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)

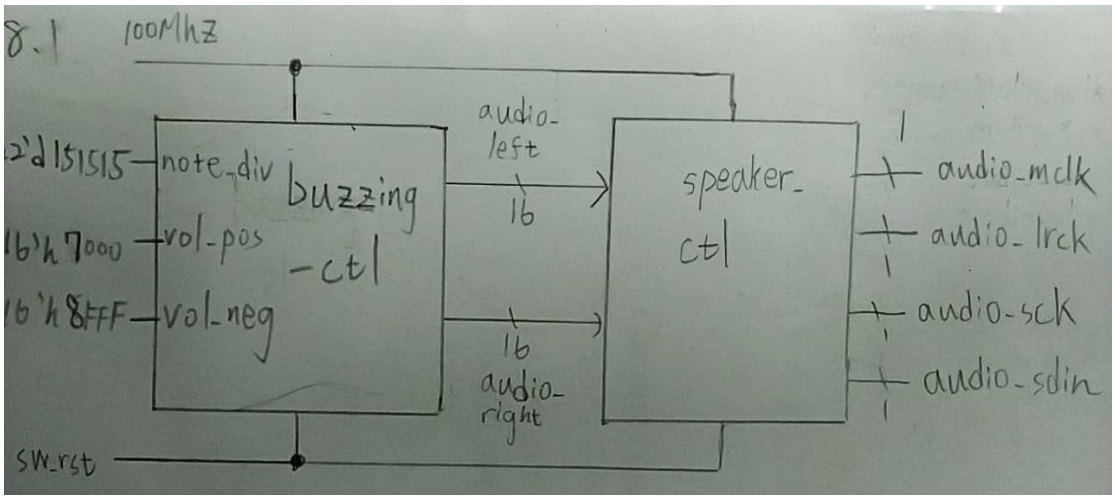
輸出: audio_mclk (master clock 輸出)

audio_lrck (left_right clock 輸出)

audio_sck (serial clock 輸出)

audio_sdin (serial data 輸出)

