**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.**

**1. Specification**

**內容: 寫下你的電路中的inputs, outputs以及其bit widths，名稱必須跟你的verilog code中相同。**

module speaker(

output audio\_mclk, // master clock

output audio\_lrck, // left-right clock

output audio\_sck, // serial clock

output audio\_sdin, // serial audio data input

input clk, // clock from the crystal

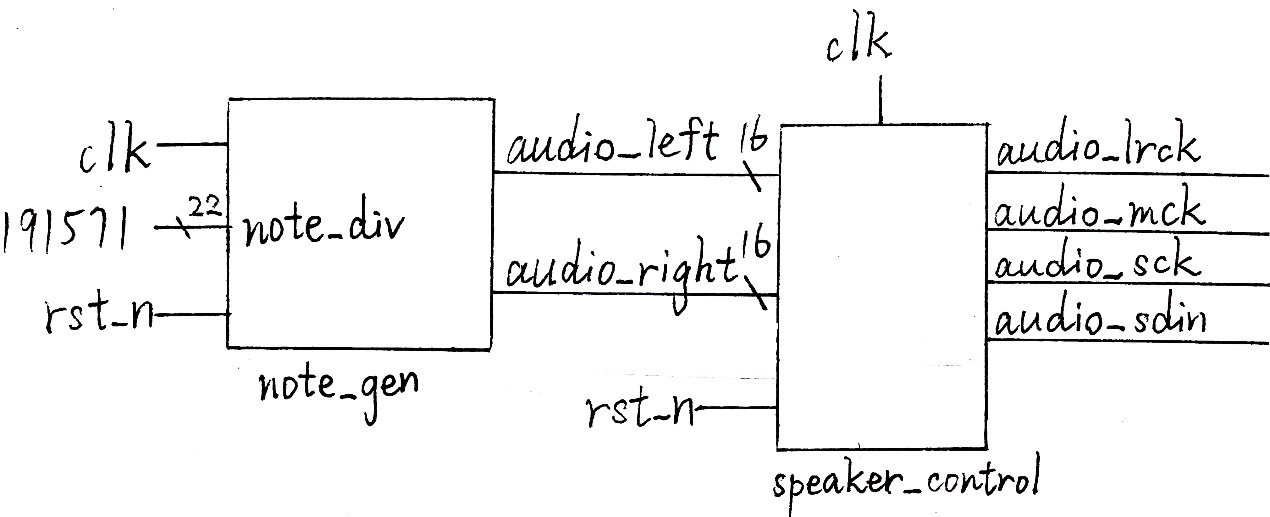
input rst\_n // active low reset

);

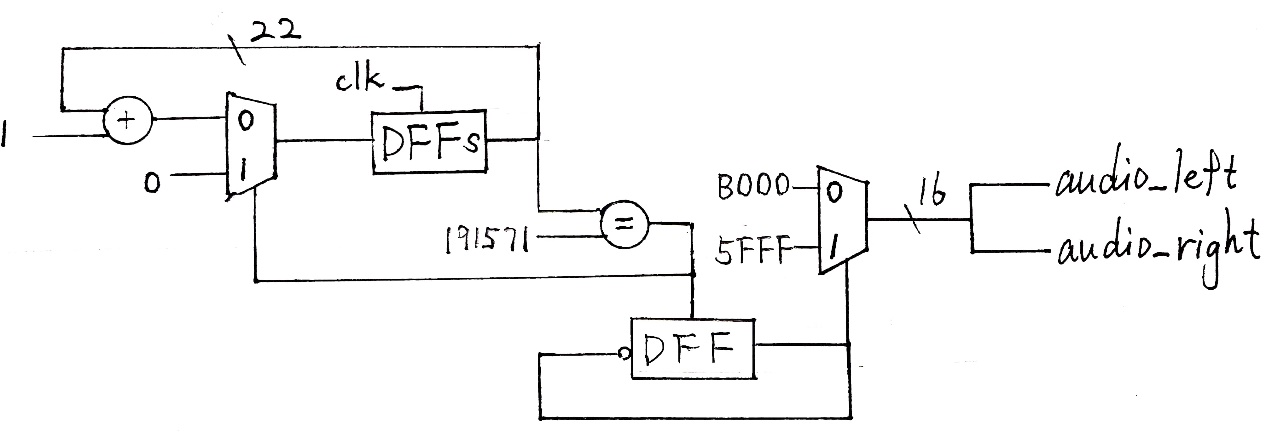
**2. Block Diagram**

**內容: 電路中的Block diagram(可以用手畫拍照或電腦繪圖)。**

speaker.v : (top)



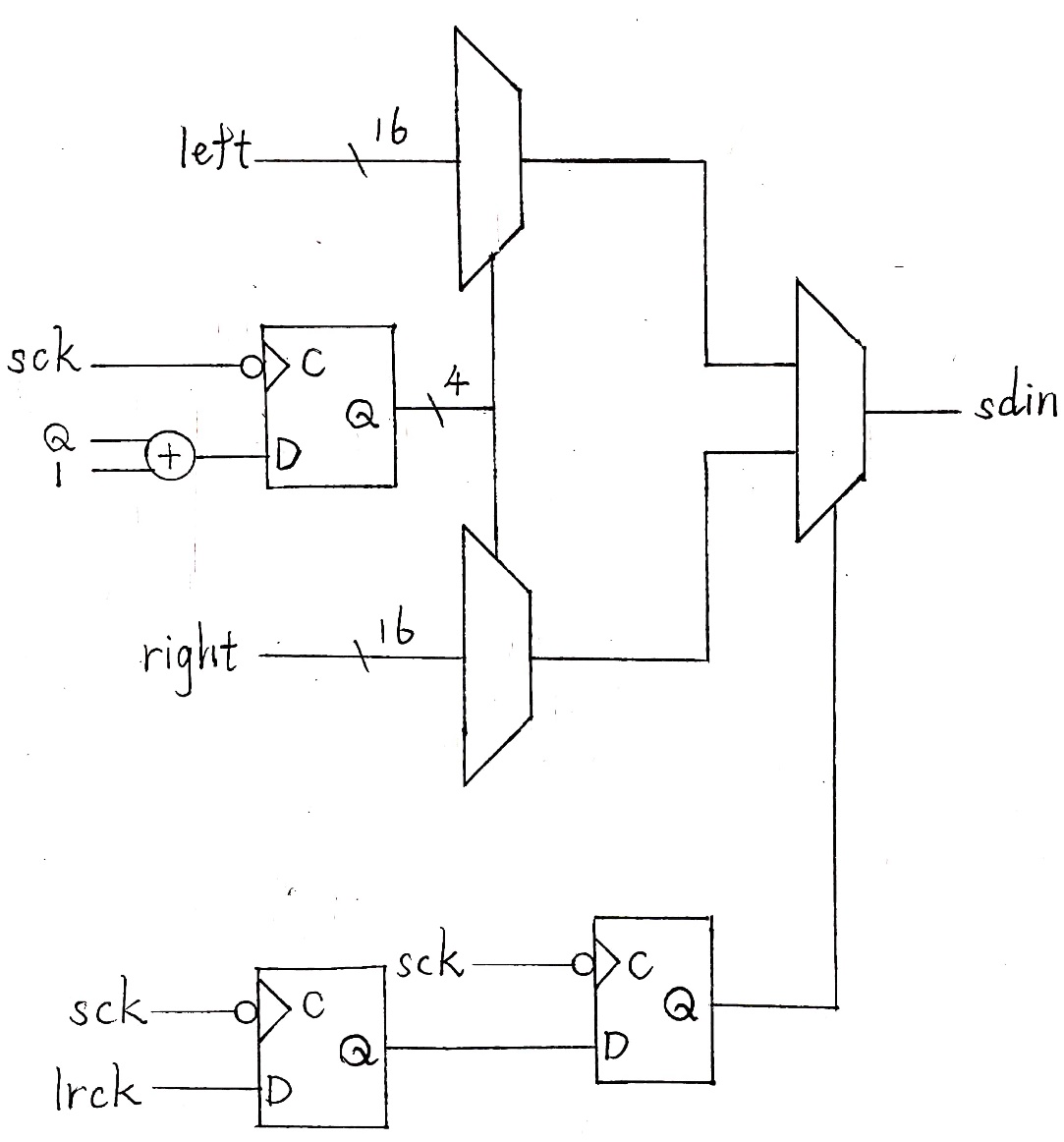
note\_gen.v :



mclk\_lrck\_sck.v :



speaker\_control.v :



