# 105061254 林士平 邏輯設計實驗報告 Lab8

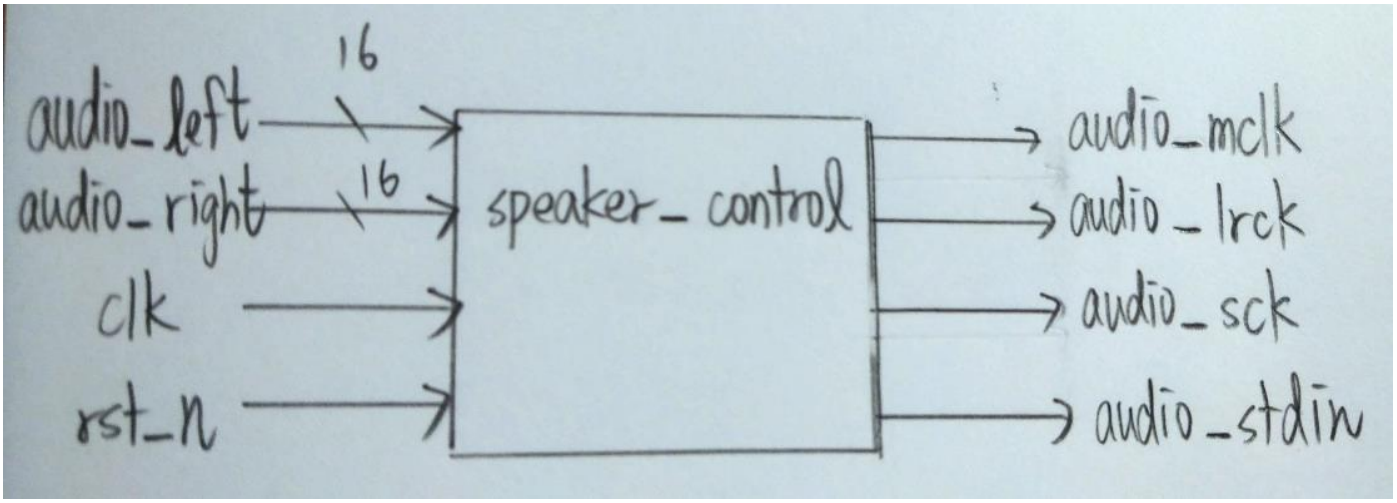
**1.**

(1) Design specification :

A. Inputs and outputs(表一)：

| | |
|---|---|
| Inputs | audio_left[15:0], audio_right[15:0], clk, rst_n |
| Outputs | audio_mclk, audio_lrck, audio_sck, audio_stdin |
| ↑表一：Inputs and outputs of 1 | |