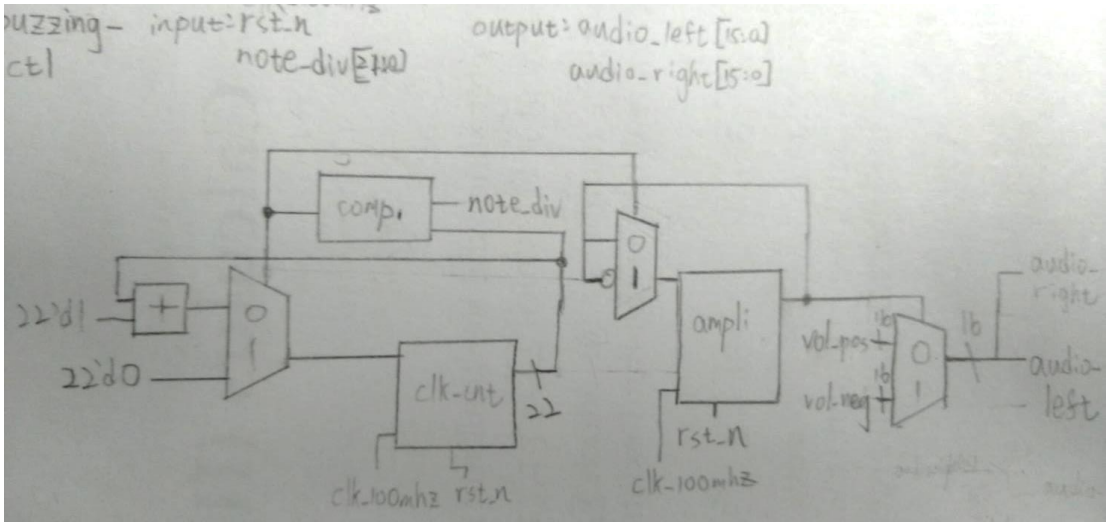
2. Design implementation



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。

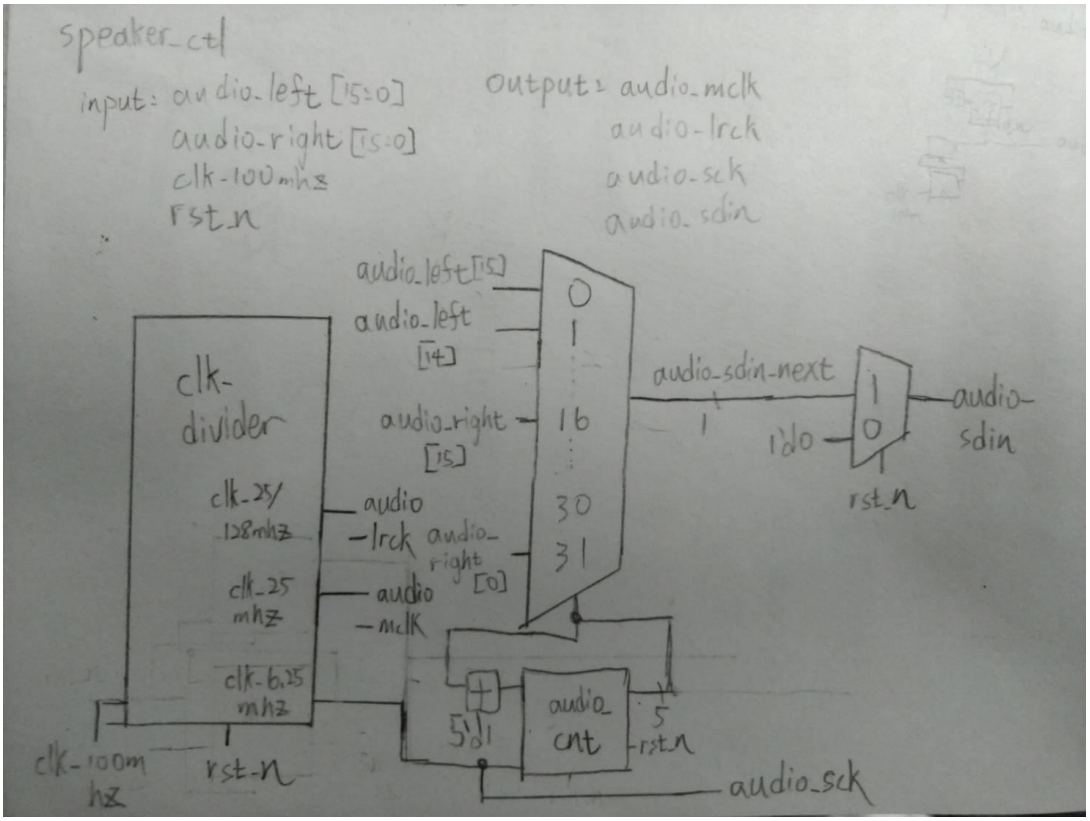
將 sw_rst(v17 DIP switch)向下撥又向上撥後，就重置了裝置一次。重置後，buzzing_ctl 會產生出原先時脈的 note_div 倍波長的時脈，接著這個時脈在控制模組產生出和音波相近頻率且震幅高低由 vol_pos 及 vol_neg 決定的訊號(32bit)給 speaker_ctl。資料傳到 speaker_ctl 後，speaker_ctl 會將此訊號轉成 1bit，並且隨同 3 個 clock 一起被傳到 speaker 去，發出聲音。

Buzzing_ctl:



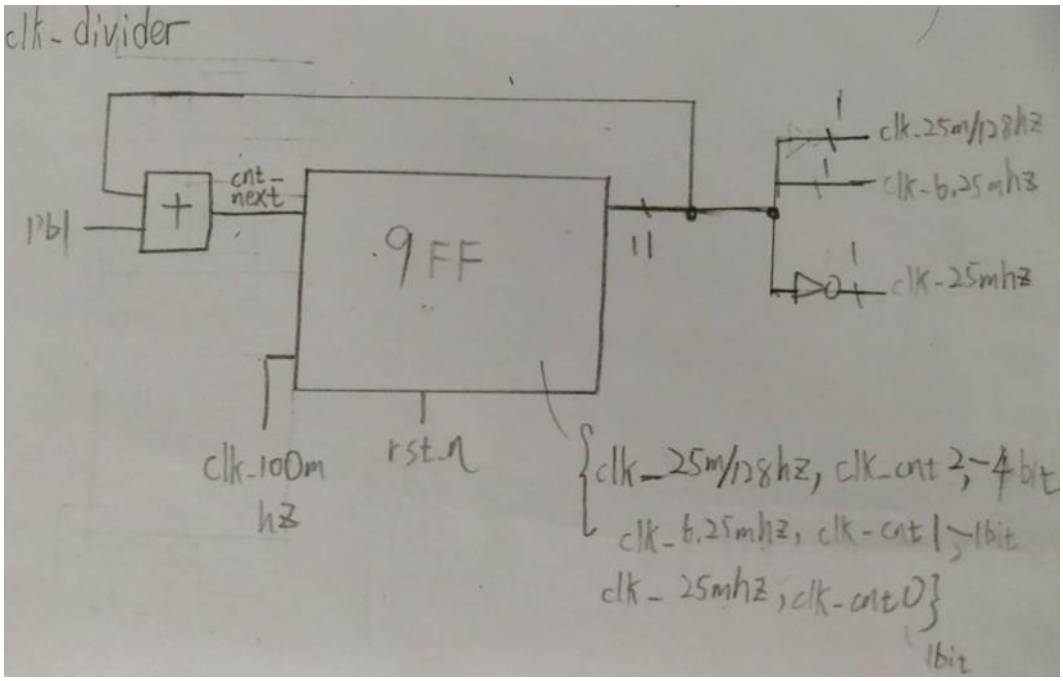
在這個模組裡，我們需要製造出和音波相近的時脈，頻率決定聲調的高低，輸出的數字決定音量的高低。我們首先會需要一個可以製造特定頻率的 frequency divider。clk_cnt 為一 22bit 的暫存器，當每過一個 100mhz 的時脈時，會向上數一，另外其值會和 note_div (100M / 2 / 想要的頻率)在 comparator 中比較，如果 clk_cnt 內的值大於或等於 note_div 的值，則該 comparator 會輸出一，不僅重置 clk_cnt 的值為零，還反置 ampli (1bit)的值。Ampli 是用來控制一個 2 to 1 多工器，ampli 為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