**3. Finite state machine**

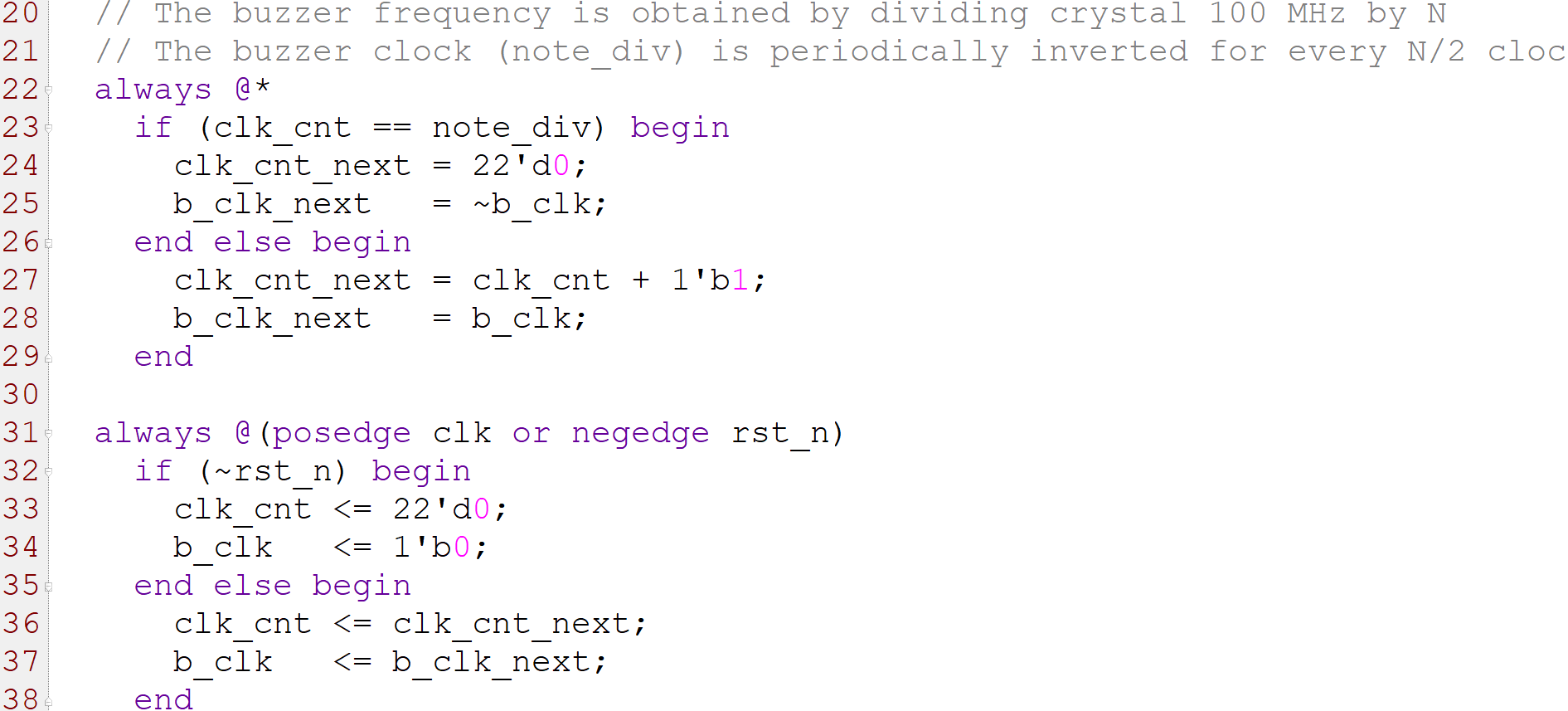
**內容: 電路中的Finite state machine，若無則寫無。**

無

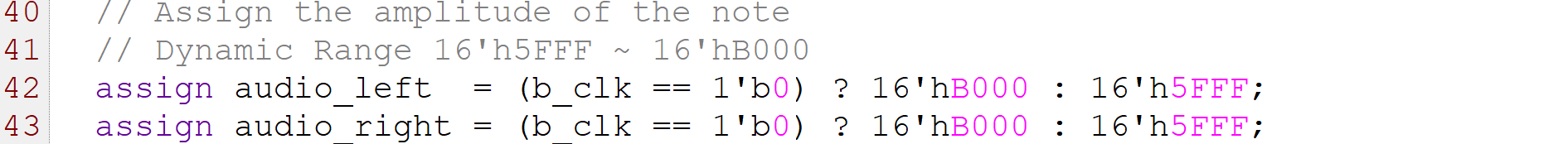
**4. Implement**

**內容: 請列出相關的logic function、詳細用文字解釋電路的運作方法、結果等等，可以貼code解釋或拍FPGA輔助解釋(但不能只貼code跟FPGA結果)。**

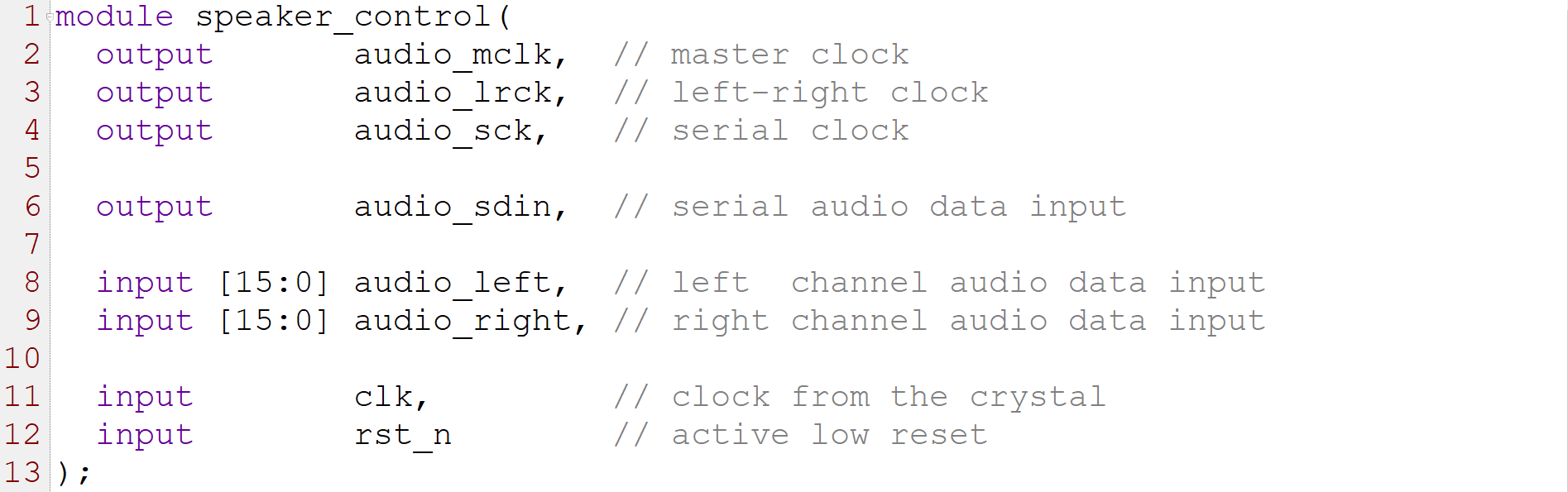
note\_gen.v跟老師的投影片一樣，用一個二十二位元的計數器和一個T Flip-flop來產生所需的頻率。