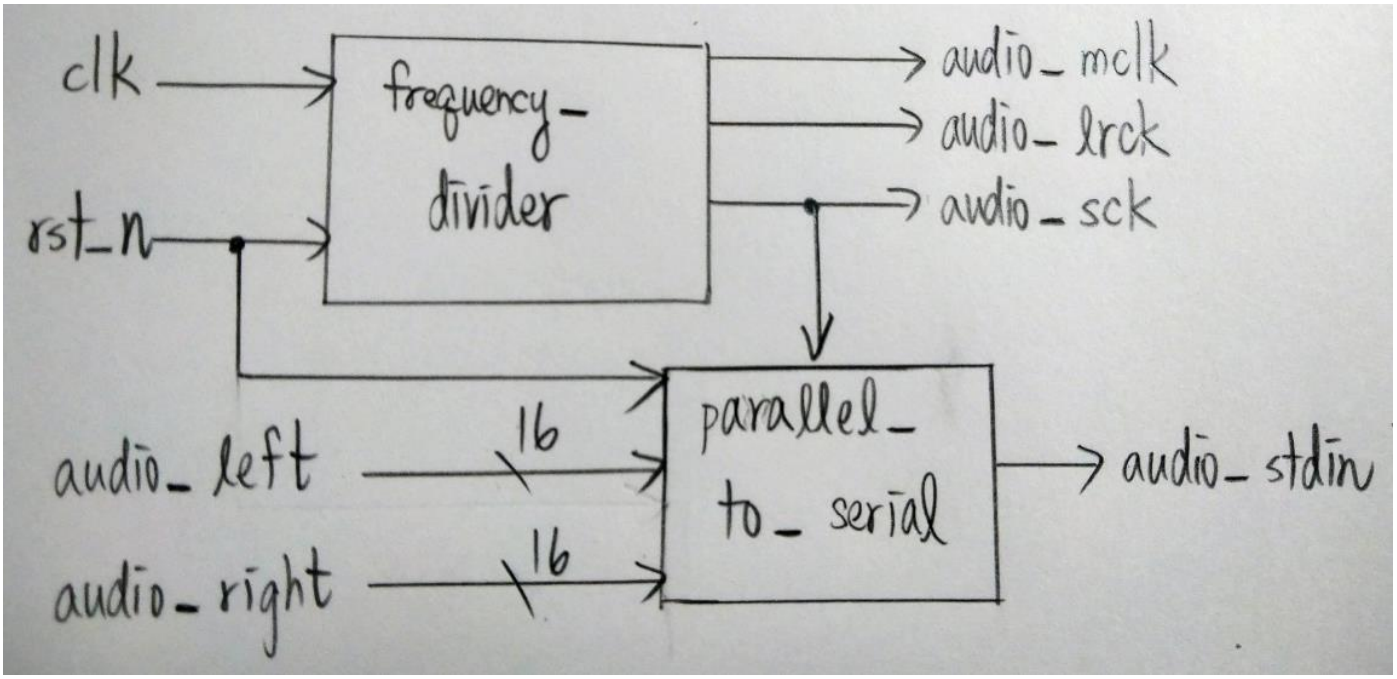
B. Block diagram(function table)(圖一)：



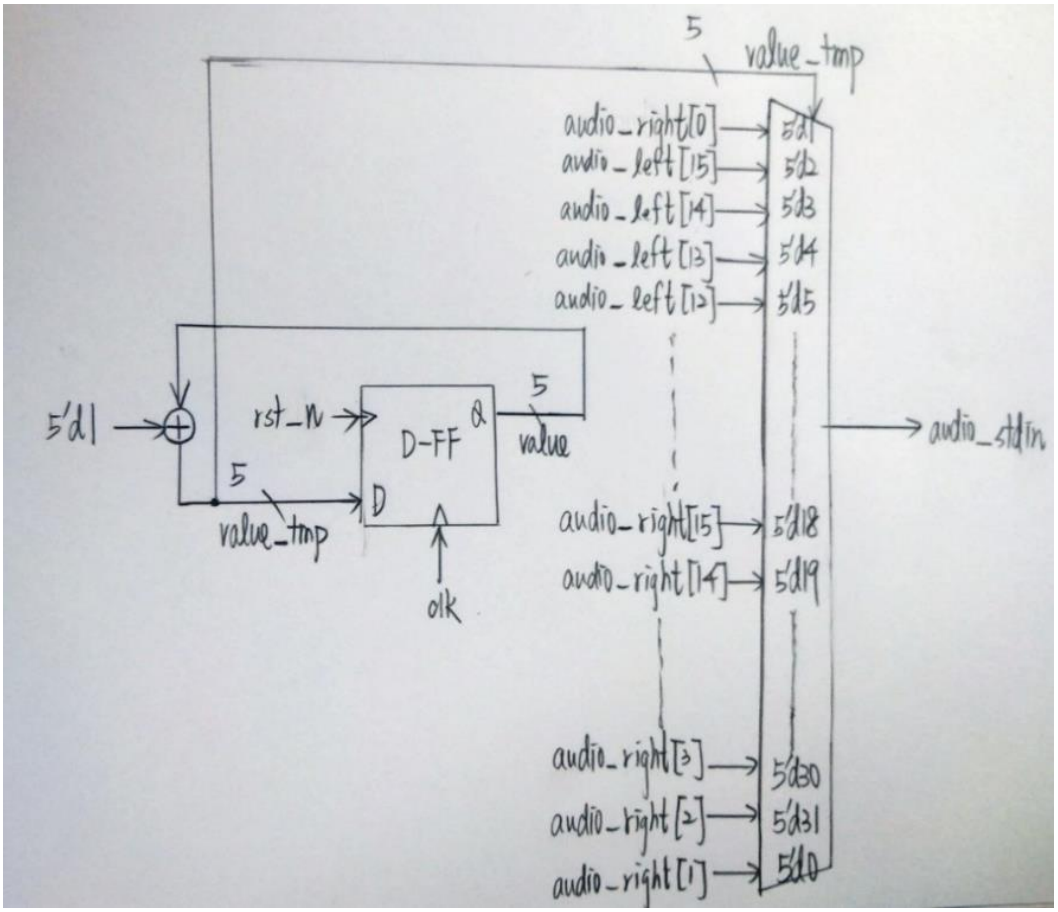

↑圖一：The block diagram of 1

(2) Design implementation :

