**Lab 7 110060027朱豐蔚**

1. **exp\_1 (audio-data parallel-to-serial signal generater)**

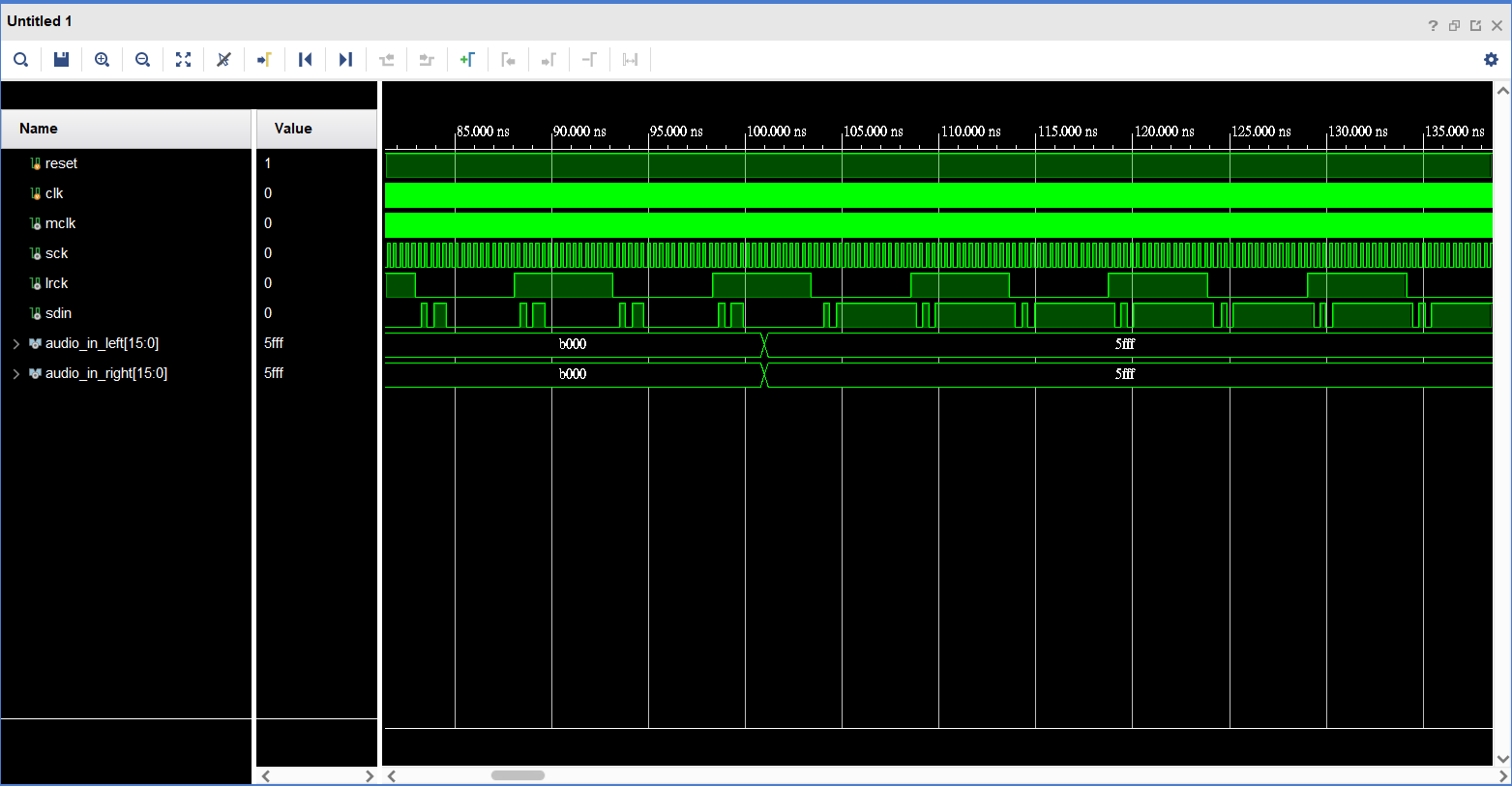
**Design Specification**

Input: clk, reset.

