**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**

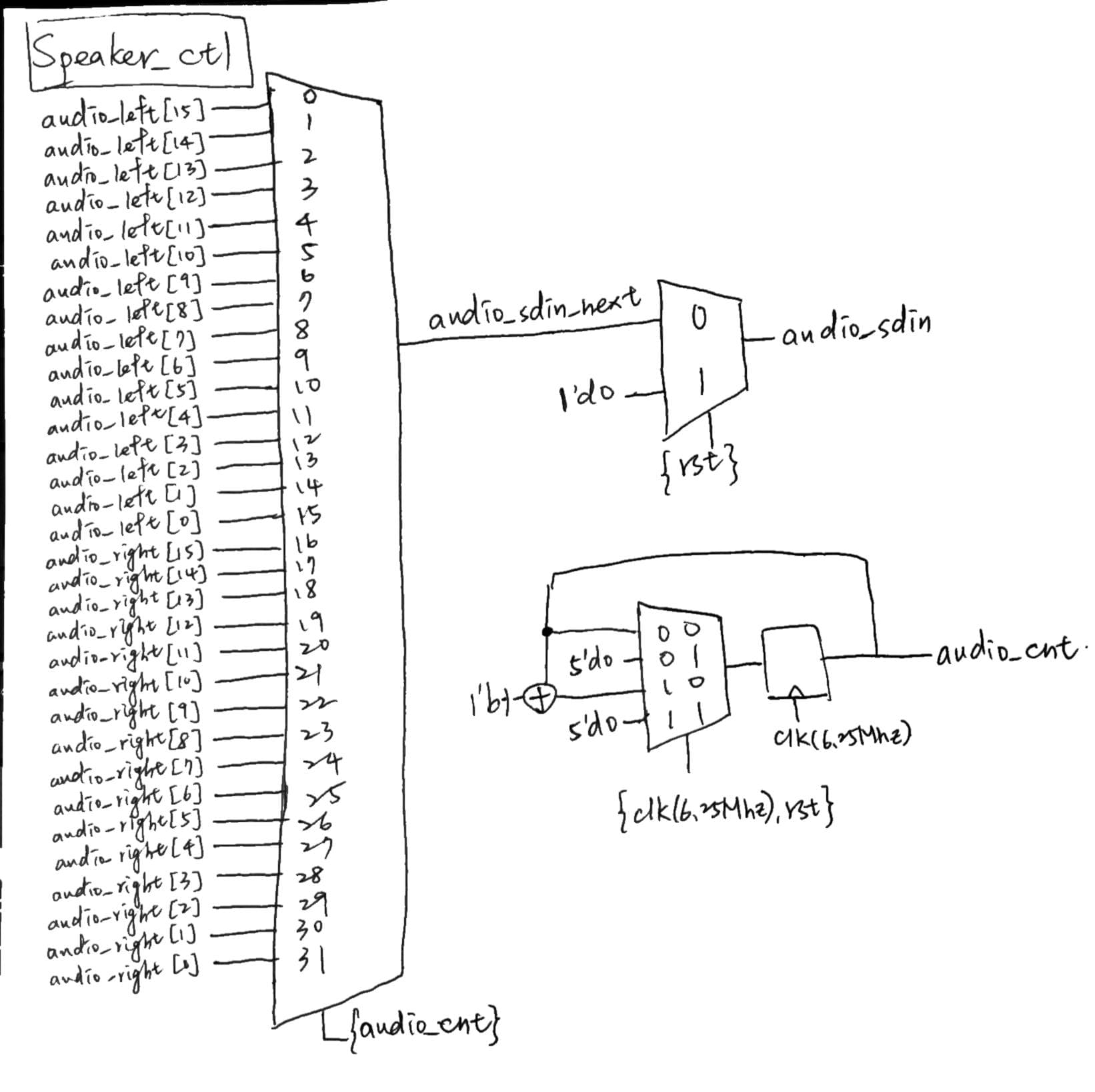