A. Logic diagram(function table)(圖二)：




↑圖二：logic diagram of 1




↑圖三：logic diagram of parallel_to_serial module

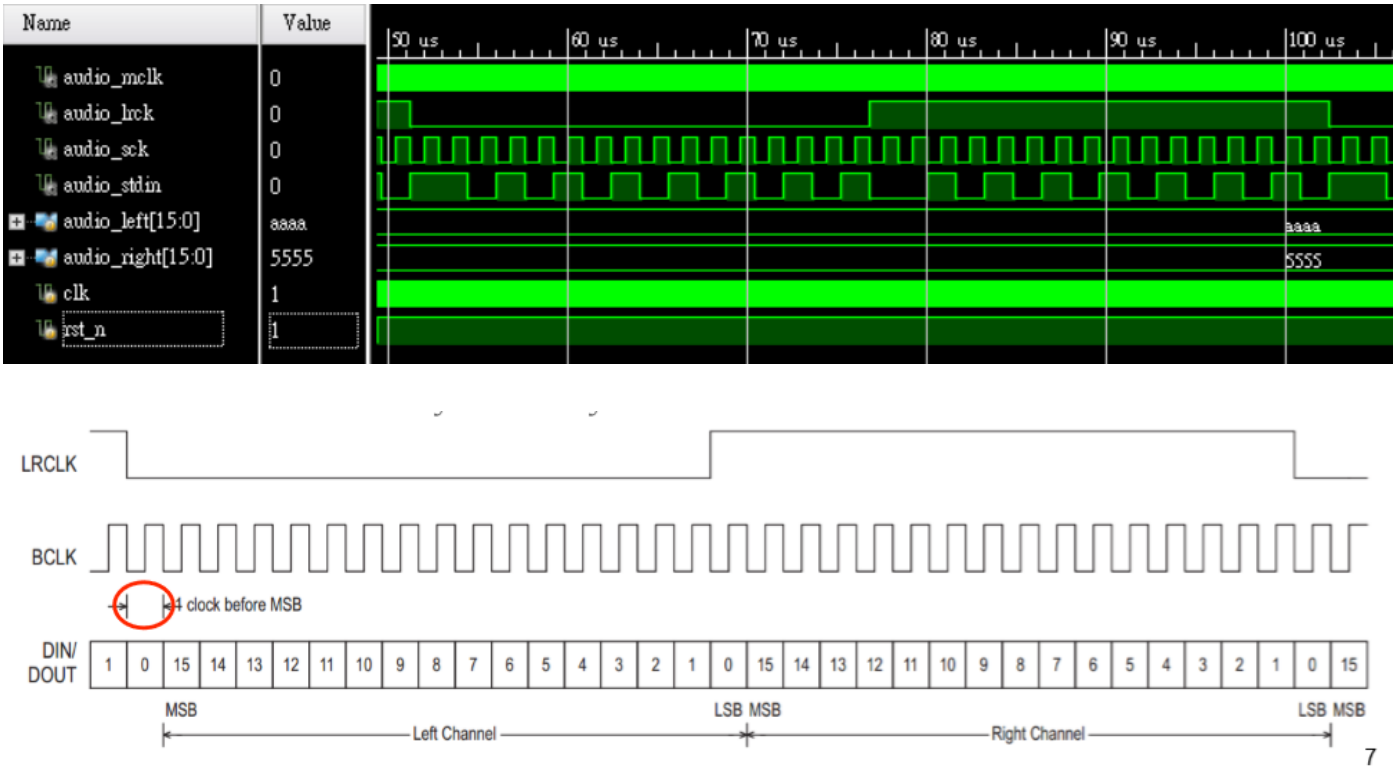
B. 功能與做法說明：

本題的第一小題為製作一個 frequency divider 來產生 audio_mclk(25MHz = 100MHz / 4)、audio_lrck(25MHz / 128 = 100MHz / 512)和 audio_sck(25MHz / 4 = 100MHz / 16)三種頻率的 clock，這個部分在 Lab3 已經做過了，簡單來說就是用 counter 數，數到想要除以的數的一半的時候去 toggle output 訊號，就能得到頻率為 100MHz / (想要除以的數) 的 output clock。

第二小題為製作一個 parallel – to – serial 的 processor 來產生 audio_stdin(speaker control signals)，圖三即為本小題的設計結果，利用一個可以數到 32 的 counter，每數到一個數分別輸出相對應的訊號(利用多功器)，便能將原本 parallel 的訊號轉為 serial 的。值得注意的是，由講義第七頁的圖，應該要先送左聲道，再送右聲道，而且從第 15 bit 開始送，並延遲一個 clk(見下頁圖四)，而且必須要用 negative edge trigger 的 clk(其他 module 的 clk 均為 positive edge trigger)。

再來將兩小題的 module 連接在一起形成 speaker control。需要注意的是，parallel – to – serial processor 的 clk 和 audio_sck 相同(25MHz / 4 = 100MHz / 16)，如此可以避免輸入的訊號和輸出的訊號塞在一起，觀察講義第七頁的圖也可以得到相同結果。

最後模擬出來的結果和講義第七頁圖的比較如下頁圖四所示，可以發現兩圖結果相同。




↑圖四：simulation result(上圖，其中 audio_left = 16'b1010101010101010、audio_right = 16'b0101010101010101)、講義第七頁圖(下圖)

