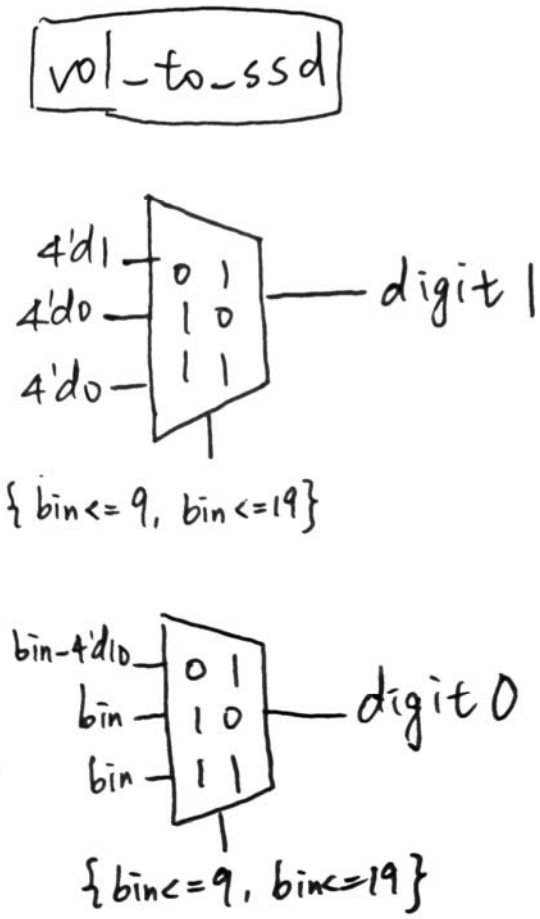







clock_divider
25Mhz
6.25Mhz
25/128Mhz
1 0
~rst
9'b0
1'b1


Speaker_ctl
audio_left[15]
audio_left[14]
audio_left[13]
audio_left[12]
audio_left[11]
audio_left[10]
audio_left[9]
audio_left[8]
audio_left[7]
audio_left[6]
audio_left[5]
audio_left[4]
audio_left[3]
audio_left[2]
audio_left[1]
audio_left[0]
audio_right[15]
audio_right[14]
audio_right[13]
audio_right[12]
audio_right[11]
audio_right[10]
audio_right[9]
audio_right[8]
audio_right[7]
audio_right[6]
audio_right[5]
audio_right[4]
audio_right[3]
audio_right[2]
audio_right[1]
audio_right[0]
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
{audio_cnt}
audio_sdin_next
0
1
1'd0
audio_sdin
{~rst}
0 0
0 1
1 0
1 1
5'd0
5'd0
1'b1
audio_cnt
clk(6.25Mhz)
{clk(6.25Mhz),~rst}



I/O pin assignment:

pb_do	pb_re	pb_mi	pb_vol_up	pb_vol_down	rst
W19	U18	T17	T18	U17	R2

audio_mclk	audio_lrck	audio_sck	audio_sdin	clk_100mhz
A14	A16	B15	B16	W5

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

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

Design Implementation

設計方法：

這個題目裡共用了 11 個.v 檔。

其中 fre_div_100.v 用來製造 100hz。debounce.v：消除按下按鈕後的 0101 震盪。

one_pulse.v：用來製造一個 clock 長度的 pulse，以確保輸入的訊號只經過一個 clock，也就是我們製造的 one_pulse 訊號。scan_ctl.v：製造視覺暫留。display.v：七段顯示器輸出。buzzer_ctl.v：用來製造特定的頻率（Do/Re/Mi）。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：用來除出三種頻率，分別為 25MHz(mclk),25/128MHz(lrck),6.25MHz(sck/sampling clock)。

上述.v 檔之原理已於之前和 lab7_1 介紹過，在此不再贅述。下列為此次新的 module 介紹：

＊fre_vol_ctl.v:

這個.v 檔會處理要輸出給 buzzer_ctl.v 的 frequency（Do/Re/Mi）以及要輸出給 vol_to_ssd.v 的 val_volume。頻率的部分，會接 3 個訊號（Do/Re/Mi）進來，因為題目要求是一直按著都會有聲音，所以要接 3 個按鈕 debounce 後的訊號進來，接著看哪一個訊號是 1，則將此訊號送到多工器作選擇，選擇出對應的頻率除數，（頻率除數：Do：191571/Re：17648/Mi：151515）送到 buzzer_ctl.v 去產生特定頻率的聲音。音量的部分，先設了一個 4bits 的 counter（val_volume)，這個 counter(val_volume)的範圍是（00~15，剛好 16 個 level），每次按 pb_vol_up 就會把 counter 加一，同理，按一下 pb_vol_up 就會減一，在此不考慮溢位處理，所以可以方便使用者在 15 的 level 按 pb_vol_up 跳回 00。接著將此 val_volume 乘以 1000 倍放大後給振幅最高值 vol_pos（16bits），再將（~vol_pos)+1 給振幅最低值 vol_neg（16bits）。vol_pos 和 vol_neg 之後會送到 buzzer_ctl 依 ampli 的 0/1 來決定 audio_left 和 audi_right 為 vol_pos 或 vol_neg。

＊vol_to_ssd.v:

這個 lab 是用來將 fre_vol_ctl.v 產生的 4 個 bit 的 val_volume 從二進位轉成十進位表示，且將十進位表示的數字拆成十位數與個位數送給 scan_ctl 去做七段顯示器呈現。因為這裡只分 16 個級別（00~15），所以語法就是直接用 if,else,下去判斷再輸出對應的數字。

＊lab7_2_top.v:

連結上述 11 個 module，形成完整的功能。

Discussion

Lab7_1

這題一開始寫的很順利，但是 simulation 出來後的波形圖都不對，值不是是 x 就是 z，原先一直以為是主要 module 寫錯，後來才發現，原來是 testbench 的地方根本沒有 initial 所以 clock 根本不會 0101 變換。而這個 lab 方便的地方就是要產生的頻率剛好都是 100/2^k 的形式，所以不用像做產生 1hz 那樣麻煩的寫，只要一個 9bits counter 再接第 2，4，9 位元出來就可以做了。結果如波形圖可見為正確。

Lab7_2

這題基本上和第一題大致相同，不同的地方多了輸入的音調頻率會有 3 種，還有音量的調控，頻率的地方很簡單，用多工器選擇就可以了。我覺得難的地方是音量的調配，因為原先並不知道 FPGA 板的最大輸出音量，所以我一開始不是把 0~15 乘以 1000 而是乘以 2184 倍，而當時的預設音量最大值 2184*15=32760 並沒有超過 2^15，所以應當沒有溢位的問題，可是在我輸出聽聲音時發現 0~10 會遞增，10 最大聲，11~15 又不斷遞減。之後我又試了好幾個不同的

加大倍率，乘以 4369 時，在 3,7,11 時有最大聲，而乘以 1092 時可以剛好讓 15 最大聲，因此我當時認為應該是板子本身會放大音訊，導致 2184 和 4369 的兩個例子中的數值超過我預設的 16bits 的 vol_pos 和 vol_neg 因而溢位，變成數值很小，導致聲音很小。之後問了助教，助教的說法是板子會預設一個音量極限，因此超過會溢位。

Conclusion

這個 lab 其實比上一個 lab 簡短許多，可是因為是新的題材-聲音，因此我還是花了很多的時間下去了解，尤其了解了很多在聲音的部分不管是人耳的聲波接收，或是產生的聲波的原理以及相關知識。但是在音量的那個部分我花了 2 天在研究，因為我原本真的不覺得會溢位，我不段用筆計算，確定沒有超出範圍，但是怎麼試就是找不出為什麼音量不會遞增的原因，但其實就是對於設備特性的不熟悉，不知道板子對於聲音訊號的處理。除了音量外，我覺得其他的部分都沒有到很難理解，下一個 lab 是做鍵盤，也是一個新穎的內容，希望可以記取這次對於硬體不了解的教訓，下一個 lab 可以做快一點。

References

教授授課頭影片：程式寫法，設計觀念，聲音與音訊相關內容。