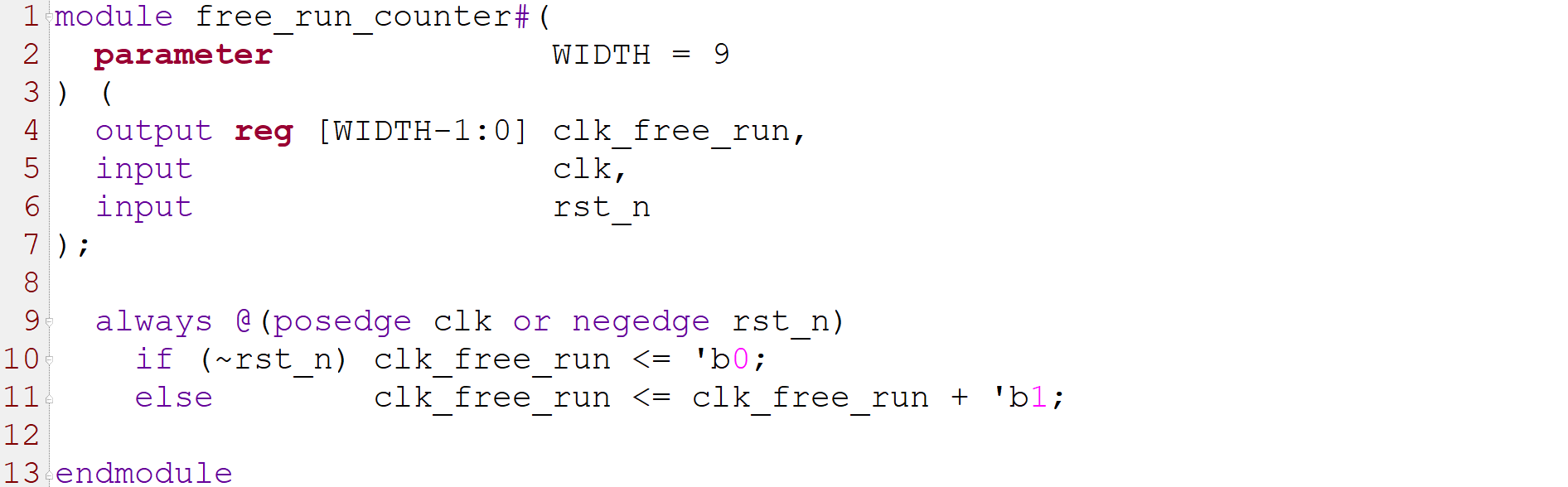


把產生的頻率連接到MUX，以產生適當大小的震幅。

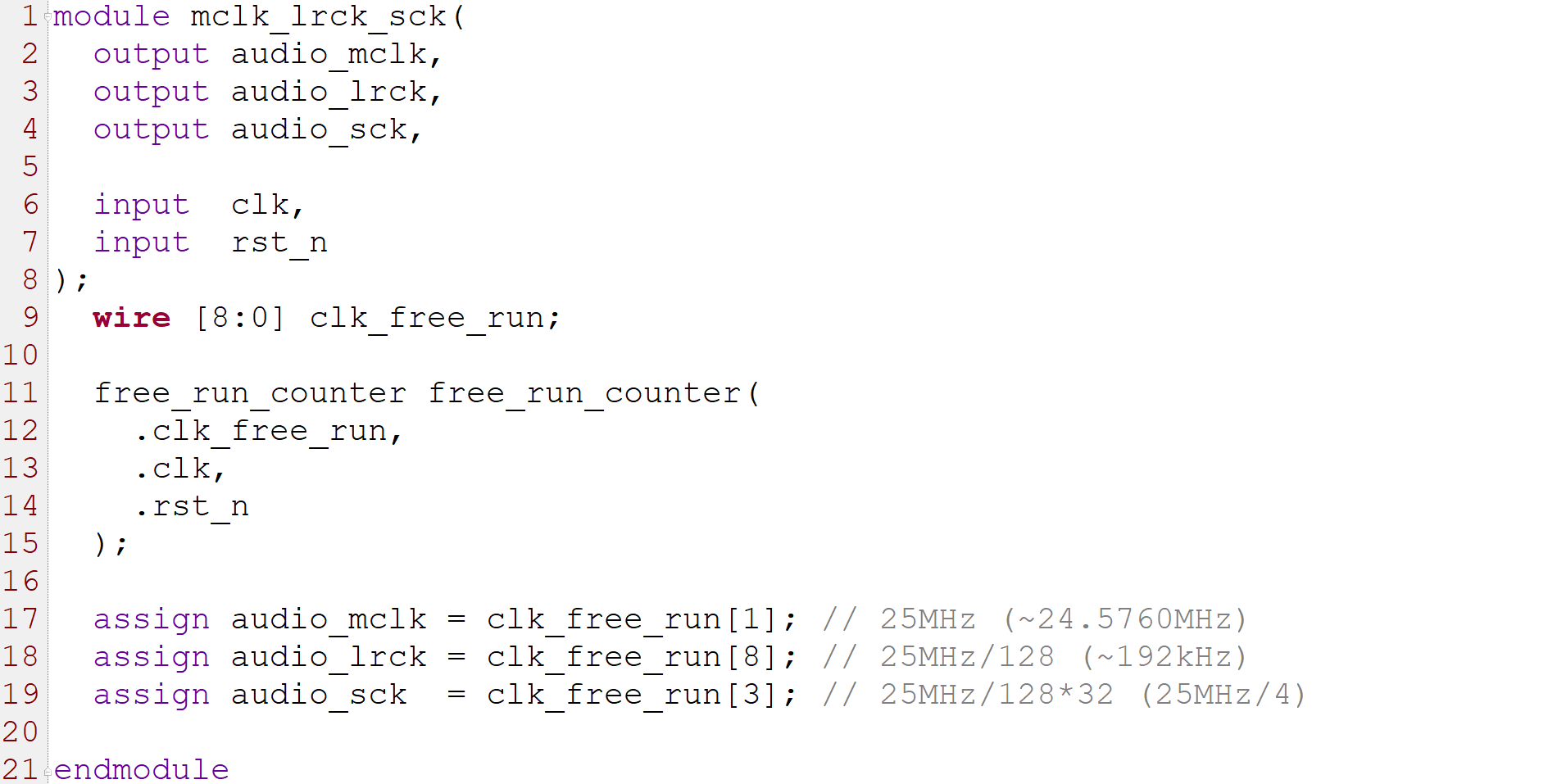


這樣就可以產生任意頻率的方波了。

我的speaker\_control.v的input和output如下。

首先，先依據100MHz的時脈，用除頻器除以四，以創造出25MHz的mclk。再除以四，則製造出sck。再除以三十二，則可以製造出lrck。為了使上述三個時脈同步，我先使用free\_run\_counter.v來製造出所有可能會用到的二次冪頻率。