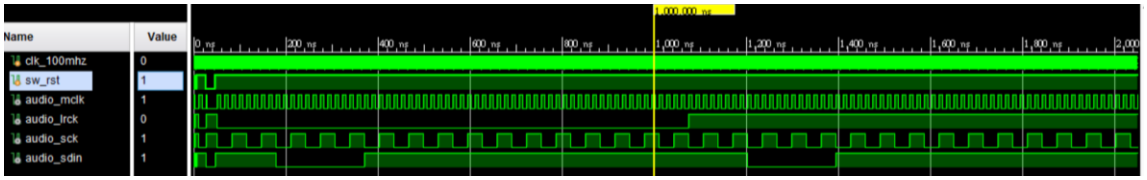
clk_divider 在此會輸出三個 speaker 需要的時鐘，結構會在下面介紹。這裡呢，主要是用來處理 audio_left 和 audio_right 的訊號轉換成 1_bit 的 audio_sdin，首先 6.25mhz 的 clk 會被接到 5-bit 的計數器，每經過一個負緣 audio_cnt 會往上加一，為 31 時下個其儲存的數字會歸零。audio_cnt 會用來選擇上面的多工器，也因此會同步 audio_sck 同步改變訊號。另外，rst_n 後 audio_cnt 的數字預設為 30，這是為了對應 rst_n 後 clk_divider 及 speaker 要求的延遲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

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)

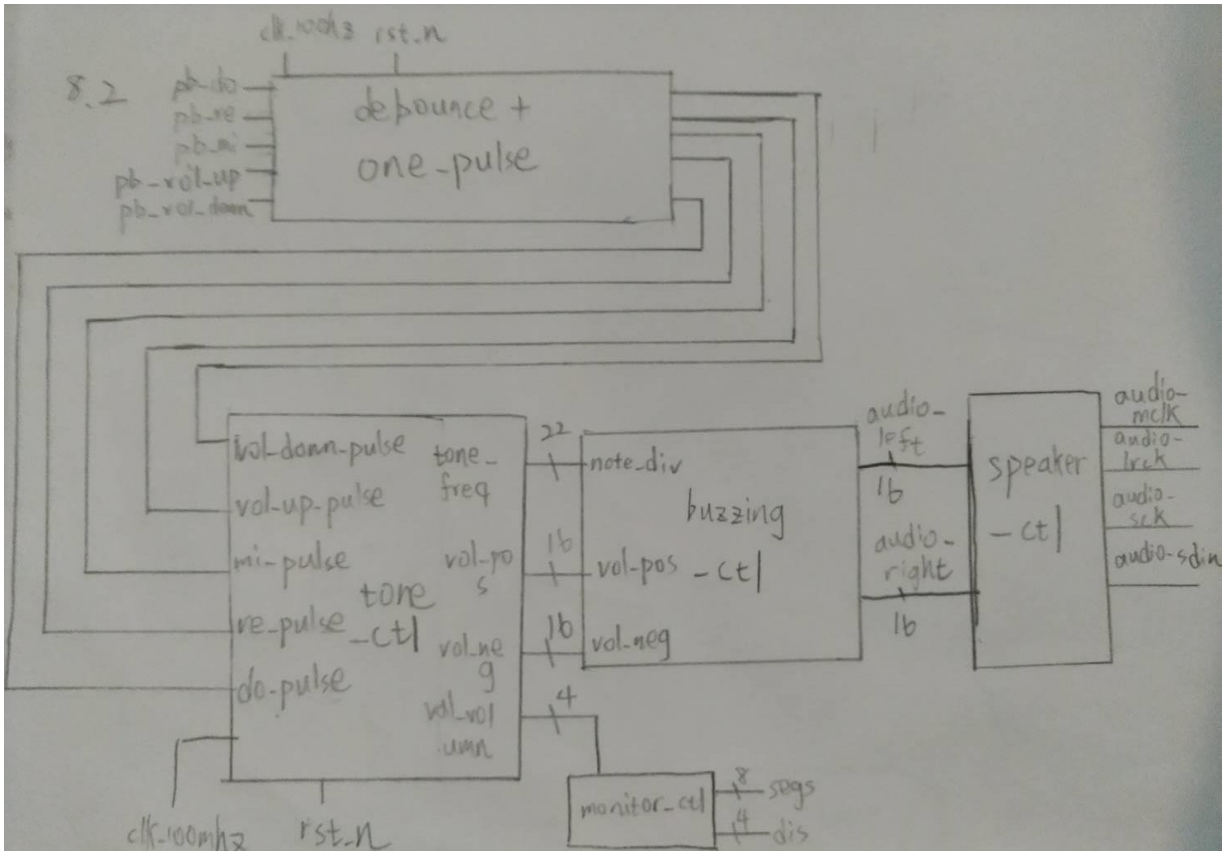
輸出: audio_mclk (master clock 輸出)