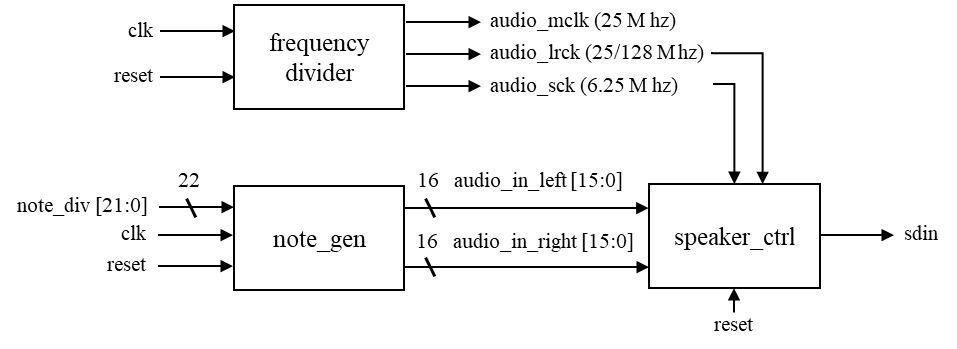
Output: audio\_mclk, audio\_lrck, audio\_sck, audio\_sdin,

[15:0] audio\_in\_left, [15:0] audio\_in\_right.

**Design Implementation**

 Simulation waveform:

如圖，當lrck為0時，會以sck作為clk，將parallel的audio\_in\_left[15:0]轉為serial的sdin訊號做為輸出，此時為左聲道；而當lrck為1時，會以sck作為clk，將parallel的audio\_in\_right[15:0]轉為serial的sdin訊號做為輸出，此時為右聲道。

 Design flow:

先使用除頻器對原先100M hz的clk訊號除頻，形成所求mclk、lrck、sck，再以lrck、sck作為parallel 轉 serial模組的大、小clk輸入，利用shift register的原理將parallel 的audio\_in\_left、audio\_in\_right訊號轉成 serial的sdin訊號，提供給Pmod I2S進行音訊的處理。

**Discussion:**

在exp\_1中，我使用的note\_div [21:0]訊號為22’d5000，而不是使用標準音高Do的數字，原因是為了讓我在觀察testbench時能夠更方便的看到lrck、sck、sdin之間的變化關係，才採用了頻率與sck不相差過多的22’d5000。

而對於audio\_in\_left、audio\_in\_right的量值，我在此次實驗中並沒有自行調整，而是直接使用講義上提供的數值16’hB000與16’h5FFF作為振福，關於振幅的細微調整，將在後續實驗中使用到。