然後mclk\_lrck\_sck.v就可以直接把特定的clk\_free\_run連接到mclk、sck、lrck。

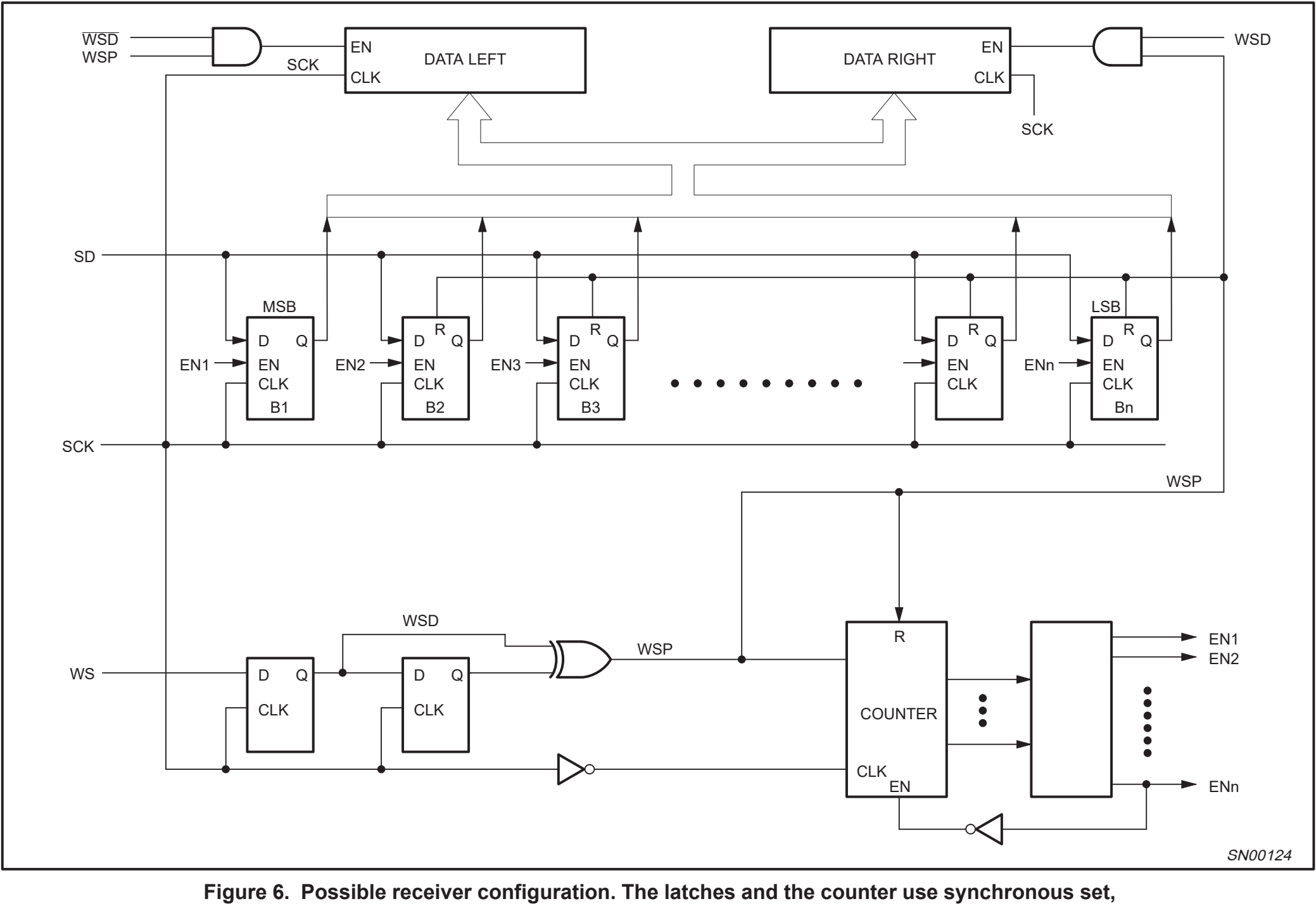


這樣就可以直接於speaker\_control.v裡面使用上述兩個模組所產生的mclk、sck、lrck。

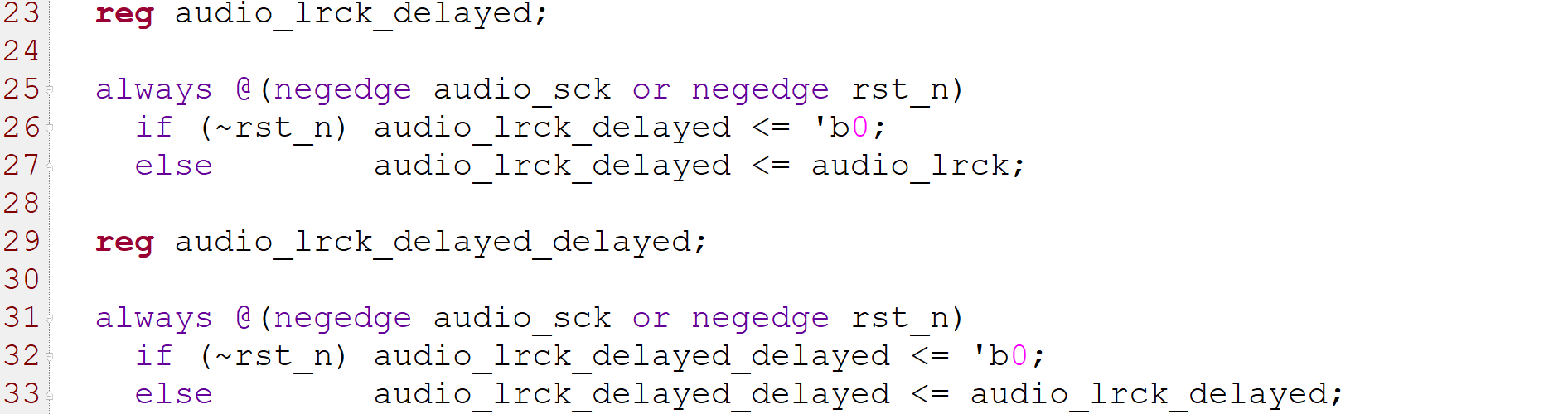


至此，speaker\_control.v的output只剩sdin還沒製造出來，其餘mclk、sck、lrck都有了。接下來要想辦法製造出符合DAC CS4344的規定的sdin。

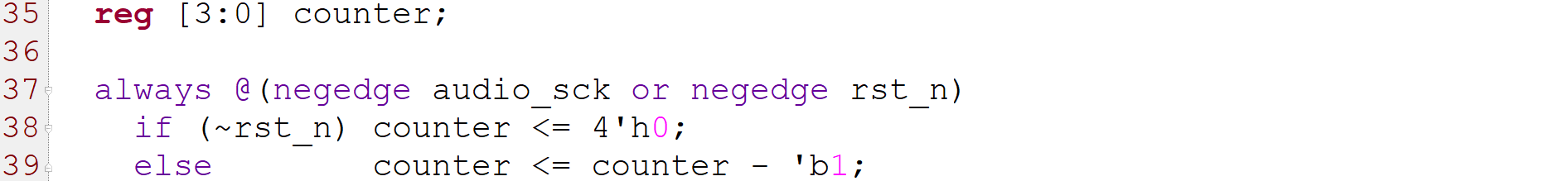
如同老師的影音檔內所述，sdin的MSB必須要比lrck的riding edge或falling edge晚一個sck週期。這個部分我的作法參考自一份I2S bus specification文件中所附的block diagram，如下圖。（Philips Semiconductors; February 1986; Revised: June 5, 1996）留意下圖的左下角的那兩個D Flip-flop，這是一個可以產生這種「sdin比lrck慢一拍」的效果的絕妙方法。

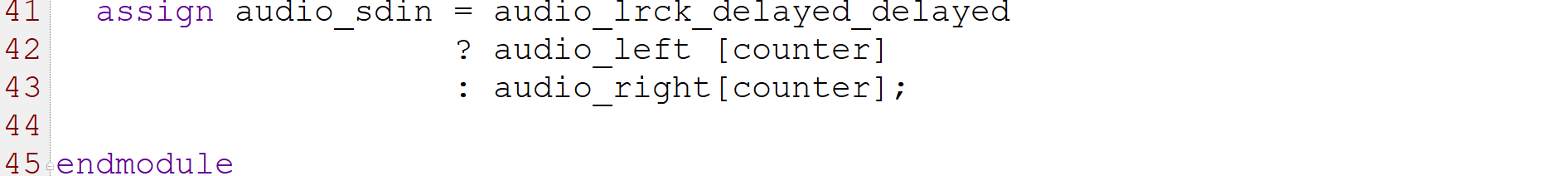


我先使用D Flip-flop來製造出一個比正常的lrck晚一個sck週期的訊號。這裡要使用兩個D Flip-flop的原因是因為，如果只使用一個的話，此D Flip-flop會因為連接到D的lrck與連接到CLK的sck本來就是同步訊號的關係，而使其產生出來的lrck\_delayed沒有被延遲一個sck週期。但是，我們並非徒勞無功，因經過這第一個D Flip-flop處理之後所產生的lrck\_delayed已經晚於sck的falling edge，所以lrck\_delayed不可能在同一個sck週期內再穿過任何其他一樣由sck的falling edge作為CLK的D Flip-flop。因此，如果在此時另外多使用一個D Flip-flop來串連，其輸出lrck\_delayed\_delayed就一定會剛好比正常的lrck晚一個sck週期。綜上所述，我可以用verilog描述如下。