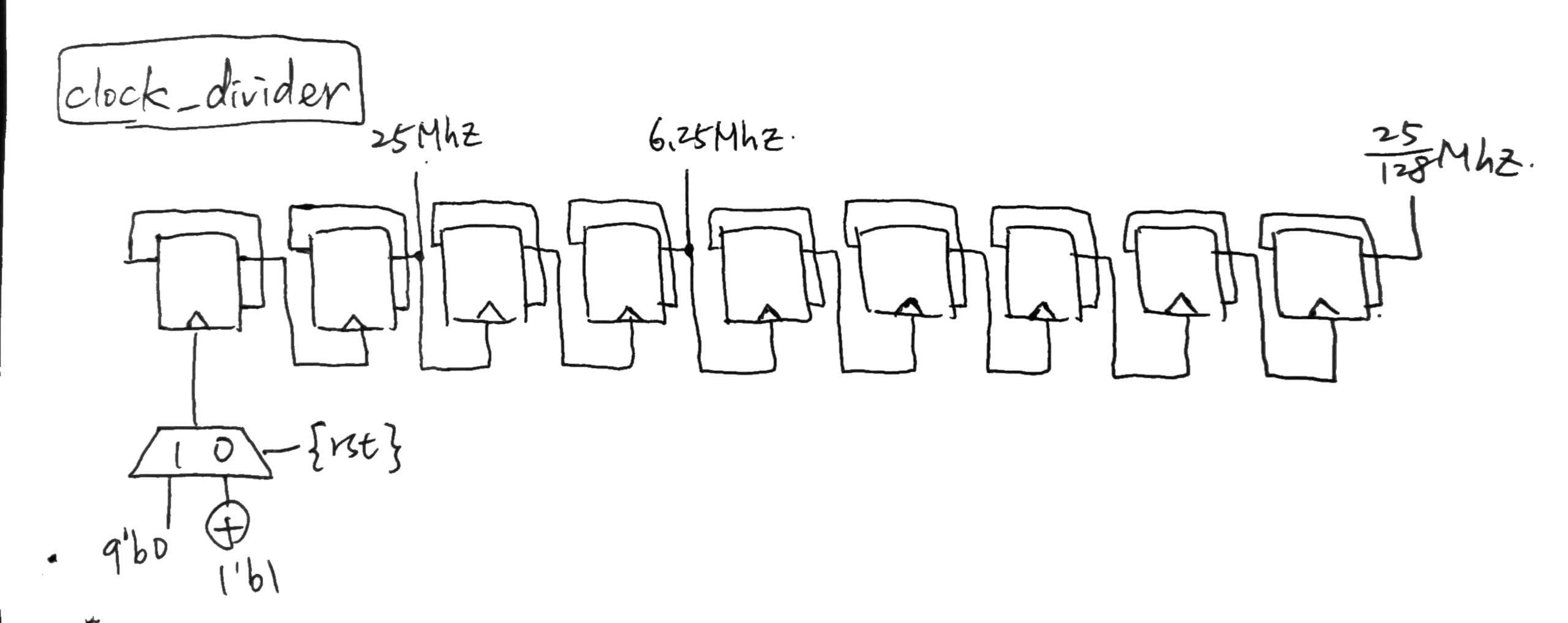
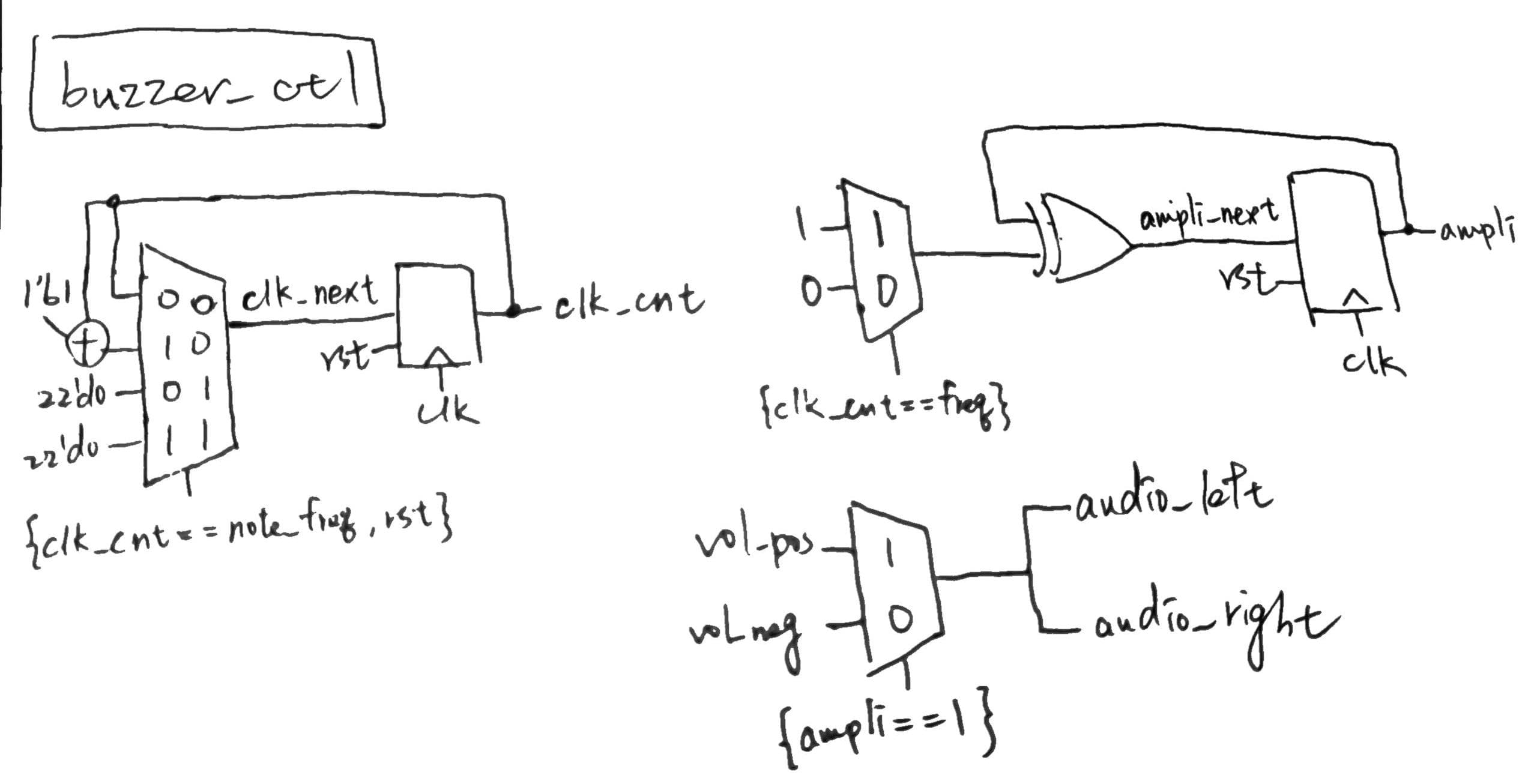
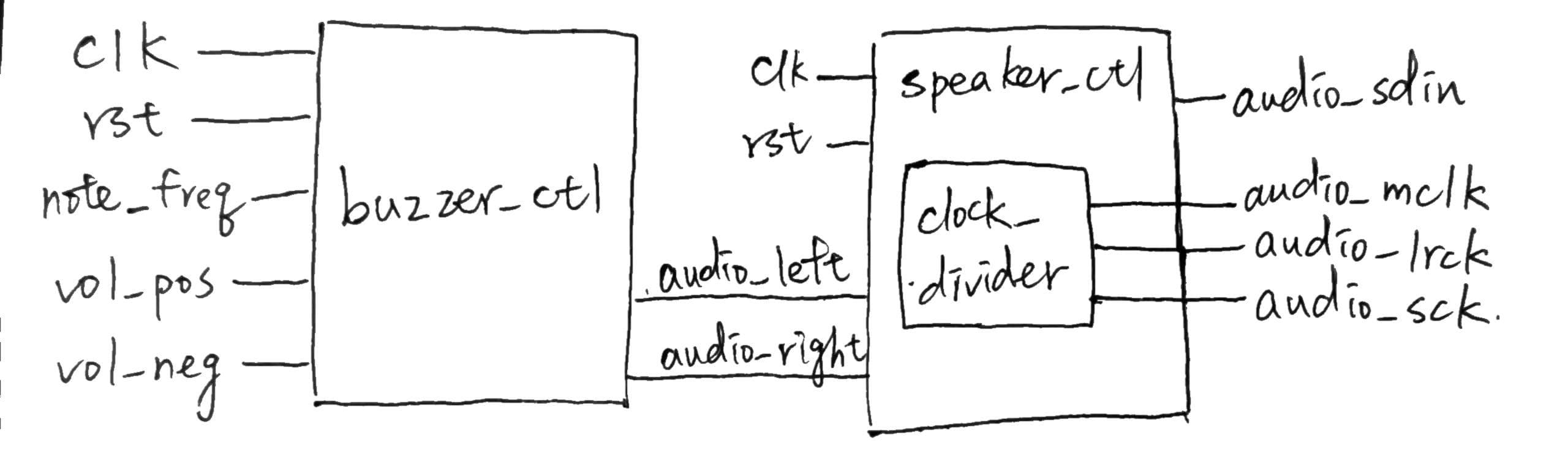
