**Lab8.1** 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, and 6.25 MHz sampling clock. And use Verilog simulation waveform to verify your control signal.

1.Design specification:

功能:二四小時/十二小時時鐘

輸入: sw\_rst(重置計數器)

clk\_100mhz (石英震盪器輸入)

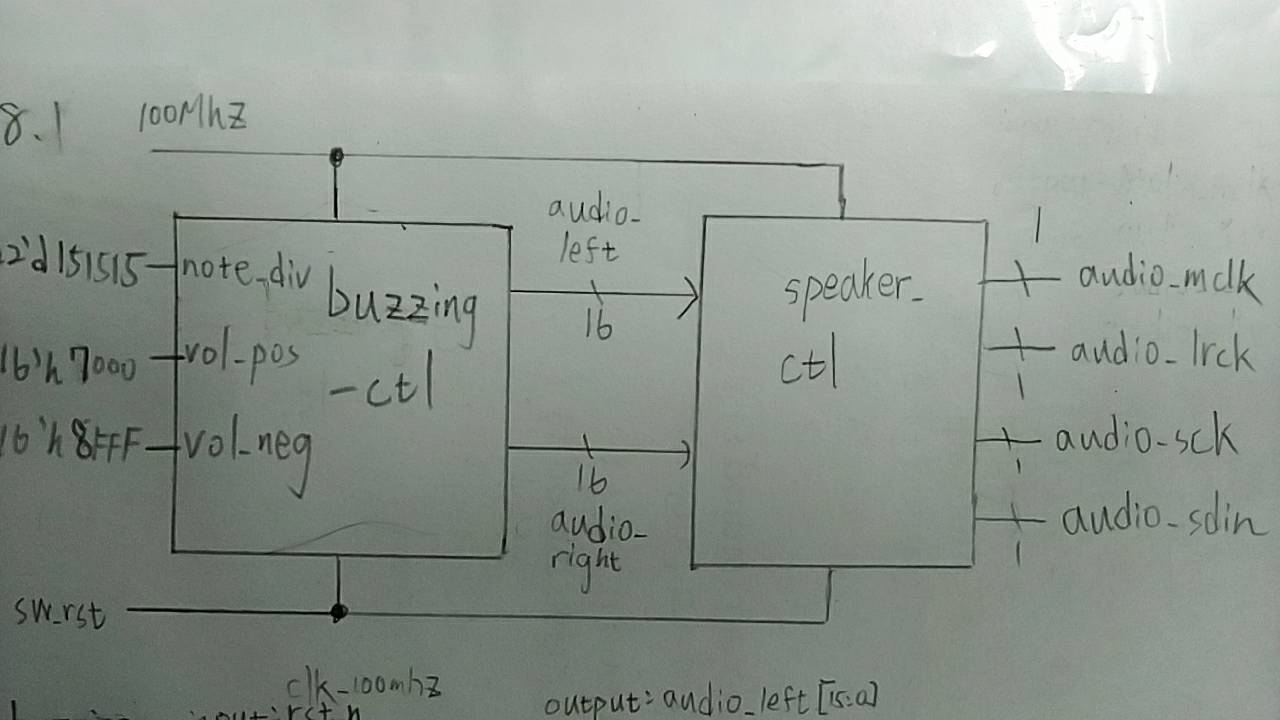
輸出: audio\_mclk (master clock輸出)

audio\_lrck (left\_right clock輸出)

audio\_sck (serial clock 輸出)

audio\_sdin (serial data 輸出)