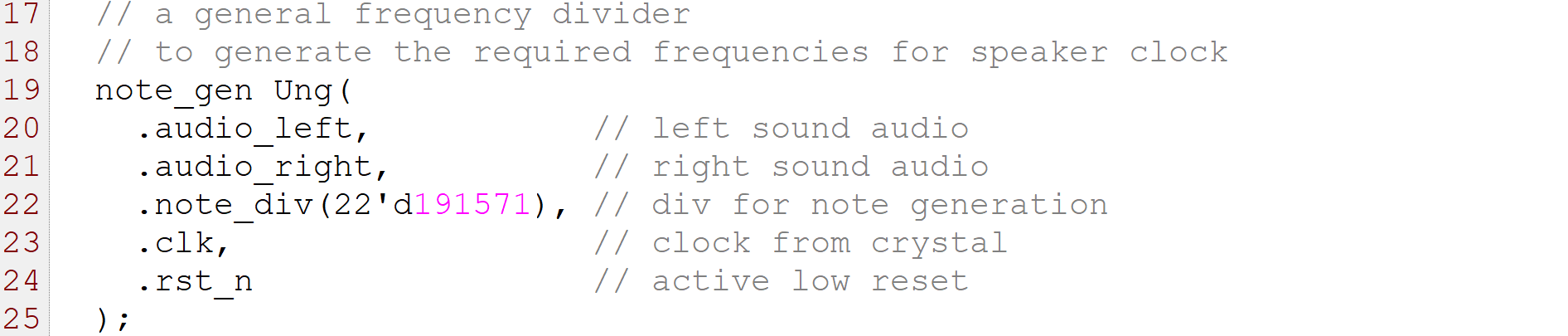
因audio\_left和audio\_right都是parallel-loading的，所以需要建構parallel-to-serial的功能。我仿照I2S bus specification文件中所附的block diagram中右下角的方法，使用一個四位元的計數器，以便在每一個lrck中的不同的sck中，分別讓audio\_left和audio\_right的不同位元連接到sdin。因為是由MSB到LSB倒序傳送，所以計數器要下數。

雖然I2S bus specification文件圖中畫的是數個latch，但其實不需要在verilog中描述出來。我們可以直接用audio\_left[counter] 和audio\_right[counter] 達到相同的效果。

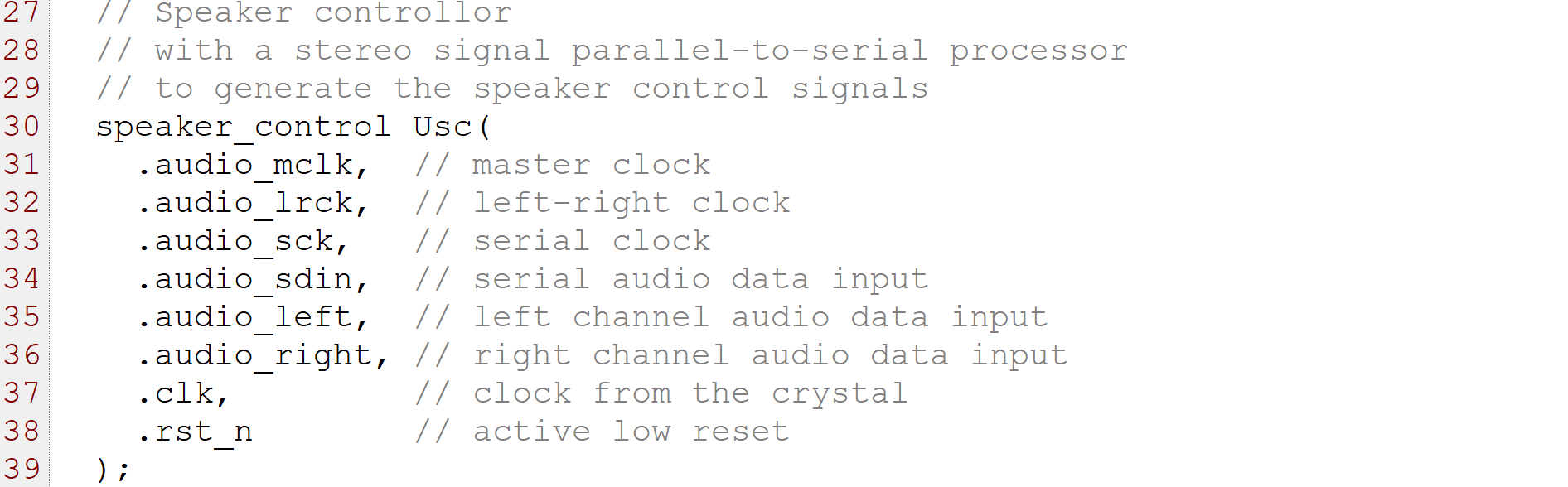


至此，我們已將所有需要使用到的模組建構完畢。

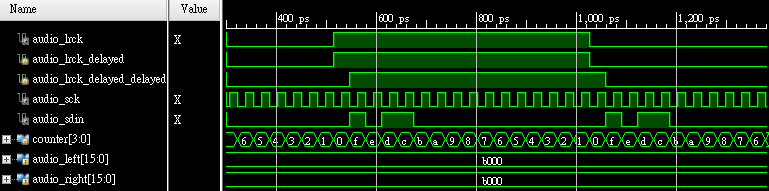
speaker.v (top) 的寫法跟著老師的PPT一步一步照樣做就好，把該接的線接一接即可。先用note\_gen.v製造特定頻率、特定震幅的聲音資料。



再連接到parallel-to-serial，並產生output所需的mclk、sck、lrck等訊號，這樣就大功告成了。



如下圖，第一個訊號lrck是輸出到DAC的，第二個lrck\_delayed是經過一次D Flip-flop處理後的訊號，而lrck\_delayed\_delayed則是經過兩次D Flip-flop的訊號。圖中可見，lrck\_delayed\_delayed比輸出到DAC的lrck慢了一個sck週期。且由於counter是在重置之後是從4’h0開始下數的，因此在lrck的rising或falling edge之後的第一個sck內的數值都會是4’h0，然後到了第二個sck的時候才會下數減一，變成4’hF。

****

上圖要傳送的資料sdin是16’hB000，可觀察到其MSB比lrck還要慢了一個sck的週期才被傳送。

****

上圖要傳送的資料sdin是16’h5FFF，可觀察到其LSB被傳送的時機在於lrck的riding或falling edge之後的第一個clk週期內。由以上simulation的波形圖可以驗證我寫的module所製造出來的lrck與sdin兩者都符合DAC CS4344的規定。

**5. Conclusion**

**內容: 可以寫下你的這個lab的想法、遇到的問題、解決方法、心得等等，請自由發揮。**

這題比較讓人感到有樂趣，尤其是在經過不斷的調整和測試之後，終究還是順利的製造出了符合DAC CS4344的規定的lrck與sdin的波形圖。這也是我第一次寫parallel-to-serial的converter，一開始我還有點不知道應該要如何下手，就很直接的寫了32個D Flip-flop，想說可以在一個lrck的週期內用一個五位元的計數器從31數到0，並拿計數器的第五的位元（即MSB）用來判斷要傳送left還是right的資訊。結果看了波型圖才發現這種做法的瓶頸在於，sck的相位剛好相反，要切換到下一個bit的資料的時機一不小心就會將rising和falling edge弄混，且修正之後的code有點複雜，難以用簡潔且讓人易懂的方法描述出該硬體的配置。所以在搜尋相關文獻之後，索性參考那份I2S bus specification文件所附的block-diagram，才改成了現在的做法。

**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 volumn of the sound by pressing button (Up) as increase and (Down) and decrease the volumn. Please also quantize the audio dynamic range as 16 levels and show the current sound level in the 7-segment display.**