audio_lrck (left_right clock 輸出)

audio_sck (serial clock 輸出)

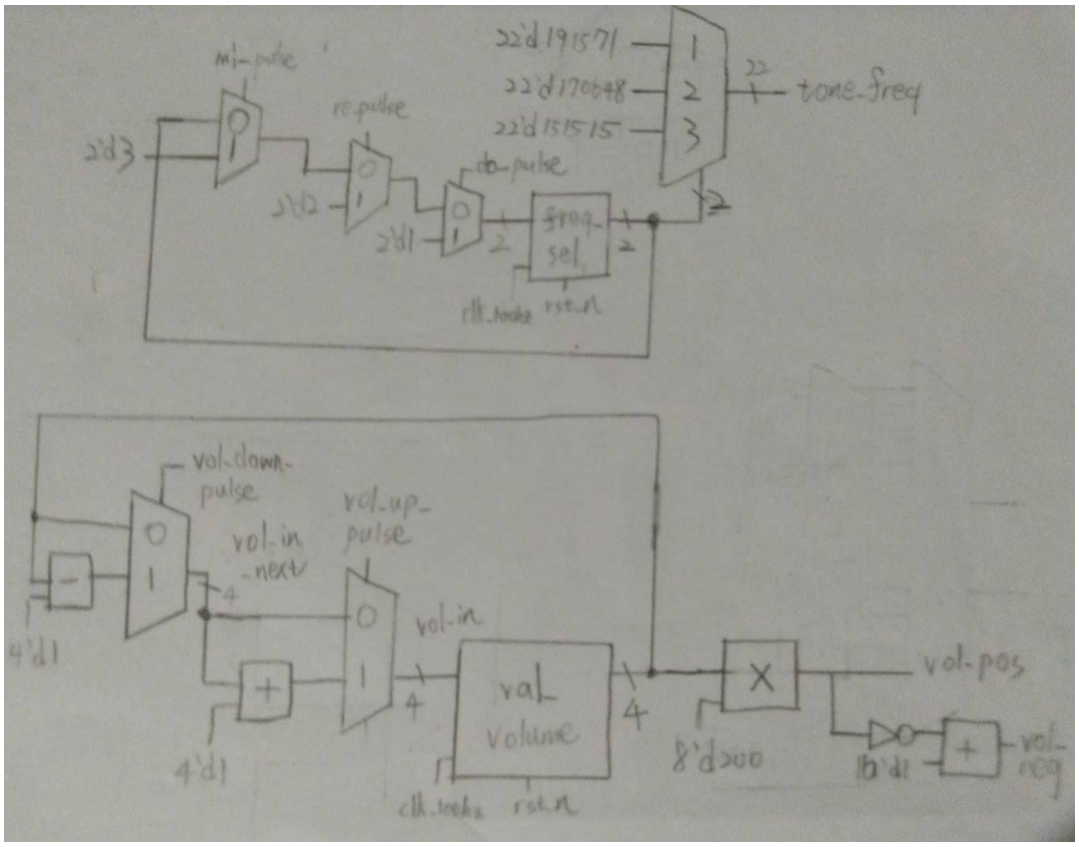
audio_sdin (serial data 輸出)

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。