# Lab 7

110060027 朱豐蔚

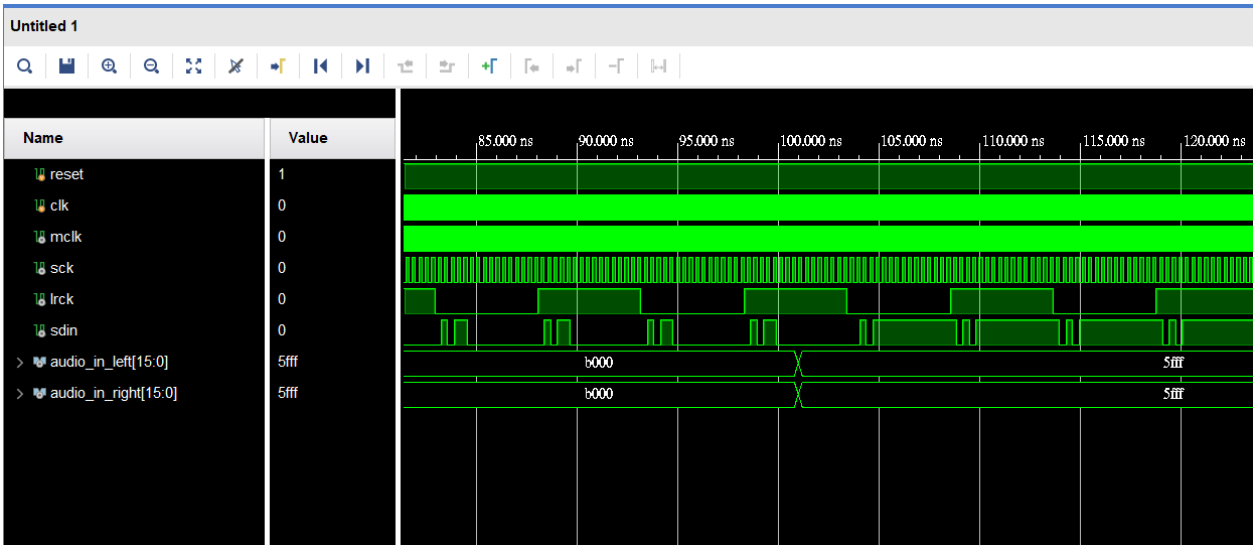
## I. exp_1 (audio-data parallel-to-serial signal generater)

### Design Specification

Input: clk, reset.
Output: audio_mclk, audio_lrck, audio_sck, audio_sdin,
[15:0] audio_in_left, [15:0] audio_in_right.