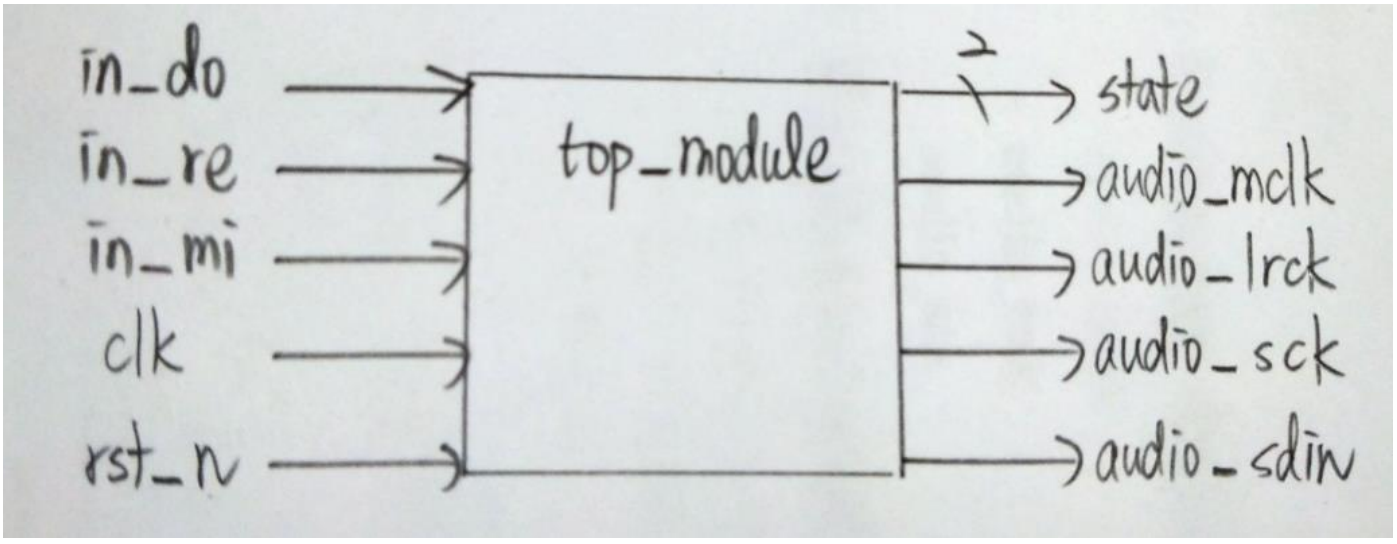
**2.1(method 1)**

(1) Design specification :

A. Inputs and outputs(表二)：

| | |
|---|---|
| Inputs | in_do, in_re, in_mi, clk, rst_n |
| Outputs | state[1:0], audio_mclk, audio_lrck, audio_sck, audio_sdin |
| ↑表二：Inputs and outputs of 2.1(method 1) | |