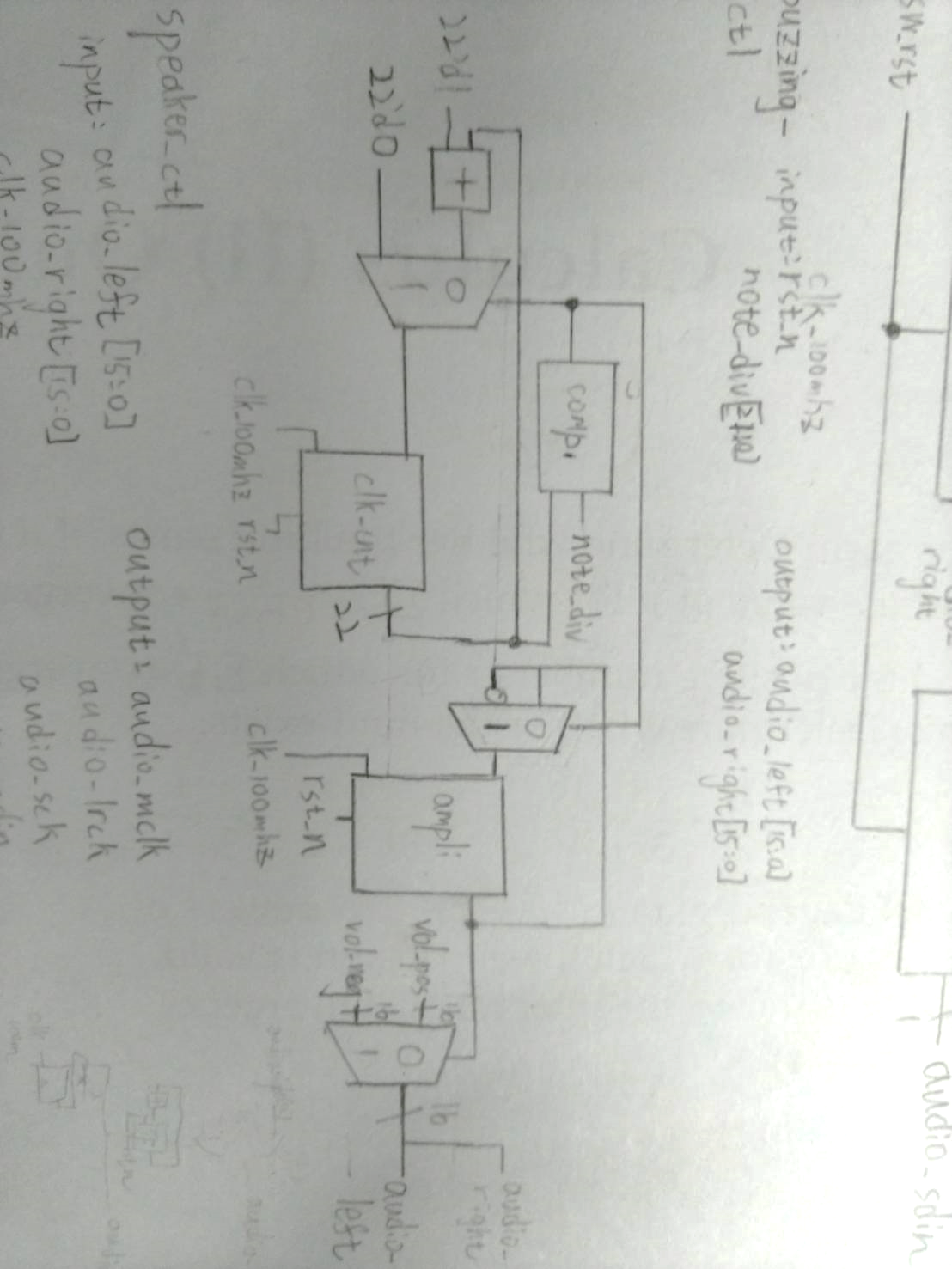
2. Design implementation



第一個lab，雖然題目只有要求要speaker\_ctl(parallel to serial and clock divider)的部分，但為了要能測試該模組，所以又加上了buzzing\_ctl以輸出訊號測試speaker\_ctl有無異常。