input : clk(100Mhz), rst(reset)

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**



**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。

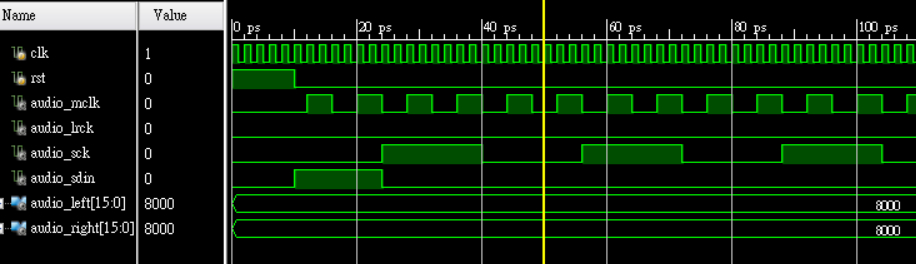
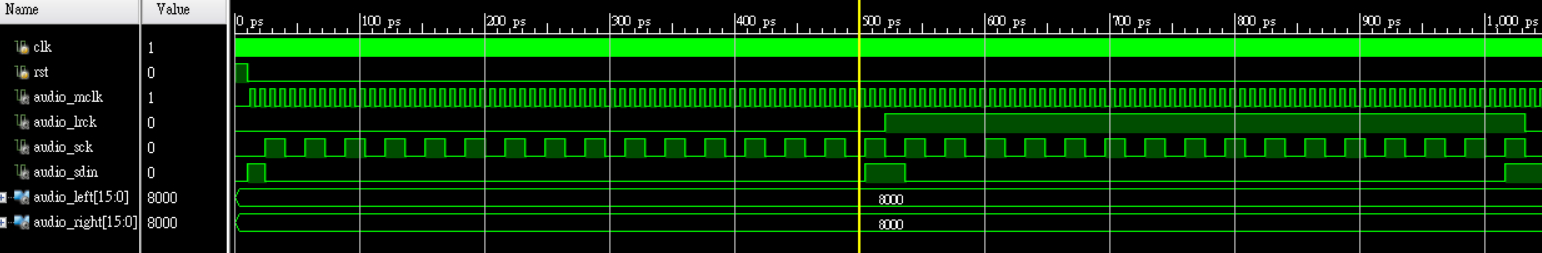
＊lab7\_1\_top.v:

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),

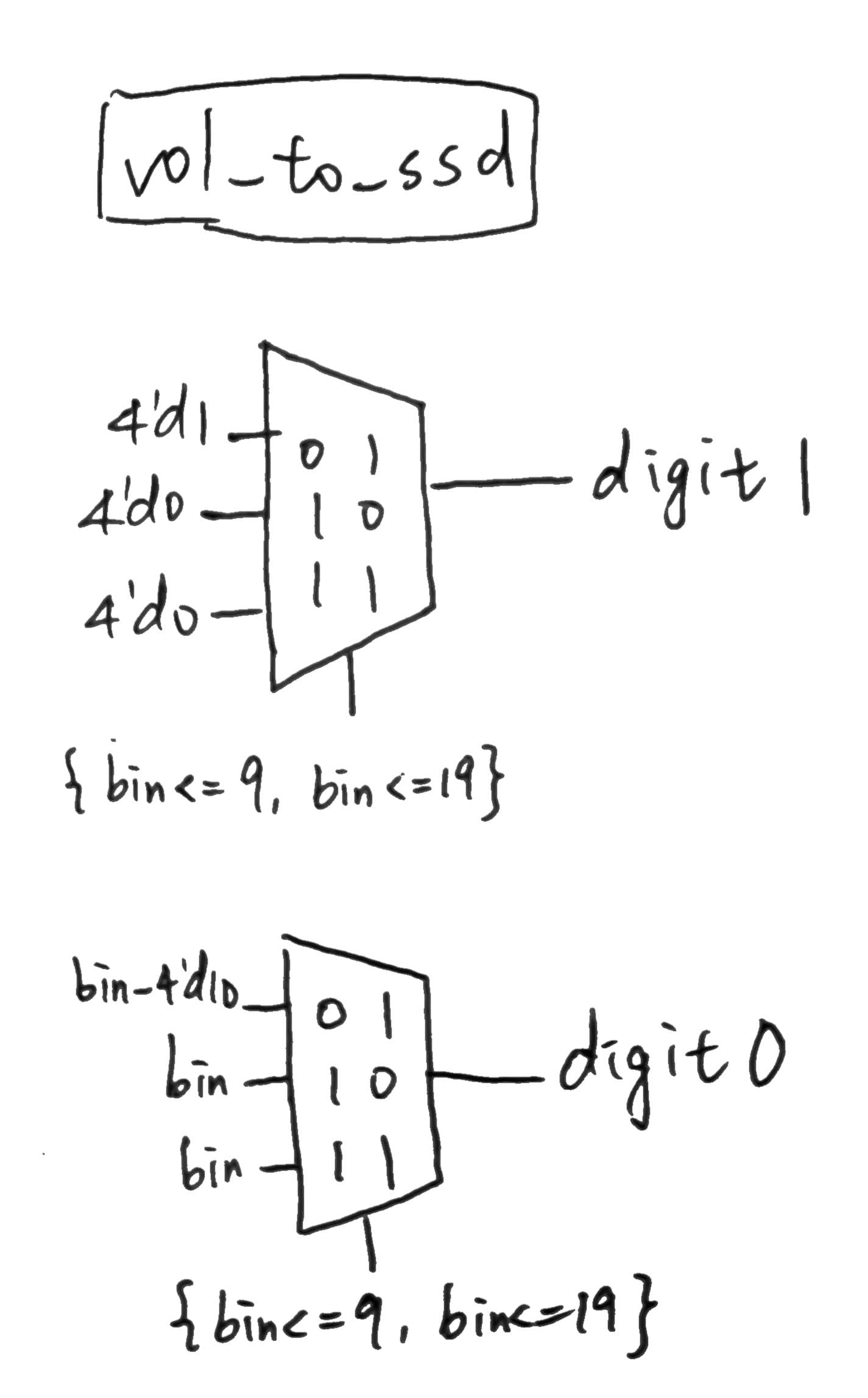