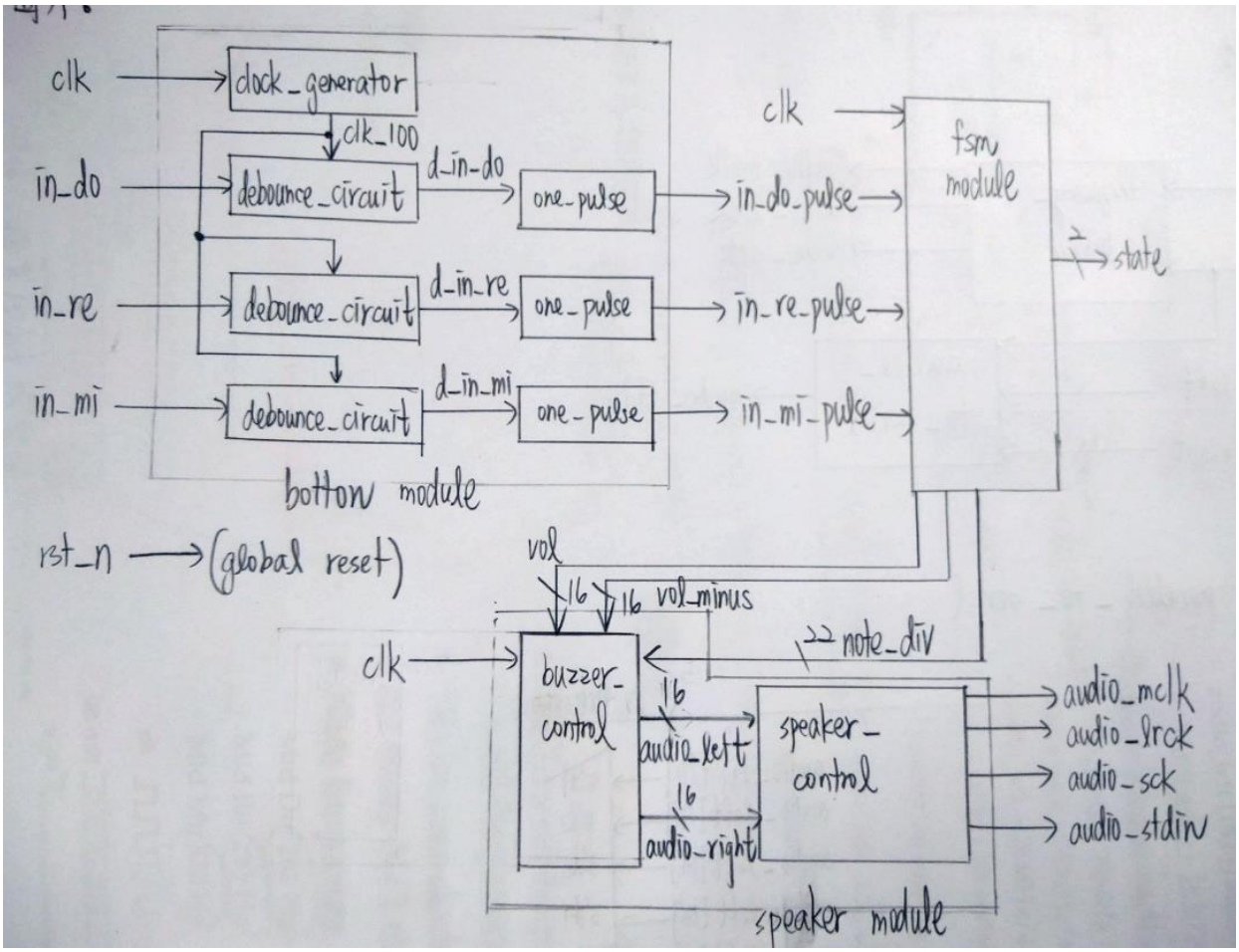
B. Block diagram(function table)(圖五)：




↑圖五：The block diagram of 2.1(method 1)

(2) Design implementation :

A. Logic diagram(function table)(圖六)：




↑圖六：logic diagram of 2.1(method 1)

B. I/O pin assignment(表三)：

| **I/O** | in_do | in_re | in_mi | clk | rst_n | state[0] | state[1] |
|---|---|---|---|---|---|---|---|
| **LOC** | W19 | U18 | T17 | W5 | V17 | U16 | E19 |
| **I/O** | audio_mclk | audio_lrck | audio_sck | audio_stdin | | | |
| **LOC** | A14 | A16 | B15 | B16 | | | |
| ↑表三：I/O pin assignment of 2.1(method 1) | | | | | | | |

C.功能與做法說明：

本題為製作一個 speaker 可以發出 Do, Re, Mi 三種聲調的聲音，每個聲調對應到一個按鈕，只要按下該按鈕便能產生那個聲調的聲音。

首先是 botton 模組，處理按鈕的 debounce 和 one_pulse，接著將處理完得到的訊號(in_do_pulse、in_re_pulse、in_mi_pulse)送入 finite state machine(fsm)模組。

fsm 模組總共有四個 state：reset state 為每次 reset 完後會進入(把開關扳下)，不會發出聲音；Do state、Re state 和 Mi state 則是分別發出 Do、Re、Mi 三種聲調的聲音。四個 state 有各自對應的 state output，state output 有三個：vol[15:0]和 vol_minus[15:0]和音量有關，vol 為聲音的正振幅、vol_minus 為聲音的負振幅，數值用 2's complement 表示，最後聲音的振幅大小為 vol – vol_minus，決定聲音的音量，另外 note_div 則是決定聲音的音高。最後送入 speaker 模組來產生相對應的聲音。

speaker 模組由 buzzer_control 和 speaker_control 組成，buzzer_control 依據收到的 note_div 除頻，結合收到的 vol[15:0]和 vol_minus[15:0]產生 audio_left[15:0]和 audio_right[15:0]，各自代表左聲道和右聲道的聲音訊號，將他們送入 speaker_control 模組，speaker_control 所做的事情如第一題所述(圖二)。

本題按下按鈕後便會一直發出聲音，例如：按下 Do 按鈕便會一直發出 Do，再來按下 Re 按鈕便會由發出 Do 改為發出 Re，以此類推，要讓聲音停下來只能扳下開關。所以我做了 method 2，在 method 2 中，只要放開按鈕聲音就會停止。