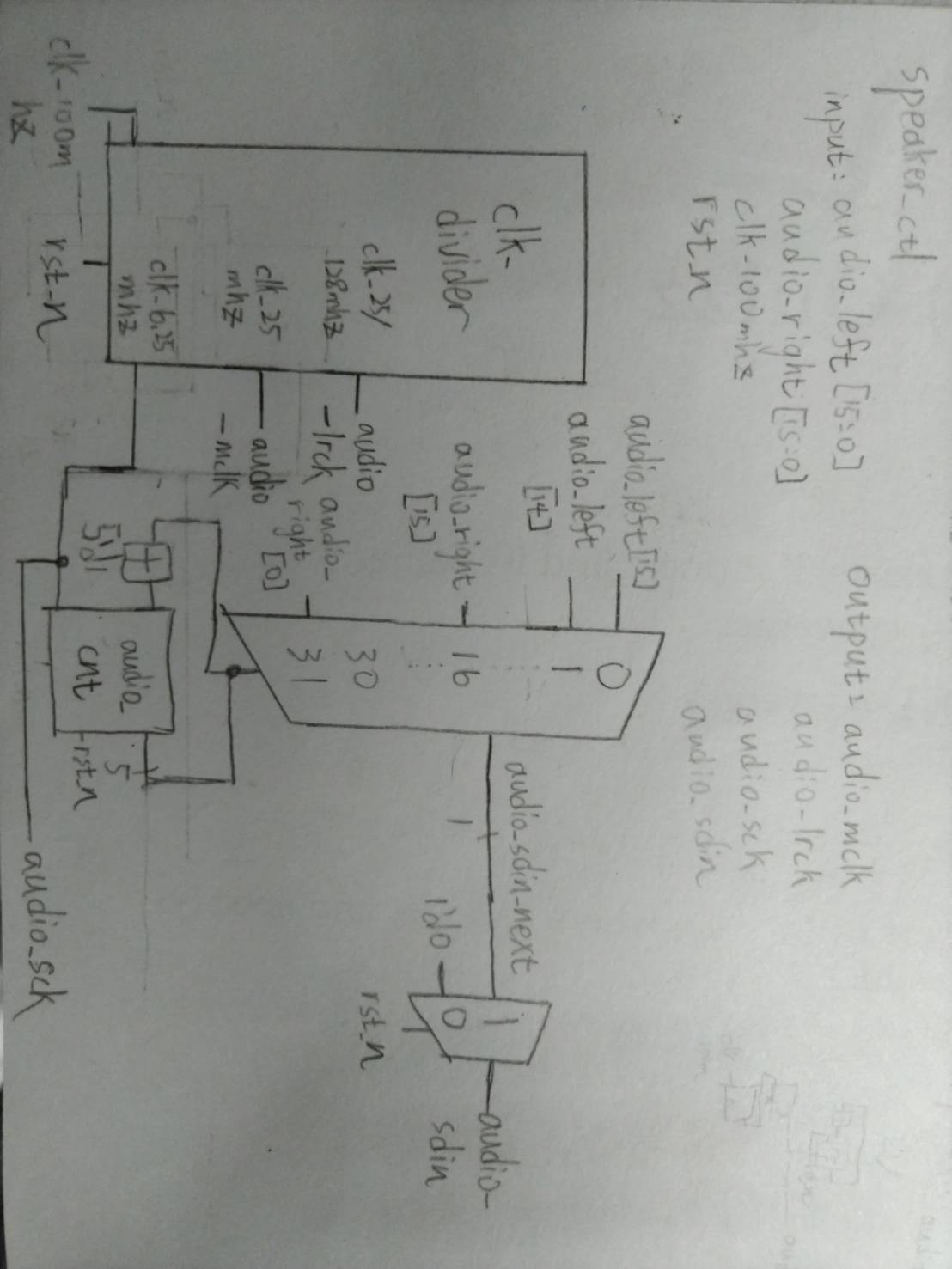
將sw\_rst(v17 DIP switch)向下撥又向上撥後，就重置了裝置一次。重置後，buzzing\_ctl會產生出原先時脈的note\_div倍波長的時脈，接著這個時脈在控制模組產生出和音波相近頻率且震幅高低由vol\_pos及vol\_neg決定的訊號(32bit)給speaker\_ctl。資料傳到speaker\_ctl後，speaker\_ctl會將此訊號轉成1bit，並且隨同3個clock一起被傳到speaker去，發出聲音。