**1. Specification**

**內容: 寫下你的電路中的inputs, outputs以及其bit widths，名稱必須跟你的verilog code中相同。**

module speaker(

output [7:0] cathode, // 七段顯示器

output audio\_mclk, // master clock

output audio\_lrck, // left-right clock

output audio\_sck, // serial clock

output audio\_sdin, // serial audio data input

input increase\_volume, // 按一下會增加音量

input decrease\_volume, // 按一下會減少音量

input play\_do, // 按下時會撥放Do

input play\_re, // 按下時會撥放Re

input play\_mi, // 按下時會撥放Mi

input clk, // clock from the crystal

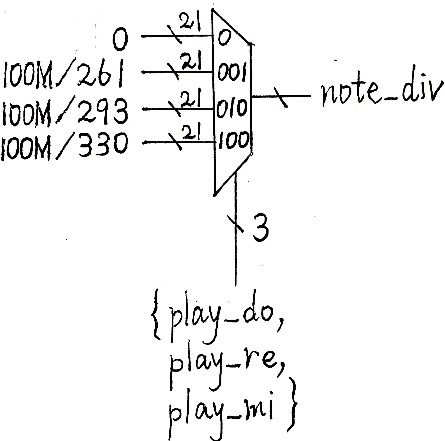
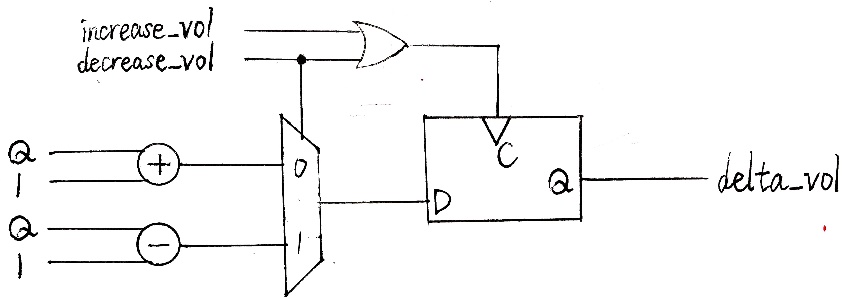
input rst\_n // active low reset

);

**2. Block Diagram**

**內容: 電路中的Block diagram(可以用手畫拍照或電腦繪圖)。**

跟上一題比較起來，這一題只要加上兩個新的模組就好。首先，因為要有改變音高的能力，所以使一個MUX來依照當下按下的按鈕來產生適當的除數，再把除數傳送到上一題的除頻器以便依照被按下的按鈕來產生對應的聲音資訊。由於還要有調整音高的能力，所以左和右的聲音資訊在傳送到DAC之前，要先跟音量的係數相乘。而音量係數由一個新的模組來產生。此外，還需要用到前幾次LAB中用來控制七段顯示器的模組。



**3. Finite state machine**

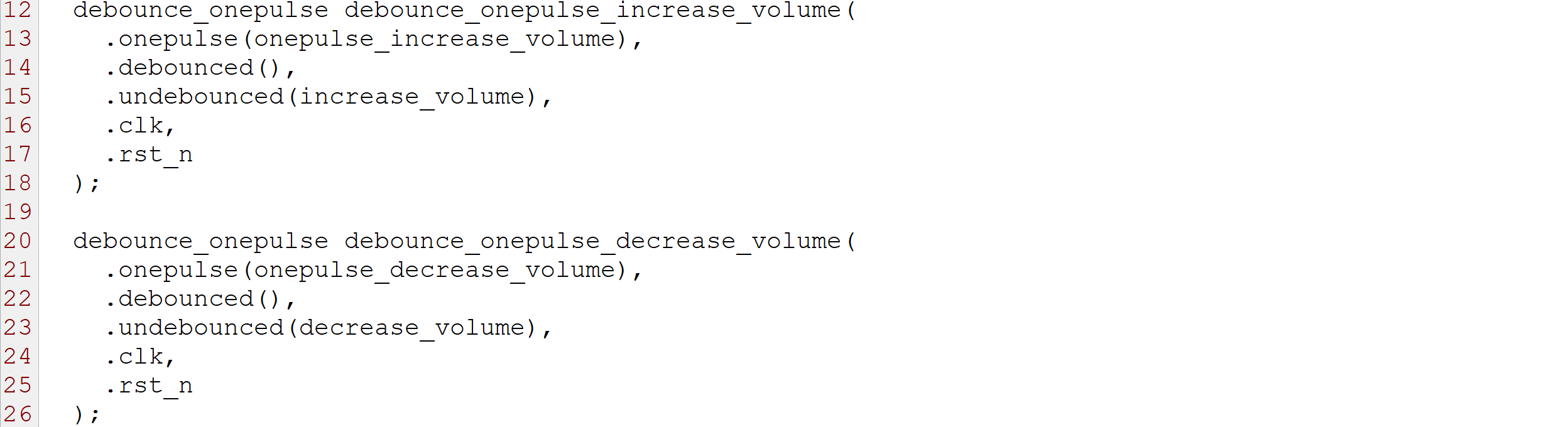
**內容: 電路中的Finite state machine，若無則寫無。**

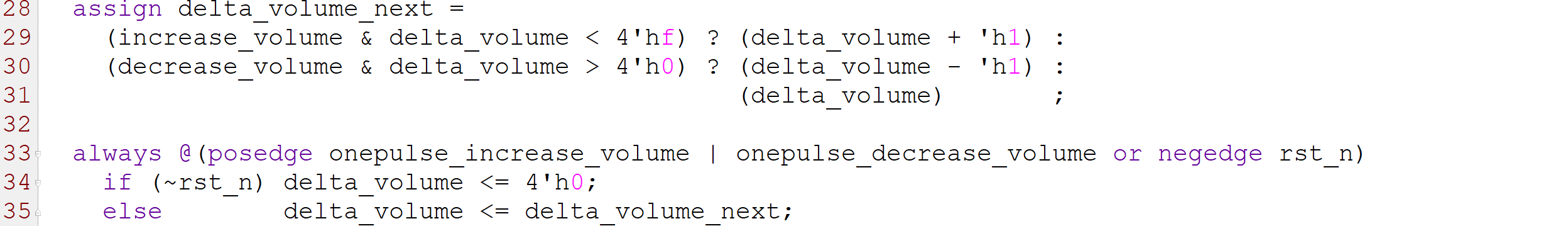
無

**4. Implement**

**內容: 請列出相關的logic function、詳細用文字解釋電路的運作方法、結果等等，可以貼code解釋或拍FPGA輔助解釋(但不能只貼code跟FPGA結果)。**

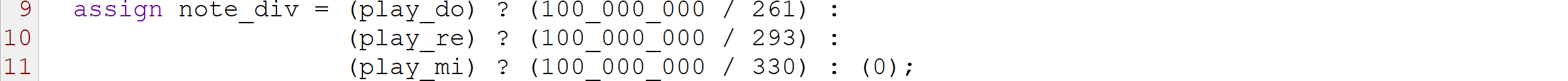
control\_volumn.v產生控制音量的係數delta\_volume，到時候在top可以直接跟note\_gen.v所產生的聲音資訊的震幅相乘，就可以改變音量大小。