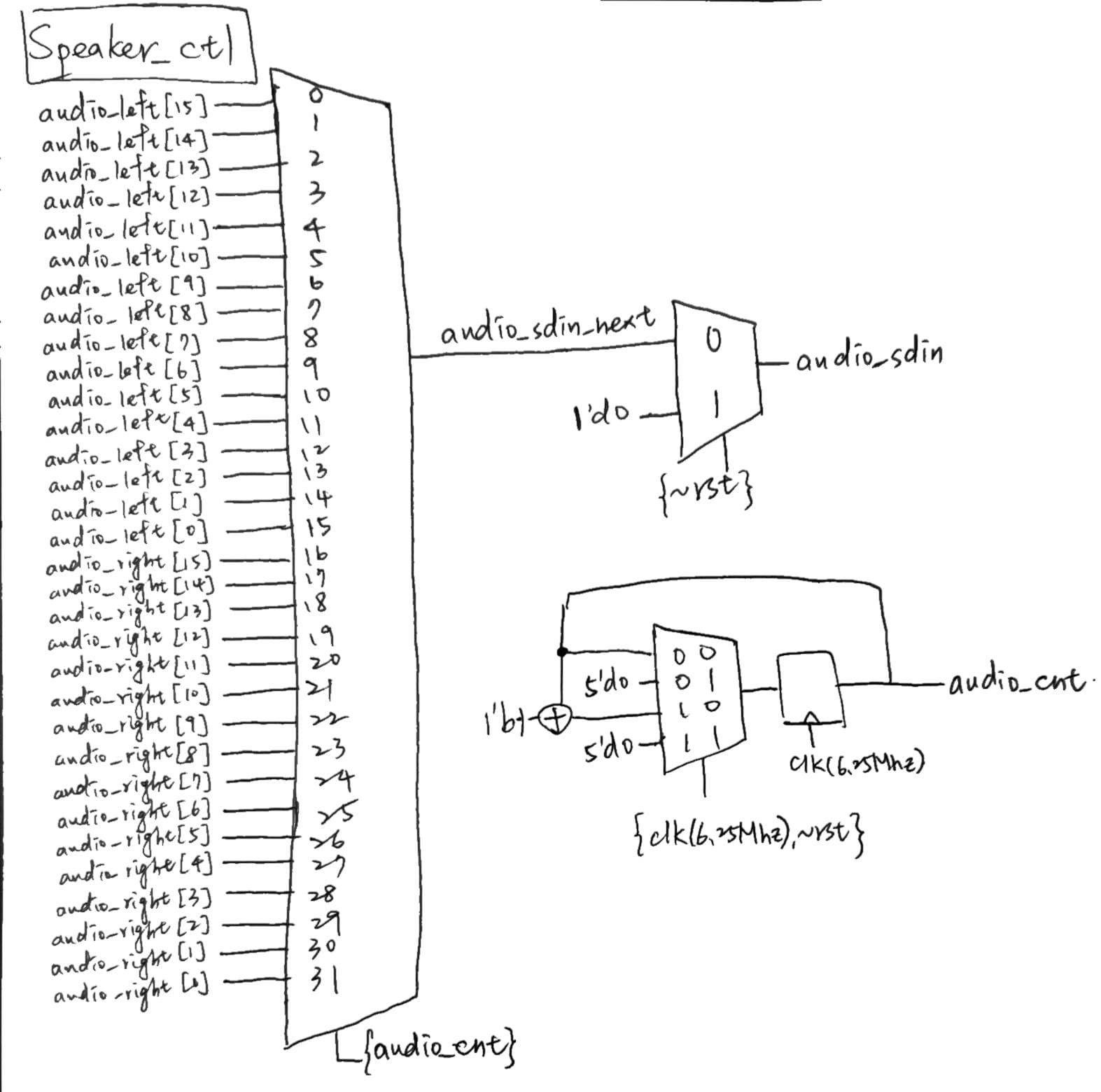
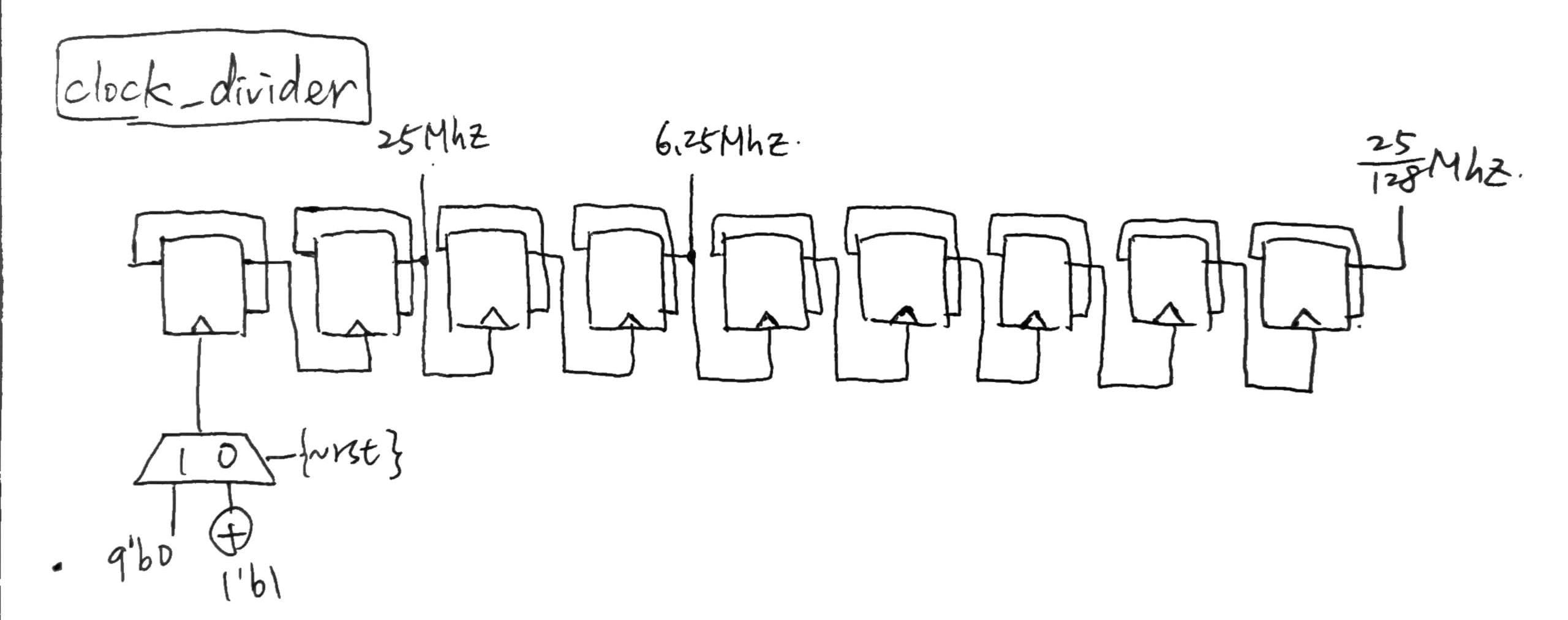
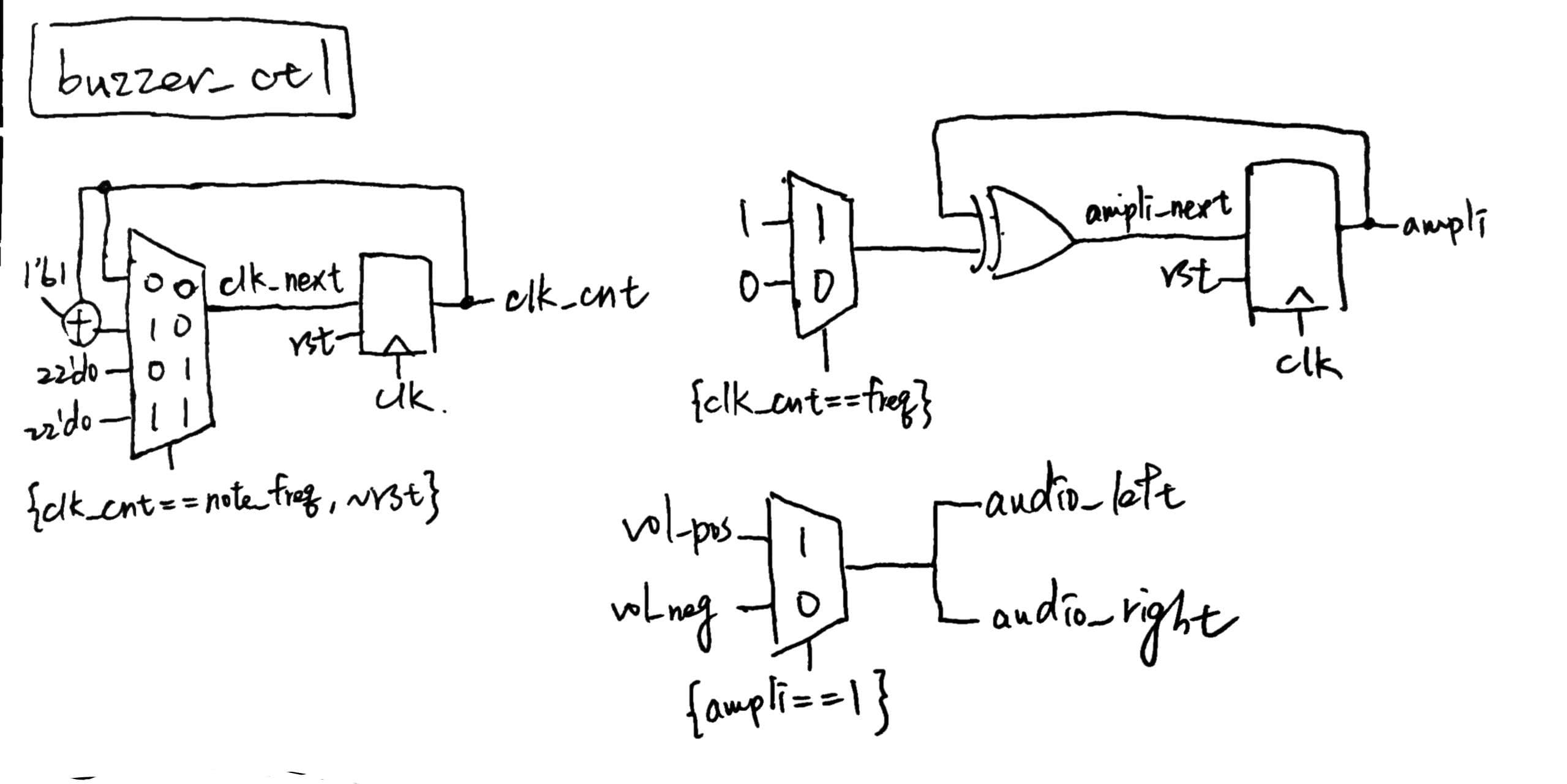