Buzzing\_ctl:



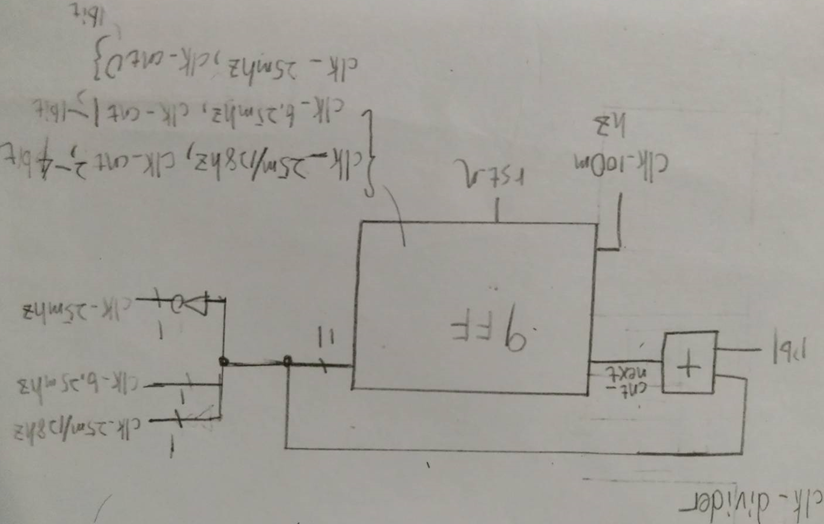
在這個模組裡，我們需要製造出和音波相近的時脈，頻率決定聲調的高低，輸出的數字決定音量的高低。我們首先會需要一個可以製造特定頻率的frequency divider。clk\_cnt為一22bit的暫存器，當每過一個100mhz的時脈時，會向上數一，另外其值會和note\_div (100M / 2 / 想要的頻率)在comparator中比較，如果clk\_cnt內的值大於或等於note\_div的值，則該comparator會輸出一，不僅重置clk\_cnt的值為零，還反置ampli (1bit)的值。Ampli是用來控制一個2 to 1多工器，ampli為零時輸出vol\_pos(16bit)的值各到audio\_right及audio\_left的值，為一時則輸出vol\_neg(16bit)的值。vol\_pos及vol\_neg共同決定speaker的聲音大小。

Speaker\_ctl:



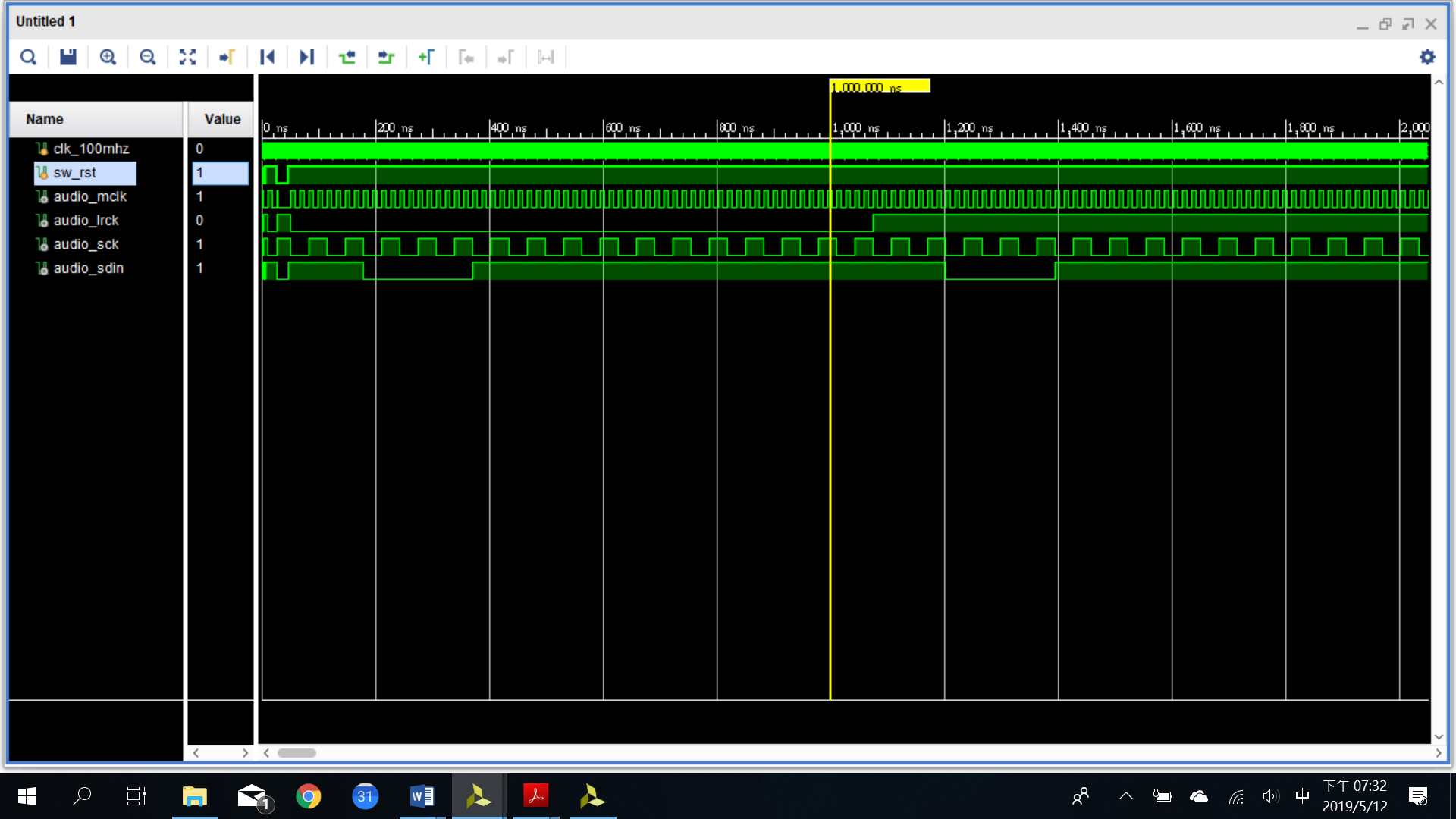
clk\_divider在此會輸出三個speaker需要的時鐘，結構會在下面介紹。這裡呢，主要是用來處理audio\_left和audio\_right的訊號轉換成1\_bit的audio\_sdin，首先6.25mhz的clk會被接到5-bit的計數器，每經過一個負緣audio\_cnt會往上加一，為31時下個其儲存的數字會歸零。audio\_cnt會用來選擇上面的多工器，也因此會同步audio\_sck同步改變訊號。另外，rst\_n後audio\_cnt的數字預設為30，這是為了對應rst\_n後clk\_divider及speaker要求的延遲輸出。

Clk\_divider:



clk\_divider是一個大型的asynchronous加法器，因為我們需要的時脈波長都是2的倍數。首先，我們需要最低25/128hz的時脈，所以需要用到9-bit的多工器，接著暫存器再過每一個clk\_100mhz時都會往上加一，到9’b111111111時會輸入歸零。拉出右下往上數第二個及第四個及最上面的暫存器作為輸出的時脈訊號，clk\_25mhz加上一個not，使得當clk\_25mhz正緣時，其他兩個clk為負緣。

Testbench:



由上圖可看出當rst後，audio\_sdin會先出現audio\_right的最後一個bit(vol\_neg先，值為8FFF)，由rst後先有兩個高電位，而後三個低電位，再來又高電位可知。然後audio\_lrck過了十六個audio\_sck後會變高電位，重新輸出vol\_neg，值也正常，可得知這個模組運作正常。

3. I/O pin assignment:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 變數接收處 | 晶片I/O點 | 描述 |
| 輸入 | clk\_100mhz | W5 | 石英震盪器時脈輸入 |
|  | sw\_rst | V16 | 重置時鐘 |
| 輸出 | audio\_mclk | A14 |  |
|  | audio\_lrck | A16 |  |
|  | audio\_sck | B15 |  |
|  | audio\_sdin | B16 |  |

**Lab8.2** Produce the buzzer sounds of Do, Re, and Mi by pressing buttons. Control volume by pressing up or down bottoms.

1.Design specification:

功能:二四小時/十二小時時鐘

輸入: sw\_rst(重置計數器)

clk\_100mhz (石英震盪器輸入)

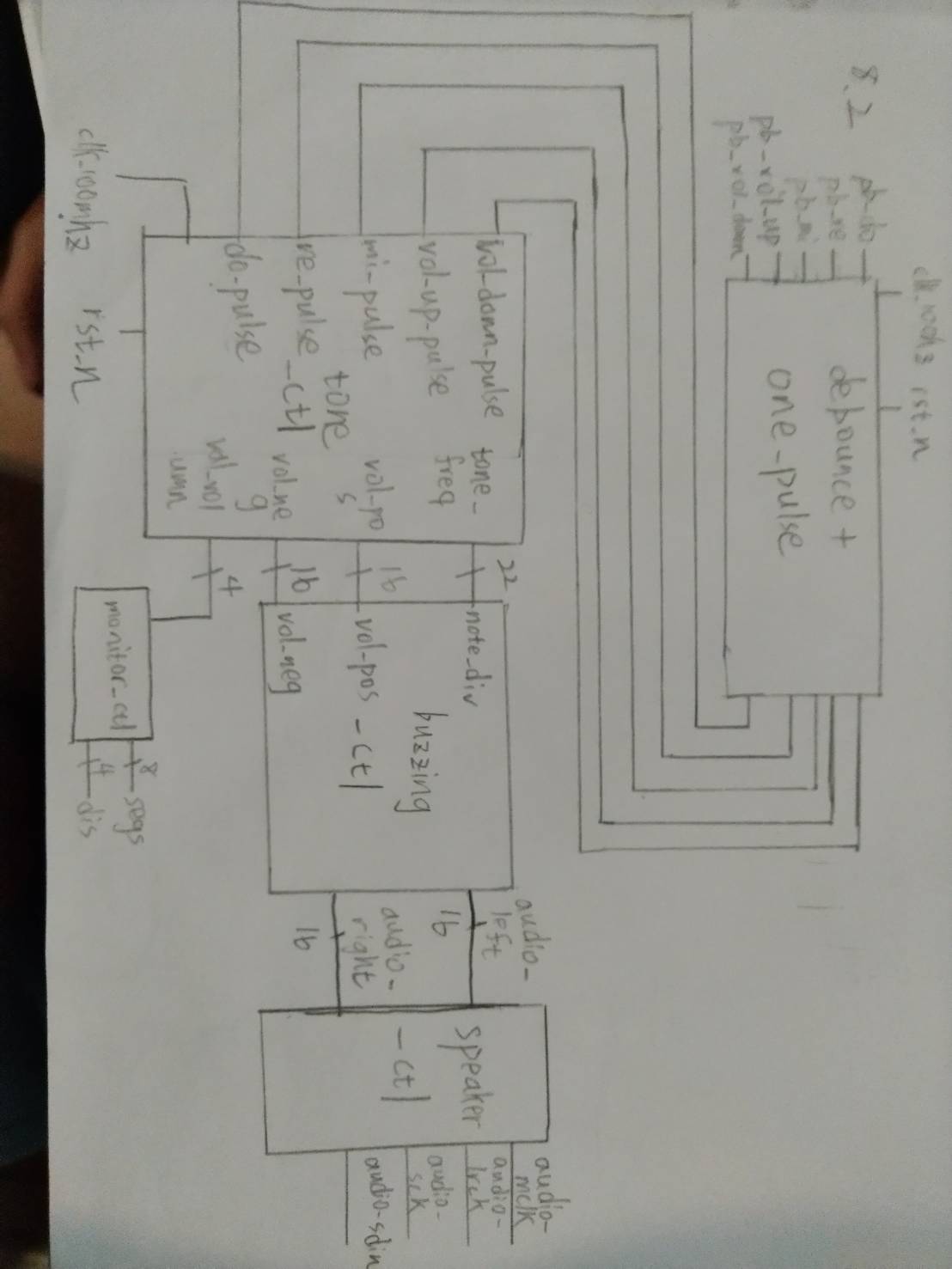
輸出: audio\_mclk (master clock輸出)