1. **exp\_2 (speaker controller with volume control function)**

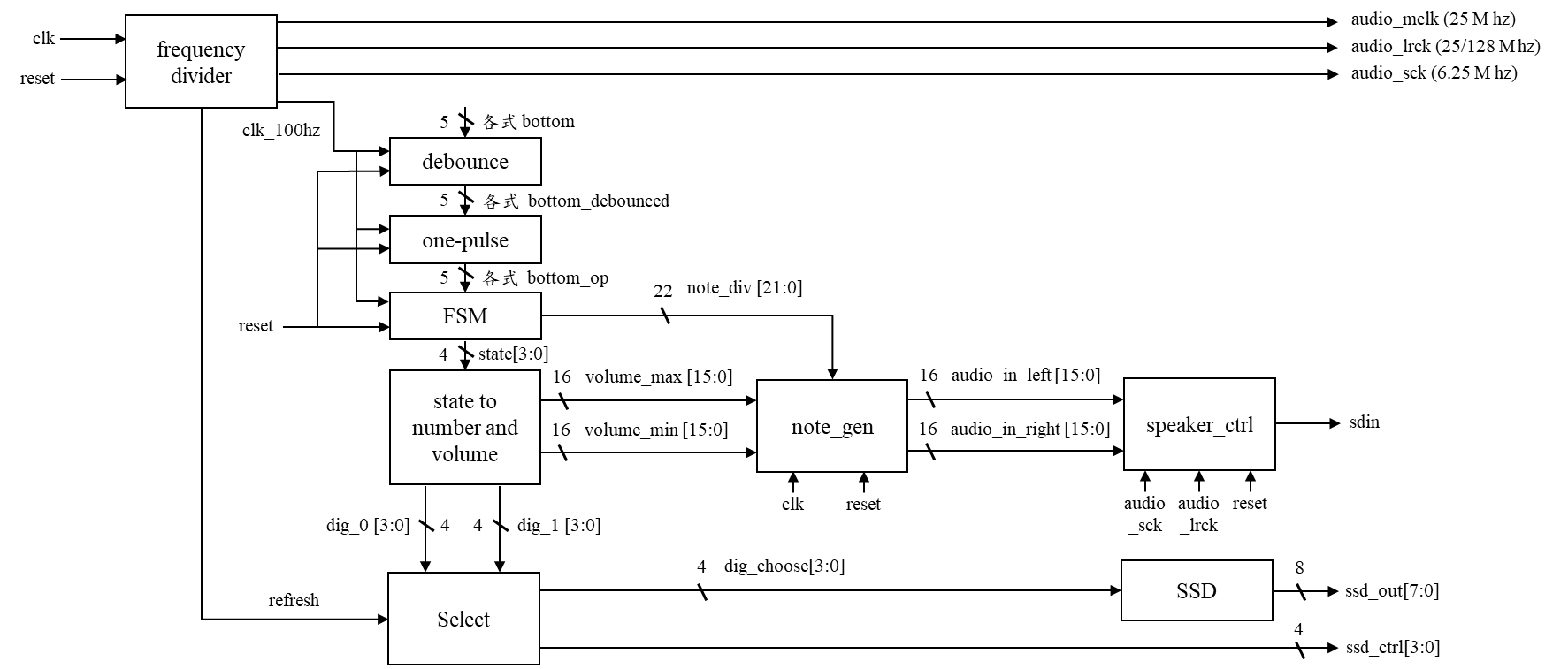
**Design Specification**

Input: clk, reset, botton\_L, botton\_R, botton\_M, botton\_U, botton\_D.

Output: audio\_mclk, audio\_lrck, audio\_sck, audio\_sdin,

[15:0] audio\_in\_left, [15:0] audio\_in\_right, [7:0] ssd\_out, [3:0] ssd\_ctrl.

**Design Implementation**

 Block diagram:

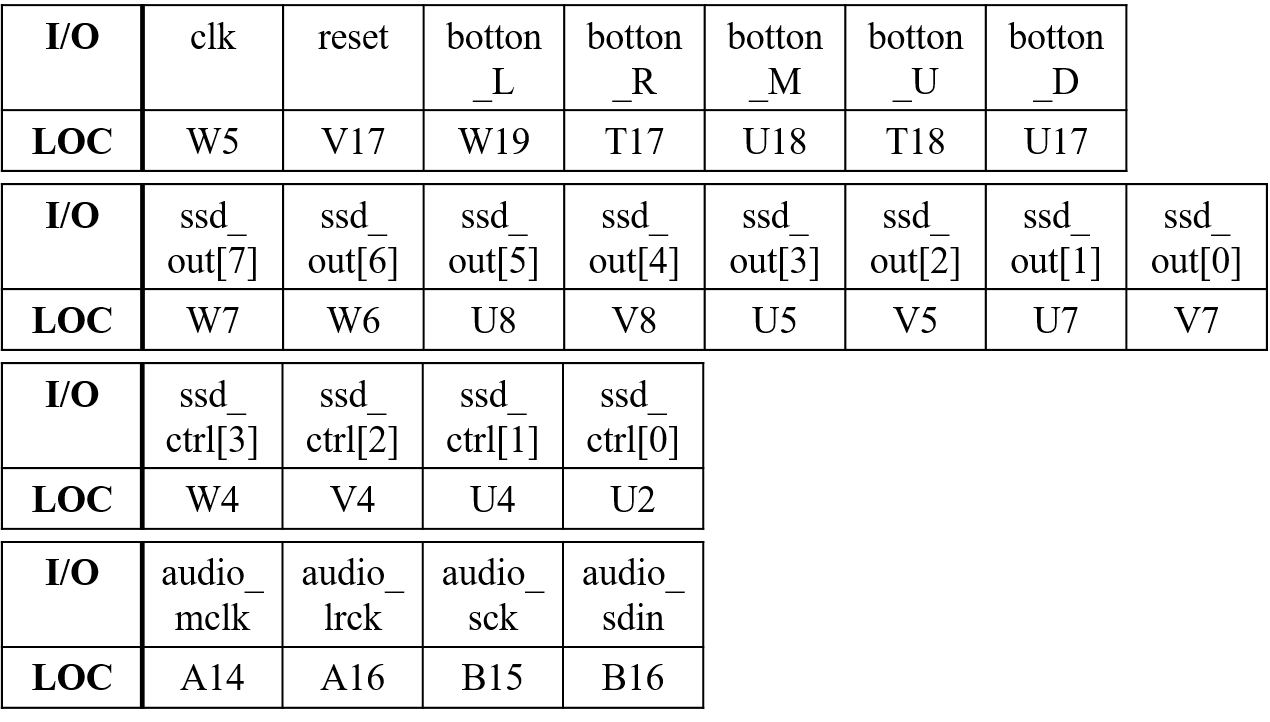
Design flow:

同exp\_1，先使用除頻器對原先100M hz的clk訊號除頻，形成所求mclk、lrck、sck，但須另外輸出clk\_100hz與refresh訊號，以分別提供給debounce、one-pulse、FSM與SSD的數字更新用。

在FSM當中分為16個state，去代表16個不同大小的聲音狀態。我使用case的寫法，在不同的state下，產生不同的volume\_max、volume\_min、dig\_1、dig\_0，前兩者提供給note\_gen去產生振幅不同的訊號，後兩者則是提供給select模組進行SSD的呈現。

在音高控制的方面，我則是簡單的偵測左、中、右按鈕是否有被按下，若屬於按下狀態，就使控制音高的note\_div訊號產生改變。並將音高、音量等訊號統整入note\_gen模組中。

最後再以lrck、sck作為parallel 轉 serial模組的大、小clk輸入，利用shift register的原理將parallel 的audio\_in\_left、audio\_in\_right訊號轉成 serial的sdin訊號，提供給Pmod I2S進行音訊的處理。

I/O pins assignment :

**Discussion:**

在exp\_2中，我比較沒有遇到太多的問題，最大的挑戰應該就是音量控制的方面，因為我們並不知道Pmod I2S將數字轉變為音量大小的關係式，有可能是向分貝定義的指數關係，也有可能是純粹的正比關係。

於是我在本次實驗中，考慮16’h0000與16’hFFFF作為上限與下限進行切割，先找出音量中線16’h8000，以向上向下振幅差不多的方式平均切割為16等分，但結果卻超乎我的想像，我原先以為數字距離中線較遠，應該會有較大的振幅，進而有較大的音量，但結果卻完全相反。

經過思考後，我認為唯一能解釋的原因是中線並非為16’h8000，而應該為16’h0000，原先在我的設想中，數字應該是unsinged的，但從結果上推論，若將數字視為singed的表示法，就能完全解釋結果中聲音遞增的方向，這點也蠻符合實際上使用的情況，第一位為1就代表低於中線，為0就代表高於中線，這樣的設計也能大大提升在音量設定上的便利性。

1. **exp\_3 (speaker controller with double tones function)**

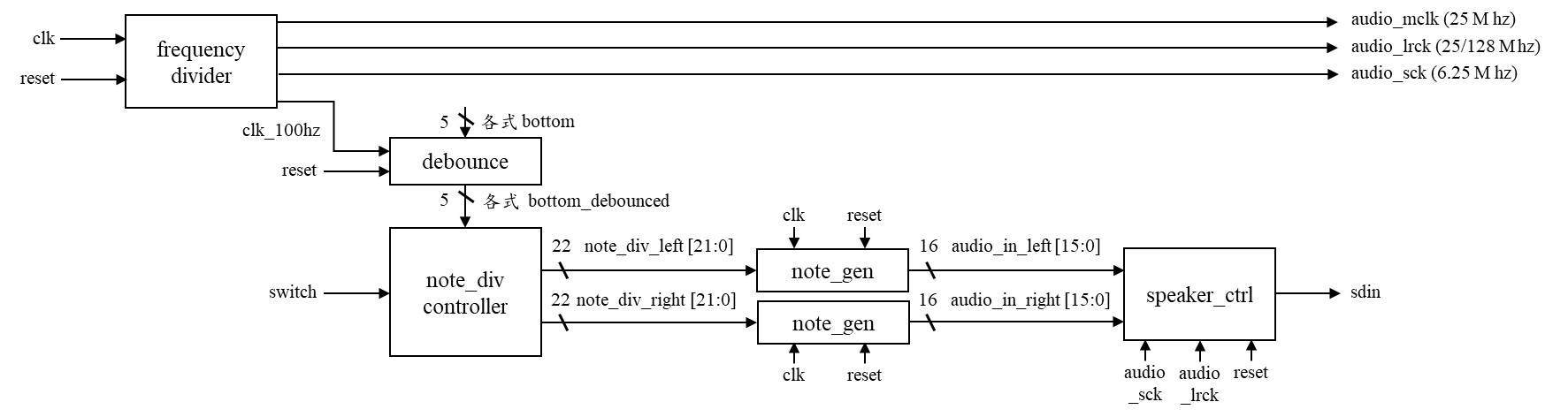
**Design Specification**

Input: clk, reset, switch, botton\_L, botton\_R, botton\_M, botton\_U, botton\_D.