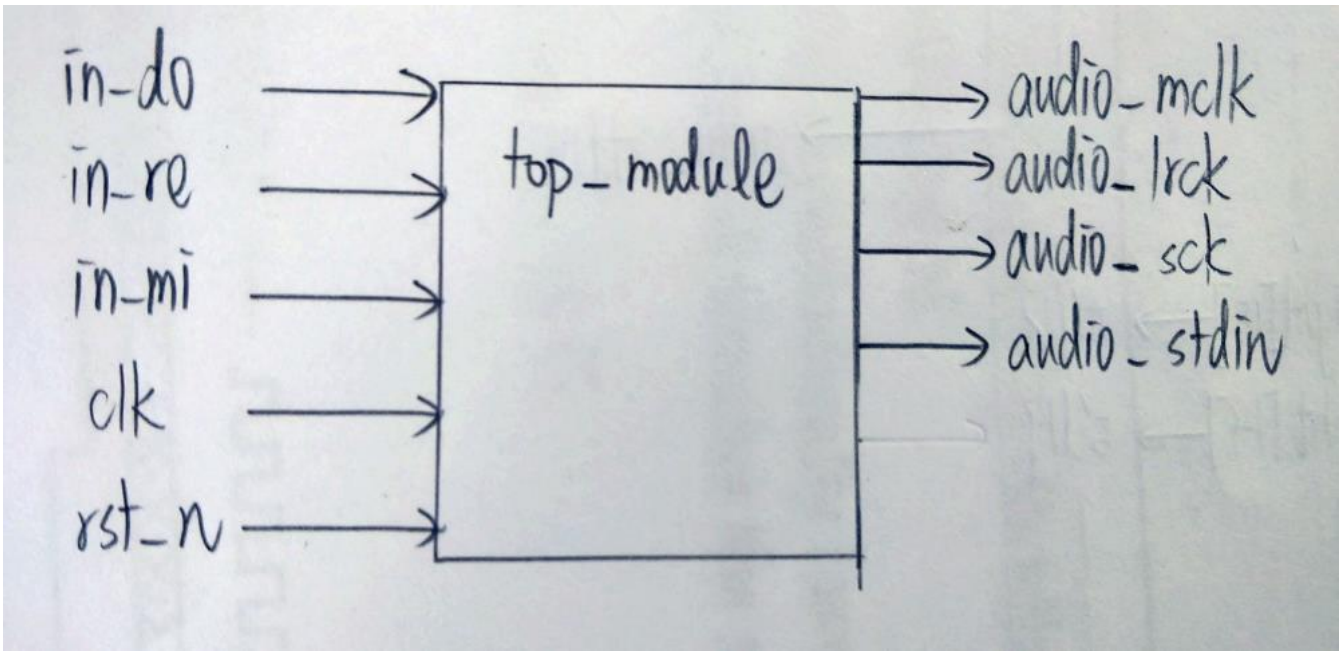
**2.1(method 2)**

(1) Design specification :

A. Inputs and outputs(表四)：

| | |
|---|---|
| Inputs | in_do, in_re, in_mi, clk, rst_n |
| Outputs | audio_mclk, audio_lrck, audio_sck, audio_sdin |
| ↑表二：Inputs and outputs of 2.1(method 2) | |

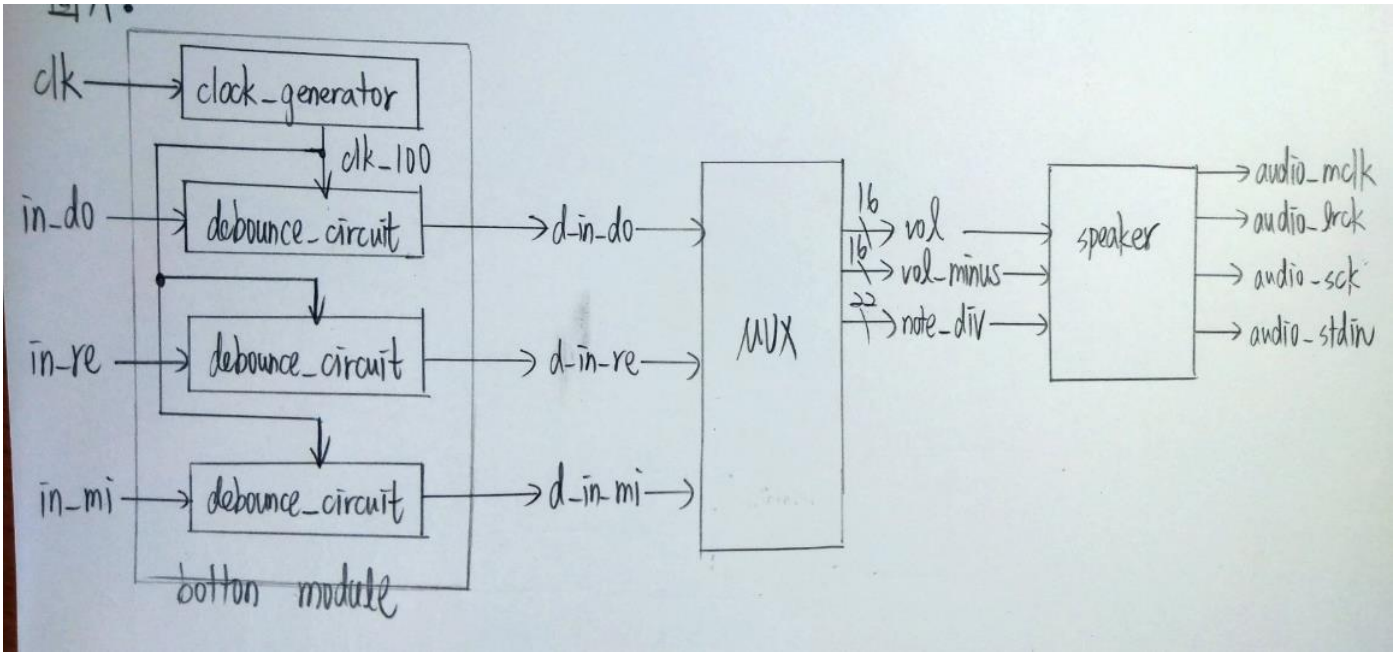
B. Block diagram(function table)(圖七)：




↑圖七：The block diagram of 2.1(method 2)

(2) Design implementation :

A. Logic diagram(function table)(圖八)：




↑圖八：logic diagram of 2.1(method 2)