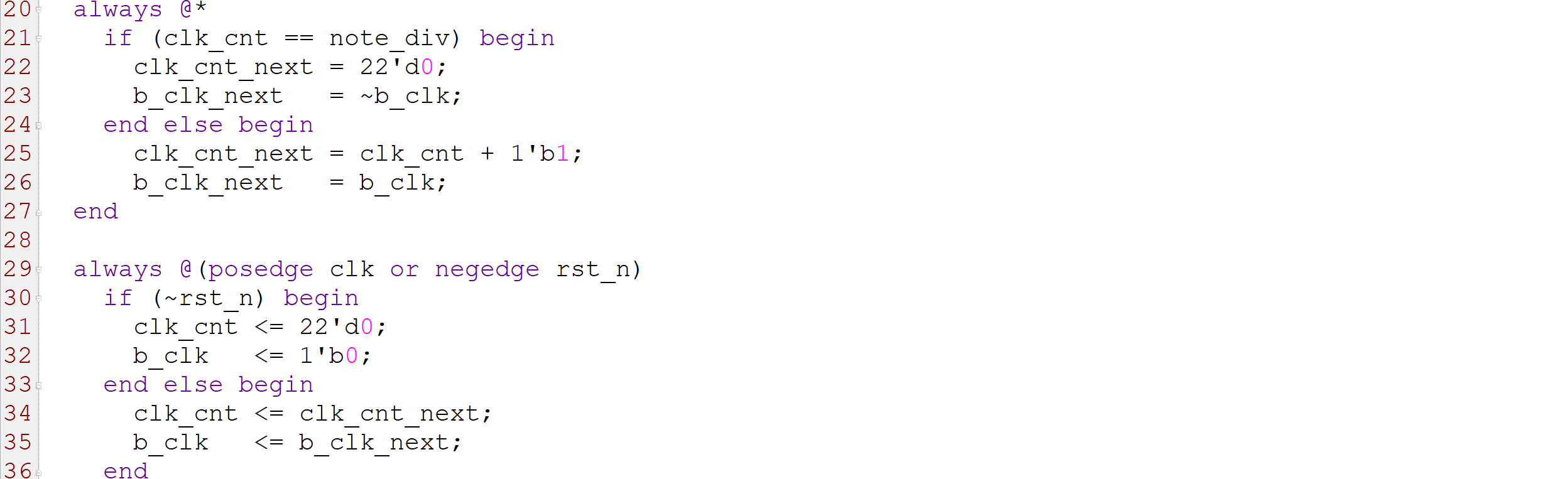
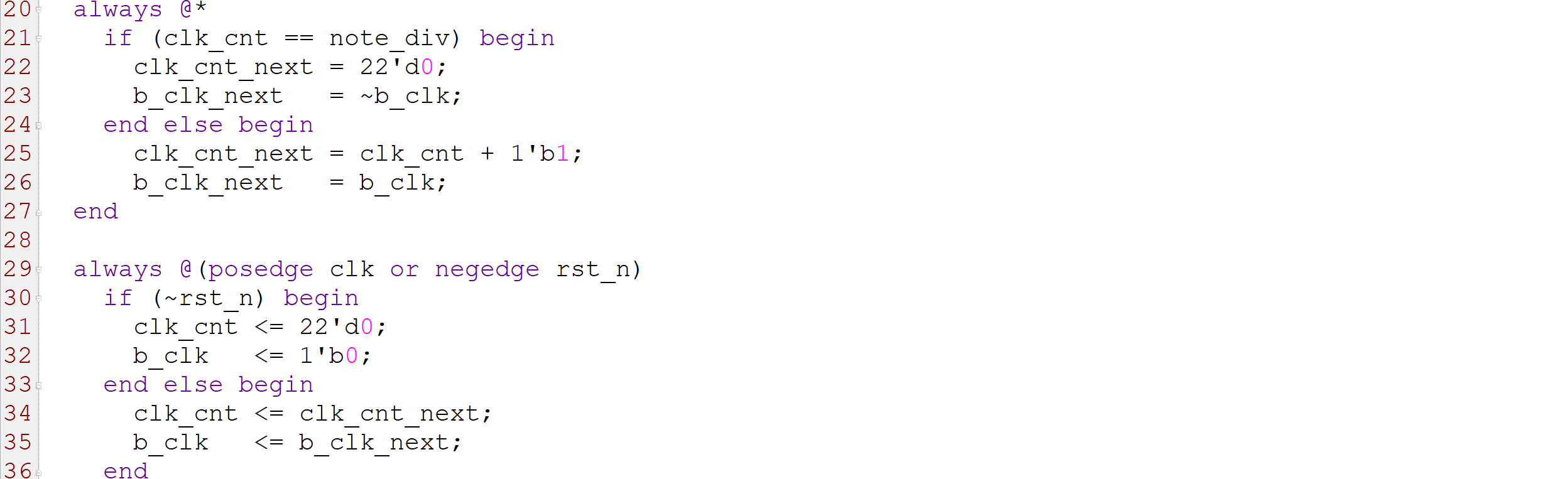
如上圖，所有除了時脈以外的input都要先debounce或onrpulse之後再使用。

如上圖，產生音量控制係數的方法很單純。如果增加音量的按鈕被按下，且當下的音量不是最大值，就加一；如果降低音量的按鈕被按下，且當下的音量不是最小值，就減一。

note\_div\_lookup.v產生對應Do、Re、Mi的頻率的除數，供note\_gen.v參照。

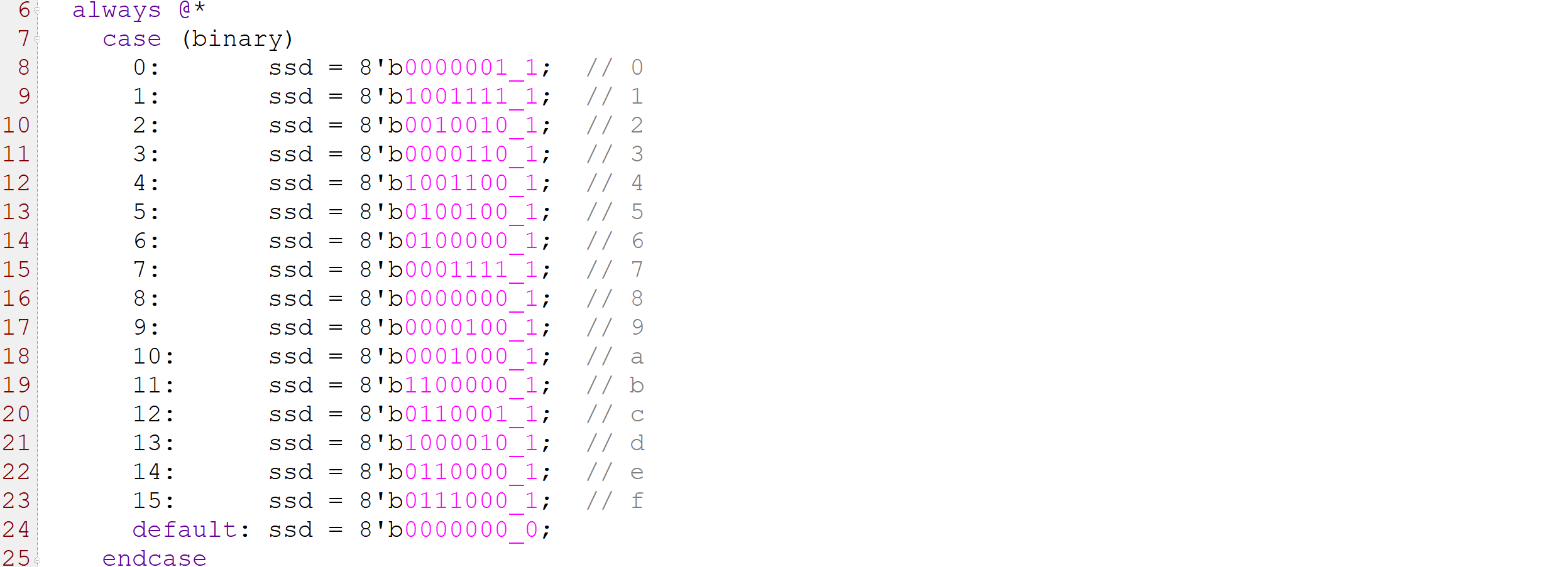
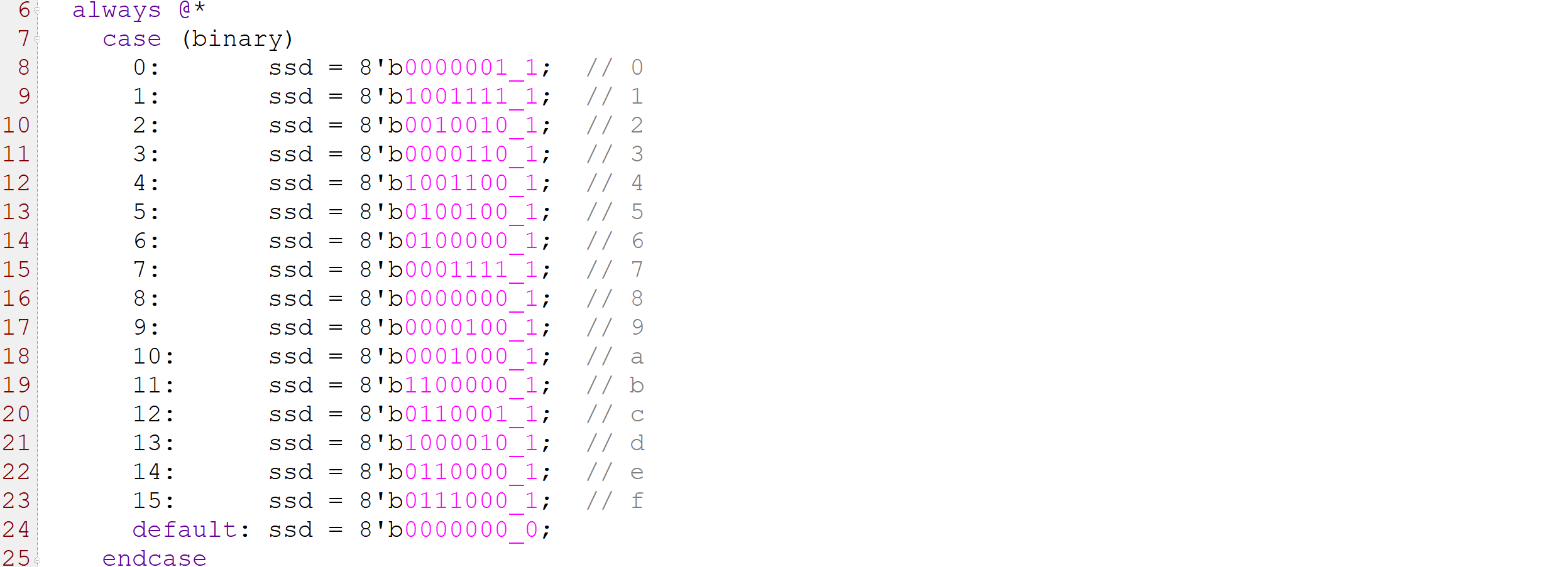