audio\_lrck (left\_right clock輸出)

audio\_sck (serial clock 輸出)

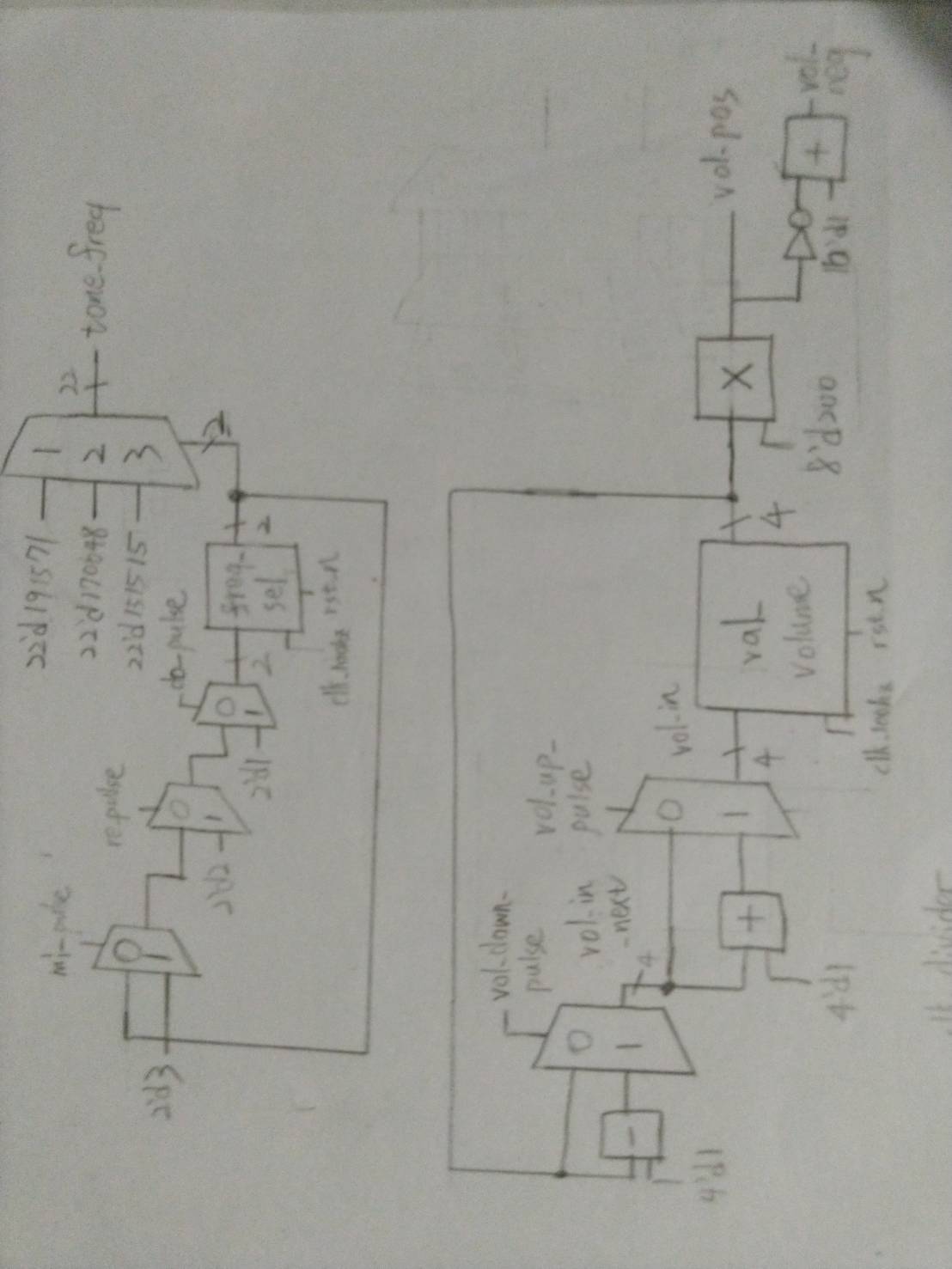
audio\_sdin (serial data 輸出)