B. I/O pin assignment(表五)：

| **I/O** | in_do | in_re | in_mi | clk | rst_n | audio_mclk | audio_lrck |
|---|---|---|---|---|---|---|---|
| **LOC** | W19 | U18 | T17 | W5 | V17 | A14 | A16 |
| **I/O** | audio_sck | audio_stdin | | | | | |
| **LOC** | B15 | B16 | | | | | |
| ↑表五：I/O pin assignment of 2.1(method 2) | | | | | | | |

C.功能與做法說明：

method2 和 method1 的不同在於將 finite state machine 以多功器代替，也因此 botton 模組不須要 one_pulse。多功器做判斷：當按下 do 按鈕發出 do 聲音、按下 re 按鈕發出 re 聲音、按下 mi 按鈕發出 mi，都不按的時候就不發出聲音。speaker 模組的內容和圖六畫的完全相同，功能如 2.1(method1)功能與做法說明所述。

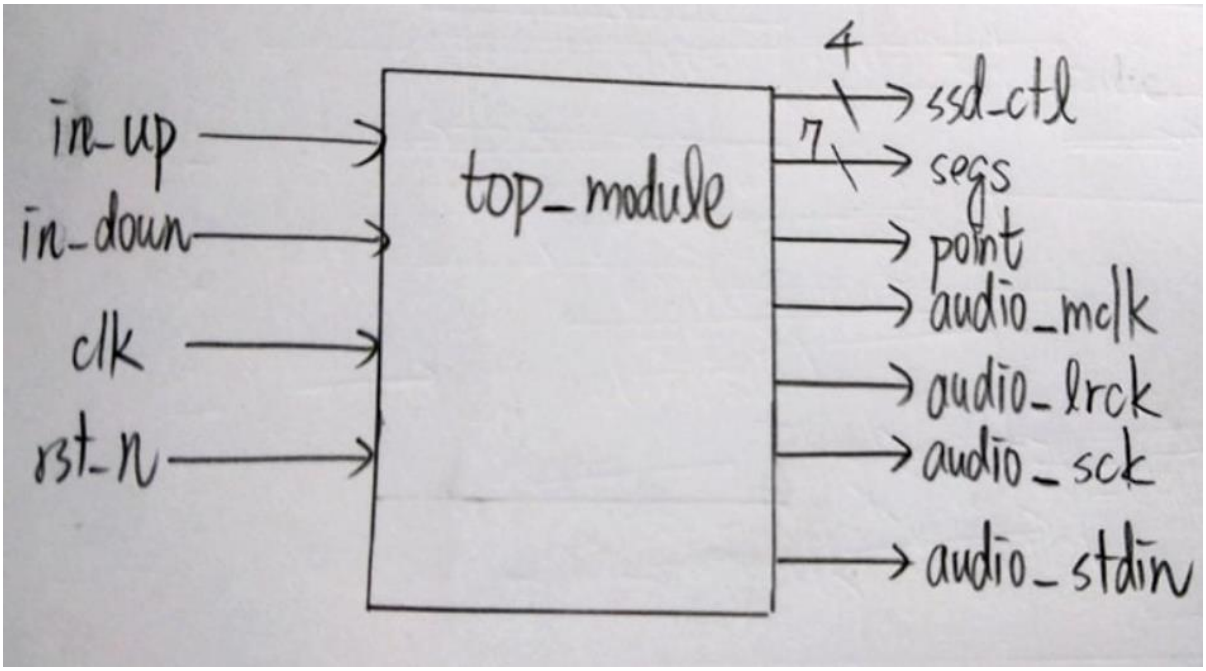
**2.2**

(1) Design specification :

A. Inputs and outputs(表六)：

| Inputs | clk, rst_n, in_up, in_down |
|---|---|
| Outputs | ssd_ctl[3:0], segs[6:0], point, audio_mclk, audio_lrck, audio_sck, audio_sdin |
| ↑表六：Inputs and outputs of 2.2 | |