如下圖，我把note\_gen.v簡化，因為到時候在top要改變音量，所以其實不用像上一題一樣產生特定震幅的左和右聲音資訊。



如上圖，因為要產生的聲音是方波，所以只要保留能依照頻率除數產生特定頻率的b\_clk即可。

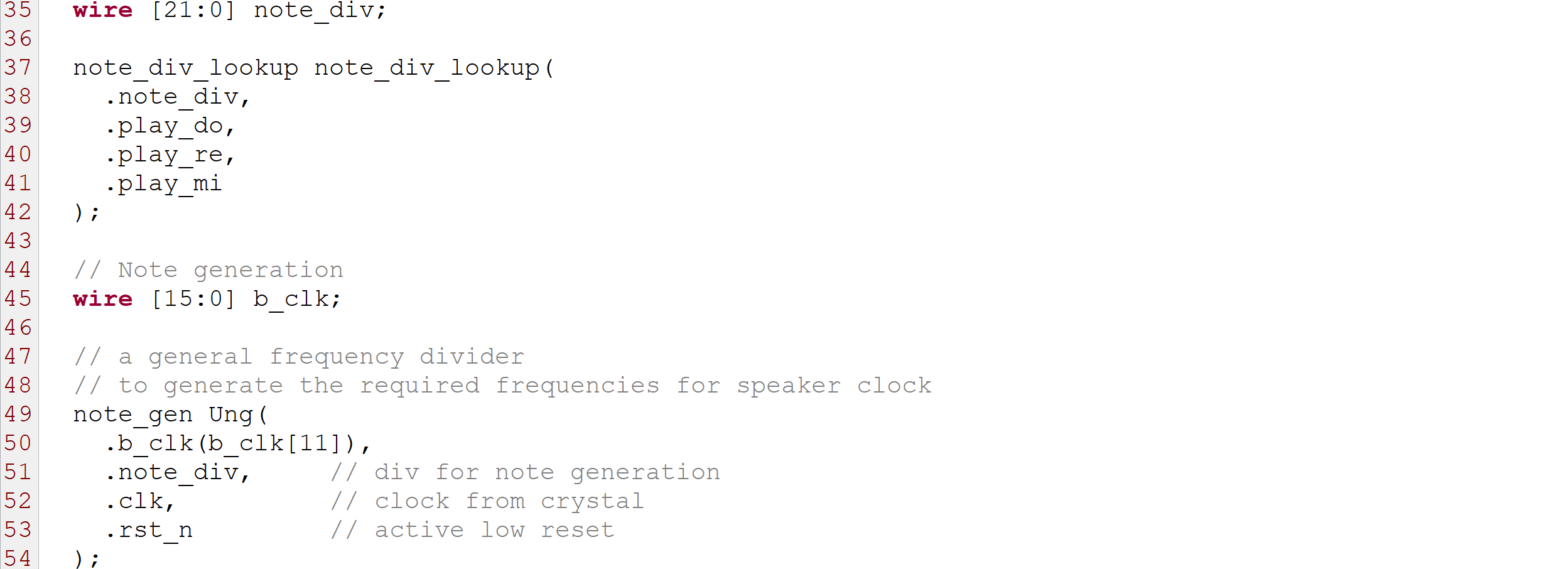
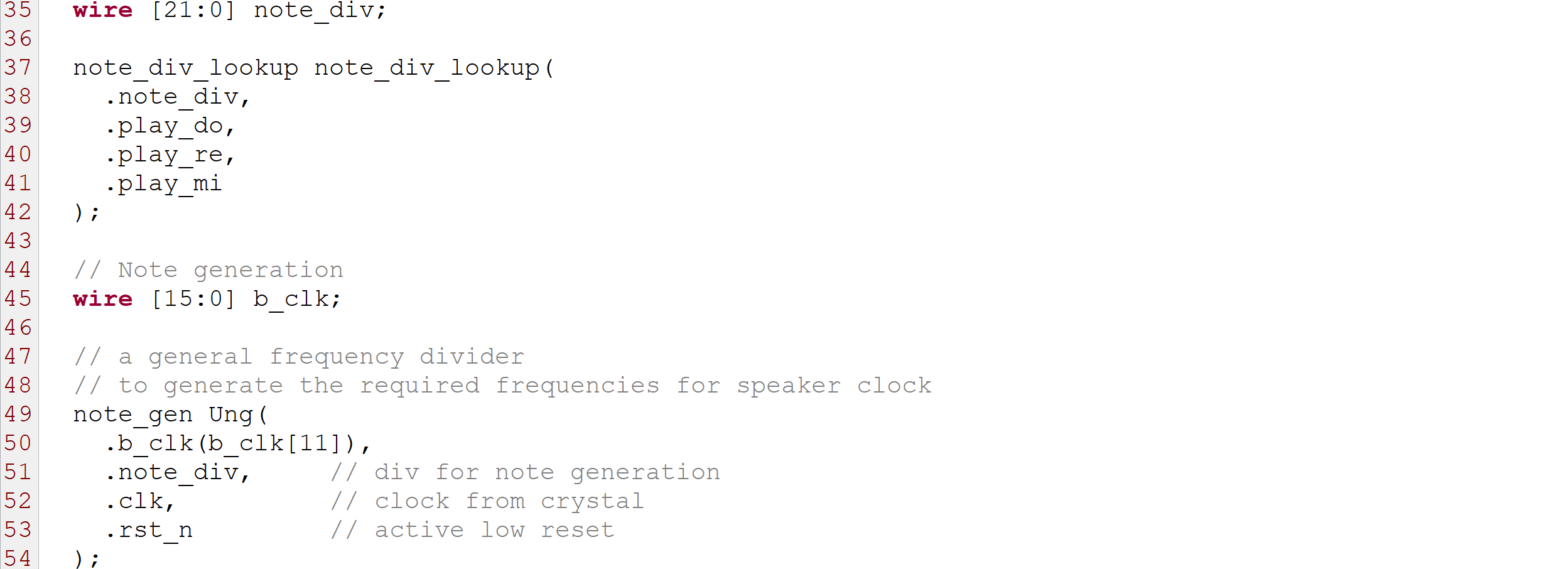
下圖是二進位轉七段顯示器樣式的解碼器，以顯示當下音量大小。



只要新增或修改上述的模組，其餘的模組直接沿用上一題的即可。接下來要建構top。



如上圖，先產生用來控制音量的係數delta\_volume，並把它轉換成七段顯示器的樣式，以便顯示音量大小。



如上，產生用來改變音高的除數note\_div，並連接到除頻器以產生方波b\_clk。



最後，把方波連接到用來控制DAC的模組即可。須注意b\_clk此時要跟音量係數相乘。

**5. Conclusion**

**內容: 可以寫下你的這個lab的想法、遇到的問題、解決方法、心得等等，請自由發揮。**

這題比較沒什麼問題，因為都是一些前幾次LAB就有寫過的東西，複製貼上再稍加修改就好。比較難的只有上一題，因為對DAC不熟，對它的運作原理也很陌生。