Output: audio\_mclk, audio\_lrck, audio\_sck, audio\_sdin,

[15:0] audio\_in\_left, [15:0] audio\_in\_right.

**Design Implementation**

 Block diagram:

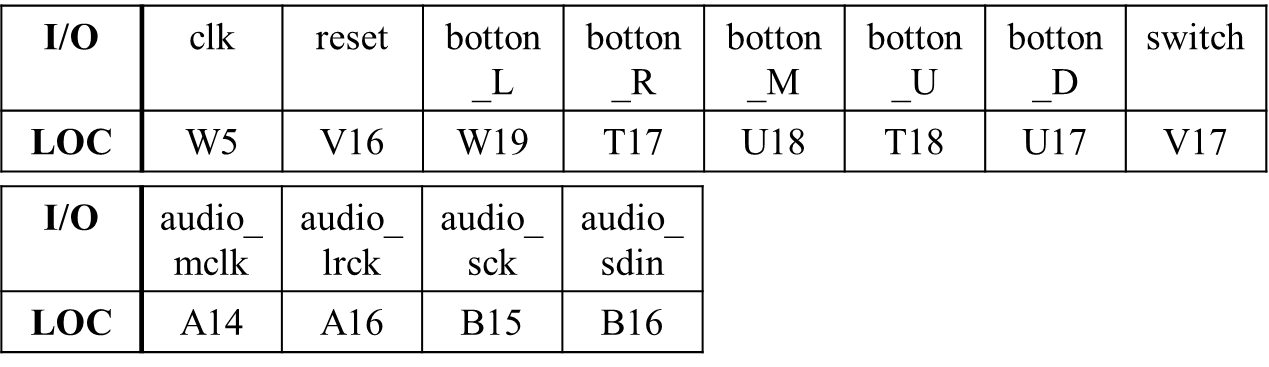
Design flow:

同exp\_1，先使用除頻器對原先100M hz的clk訊號除頻，形成所求mclk、lrck、sck，但須另外輸出clk\_100hz，以提供給debounce使用，此題未用到one-pulse，因只需判斷使否按下即可。

為了分別左聲道右聲道的音高，我將左、右的聲音產生模組進行分離，分別控制兩者，最後再由speaker\_ctrl進行整合。

在音高控制的方面，我則是簡單的偵測左、中、右、上、下按鈕是否有被按下，若屬於按下狀態，就使控制音高的note\_div訊號產生改變，而在switch = 1時，左右訊號要不同，在switch = 0時，左右訊號要相同，皆依題目所求去進行設定。

最後再以lrck、sck作為parallel 轉 serial模組的大、小clk輸入，利用shift register的原理將parallel 的audio\_in\_left、audio\_in\_right訊號轉成 serial的sdin訊號，提供給Pmod I2S進行音訊的處理。

I/O pins assignment :

**Discussion:**

在本次實驗，我並未遇到什麼問題，但有聽說同學有遇到左邊耳機是右聲道的聲音，右邊耳機是左聲道的聲音這樣類似聲因互換的問題。

我當下認為是lrck的順序錯誤，左聲道應該是0，右聲道應該是1，才不會聲因互換，也成功幫同學解決問題。這次經驗也讓我發現，在進行複雜度高且需要各項設備彼此連接的電路時，往往忘記加一個 ~ 或者是01搞錯，就會讓整個設計與原先的預期有很大的不同，尤其是在彼此協定寫得極為嚴謹的情況，更要小心這類失誤，才不會導致結果不如預期或失敗。

**======================================================================**

**Conclusion**

我認為這次Lab最大的收穫，是探索到了FPGA版的可延伸與發展性，透過本次實驗的Pmod I2S版，以及未來會用到的鍵盤、VGA，都可以大大提升FPGA版的功能豐富度，讓其能完成更完善且完整的功能。

但另一方面，由於各式附加裝置彼此的協定不同，更讓我對工程師感到敬佩，不管是各樣協定的訂定，還是協定的應用與發展，都讓我深深佩服工程師們的辛苦與創意。未來在協定的撰寫或應用上，一定還會遇到許多問題，這次Lab雖說並未用到雙向的溝通，但也在我心中留下一個底，以做足準備面對未來的挑戰。

**References**

音訊傳遞範例: (from 第七週上課講義)

音訊接腳代號查詢: (from 第七週上課講義)

note\_gen 模組寫法範例: (from 第七週上課講義)

speaker\_ctrl 模組寫法範例: (from 第七週上課講義)