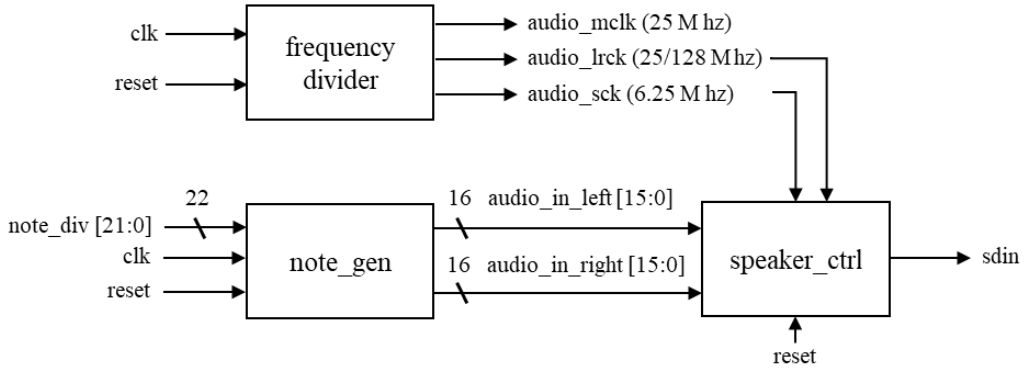
### Design Implementation

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。

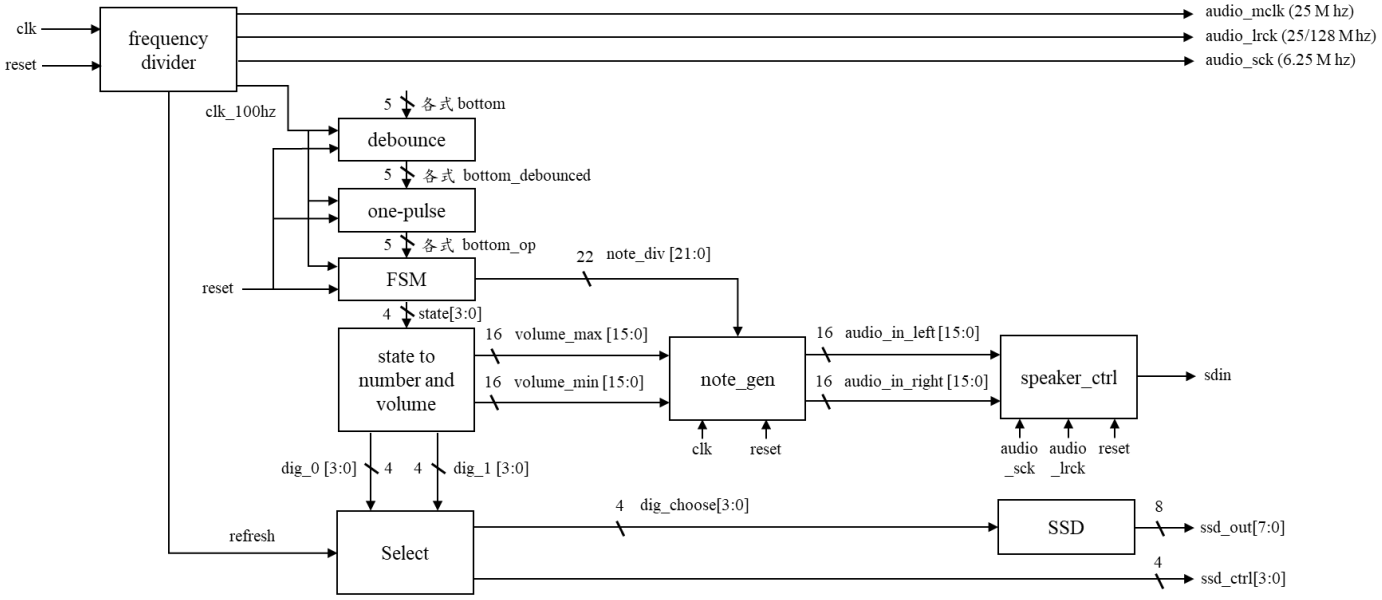
# II. exp_2 (speaker controller with volume control function)

## 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:

| **I/O** | clk | reset | botton_L | botton_R | botton_M | botton_U | botton_D |
|---|---|---|---|---|---|---|---|
| **LOC** | W5 | V17 | W19 | T17 | U18 | T18 | U17 |

| **I/O** | ssd_out[7] | ssd_out[6] | ssd_out[5] | ssd_out[4] | ssd_out[3] | ssd_out[2] | ssd_out[1] | ssd_out[0] |
|---|---|---|---|---|---|---|---|---|
| **LOC** | W7 | W6 | U8 | V8 | U5 | V5 | U7 | V7 |

| **I/O** | ssd_ctrl[3] | ssd_ctrl[2] | ssd_ctrl[1] | ssd_ctrl[0] |
|---|---|---|---|---|
| **LOC** | W4 | V4 | U4 | U2 |

| **I/O** | audio_mclk | audio_lrck | audio_sck | audio_sdin |
|---|---|---|---|---|
| **LOC** | A14 | A16 | B15 | B16 |

## 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。

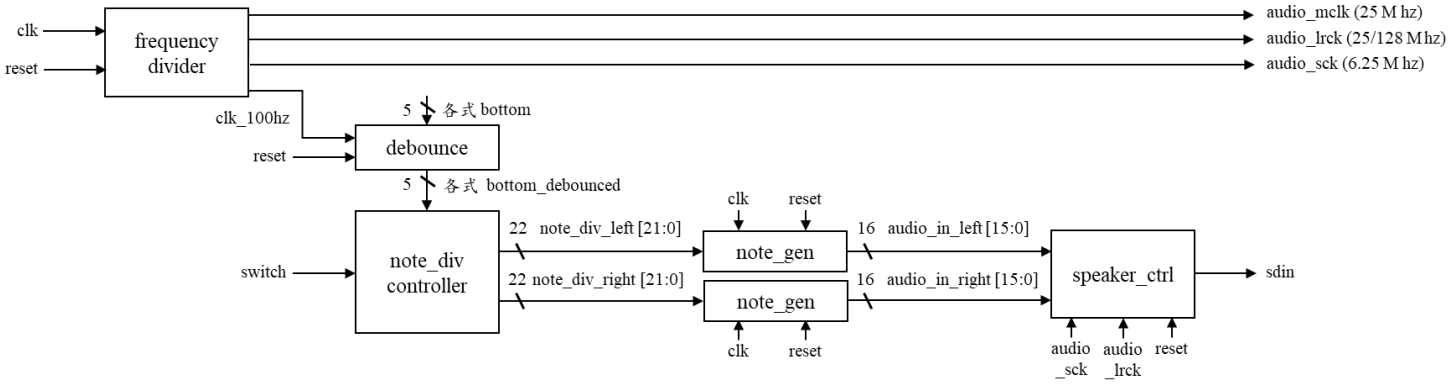
## III. exp_3 (speaker controller with double tones function)

## 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:

| **I/O** | clk | reset | botton_L | botton_R | botton_M | botton_U | botton_D | switch |
|---|---|---|---|---|---|---|---|---|
| **LOC** | W5 | V16 | W19 | T17 | U18 | T18 | U17 | V17 |

| **I/O** | audio_mclk | audio_lrck | audio_sck | audio_sdin |
|---|---|---|---|---|
| **LOC** | A14 | A16 | B15 | B16 |

## 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)