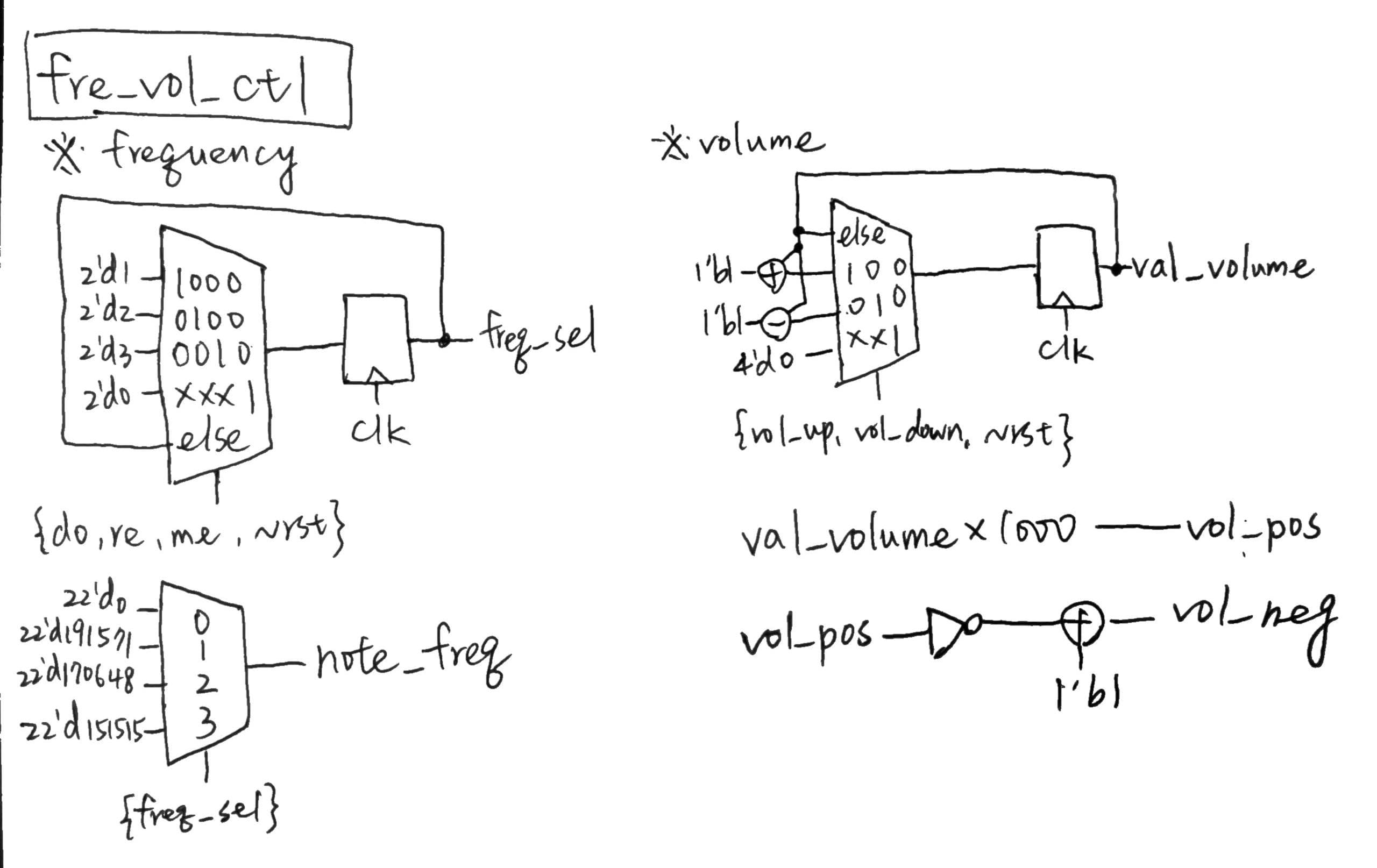
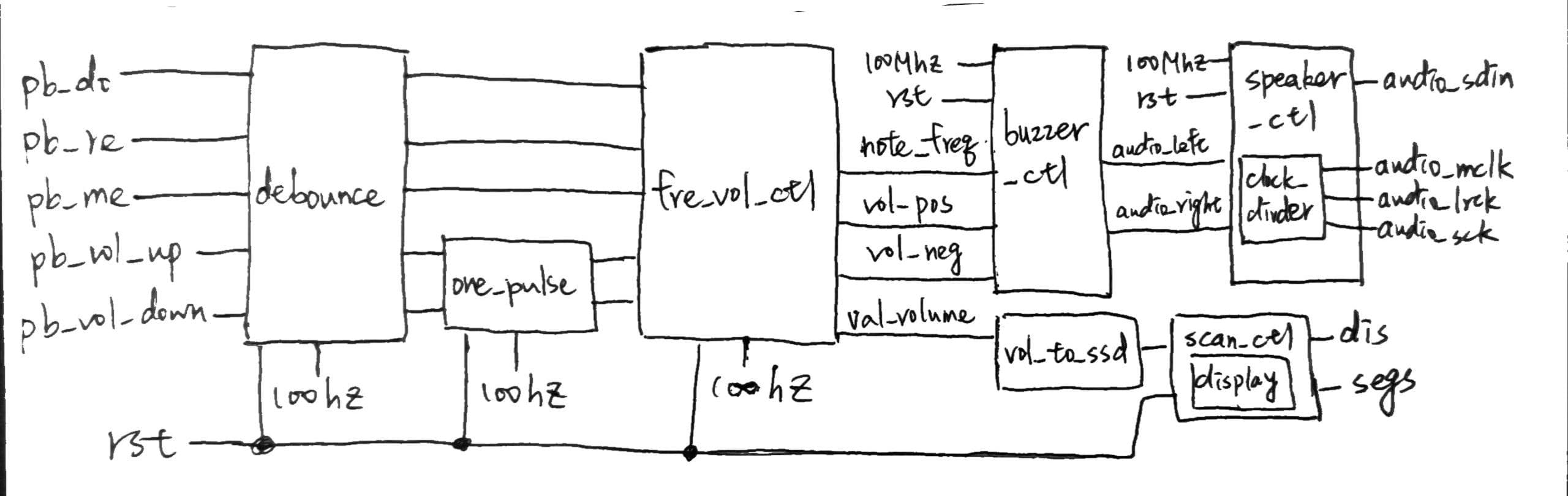
pb\_vol\_up(音量加1), pb\_vol\_down (音量減1)

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**



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。