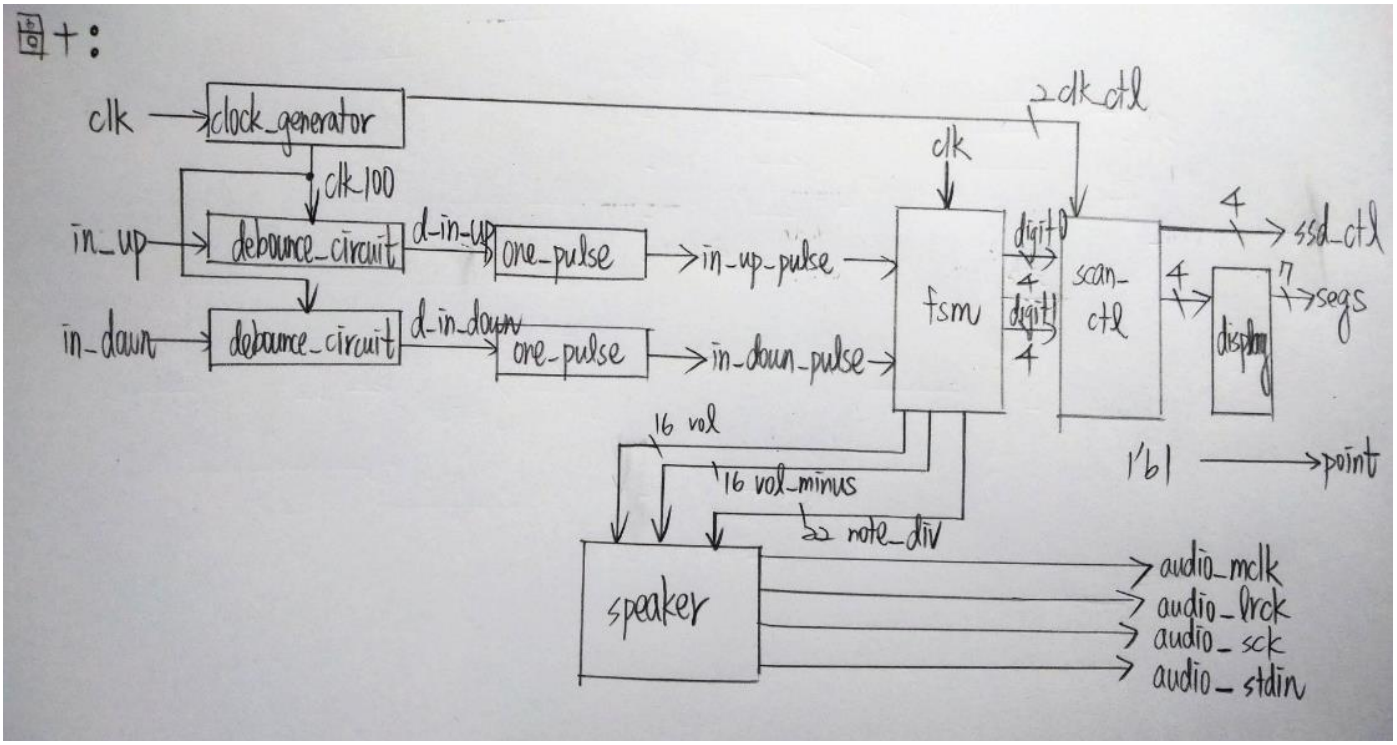
B. Block diagram(function table)(圖九)：




↑圖九：The block diagram of 2.2

(2) Design implementation :

A. Logic diagram(function table)(圖十)：




↑圖十：logic diagram of 2.2

B. I/O pin assignment(表七)：

| **I/O** | in_up | in_down | clk | rst_n | point | audio_mclk | audio_lrck |
|---|---|---|---|---|---|---|---|
| **LOC** | T18 | U17 | W5 | V17 | V7 | A14 | A16 |
| **I/O** | audio_sck | audio_stdin | ssd_ctl[3] | ssd_ctl[2] | ssd_ctl[1] | ssd_ctl[0] | segs[6] |
| **LOC** | B15 | B16 | W4 | V4 | U4 | U2 | W7 |
| **I/O** | segs[5] | segs[4] | segs[3] | segs[2] | segs[1] | segs[0] | |
| **LOC** | W6 | U8 | V8 | U5 | V5 | U7 | |
| ↑表七：I/O pin assignment of 2.2 | | | | | | | |

C.功能說明：

本題為設計一個可以改變音量的 speaker，要將音量分為 16 個 level，並且顯示現在的 level 在七段顯示器上。

botton 模組將 input 的按鈕訊號做 debounce 和 one_pulse，送入 finite state machine 做為 state transition input。

fsm 總共有 17 個 state，分別表示音量的 level，state 5'd0 為音量 = 0(沒有聲音)、state 5'd16 為音量最大，從 state 5'd0 ~ 5'd16 音量遞增。每個 state 對應到的 state output 不同之處在於 vol 和 vol_minus，也就是聲音的振幅，隨著 state 的值越大而遞增。

最後將 vol、vol_minus、note_div 送入 speaker 模組(和圖六右下角相同)，產生 audio_mclk、audio_lrck、audio_sck 和 audio_stdin，並且把目前所在的 state 值轉換成 digit0(個位)和 digit1(十位)送入 scan_ctl 和 display，使得 state 值(音量的 level)可以顯示在七段顯示器上。

**5. Discussion**

本次的 Lab 重點為學會如何使用 speaker，有兩個重點：frequency divide 和 parallel to serial，其中需要注意：parallel to serial input 訊號和 output 訊號的頻率關係、clock 應該使用 negative edge trigger，這些都是為了最後能得到講義第七頁圖(圖四)timing diagram，做完之後基本上就沒有太大的問題了。第二題的第一小題和第二小題均為調整 buzzer control 的參數，並且加入 finite state machine，使得不同狀況下(按不同的按鈕)有不同的 state output。

**6. Conclusion**

本次的 Lab 我學會了一個 FPGA 板的外接功能，除了讓我更了解 parallel to serial 的實現，也讓我的期末 project 又多了一個東西可以使用。