2. Design implementation



第二題和第一題最大的區別是能調整發出聲音的頻率和大小。控制的地方在於buzzing\_ctl的note\_div控制頻率，vol\_neg及vol\_pos控制聲音大小。所以需要tone\_ctl來控制這些變數。按鈕的訊號經過debounce和one\_pulse處理後會變成one\_pulse，然後輸入tone\_ctl後，改變聲音和頻率的參數，之後經由buzzing\_ctl及speaker\_ctl，輸入到speaker。

tone\_ctl:



這個模組可以分成兩個部分，圖的上方專門用來控制頻率，圖的下方則用來控制音量大小。上面有暫存器freq\_sel，其儲存的值會接到多工器，用以選擇評綠的輸出。1是do，2是re，3是mi。當什麼都不按時，freq\_sel會一直輸入原本的值，按下do\_pulse時，多工器會輸入1，以此類推。

下面呢，則是有暫存器vol\_volume，其儲存的值會接到乘法器，讓輸出到vol\_pos的值為200 \* val\_volume。而vol\_neg的值則是vol\_pos的2’s complement。當按下down按鈕時，下個正緣會輸入比原先減一的值進去，按up時則輸入加一的值，不按則不動。

3. I/O pin assignment:

|  |  |  |  |
| --- | --- | --- | --- |
|  | 變數接收處 | 晶片I/O點 | 描述 |
| 輸入 | clk\_100mhz | W5 | 石英震盪器時脈輸入 |
|  | sw\_rst | V16 | 重置時鐘 |
| 輸出 | audio\_mclk | A14 |  |
|  | audio\_lrck | A16 |  |
|  | audio\_sck | B15 |  |
|  | audio\_sdin | B16 |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | seg[7] | W7 | 控制七段顯示器腳位 |
|  | seg[6] | W6 |  |
|  | seg[5] | U8 |  |
|  | seg[4] | V8 |  |
|  | seg[3] | U5 |  |
|  | seg[2] | V5 |  |
|  | seg[1] | U7 |  |
|  | seg[0] | V7 |  |
|  | dis[3] | W4 | 控制每個七段顯示器開關 |
|  | dis[2] | V4 |  |
|  | dis[1] | U4 |  |
|  | dis[0] | U2 |  |

問題與討論:

這次的lab雖然是新的東西，但是難度和上一個計時器來比，難度簡直是天壤之別。整體上打的都很順，只有在接腳時出了問題，因為我是照著老師第二頁投影片去弄，結果聲音一直都出不來，結果是lrck和mlck畫反了，浪費了我好幾個小時debug，希望教授的投影片能改正一下，以免讓後人誤會。