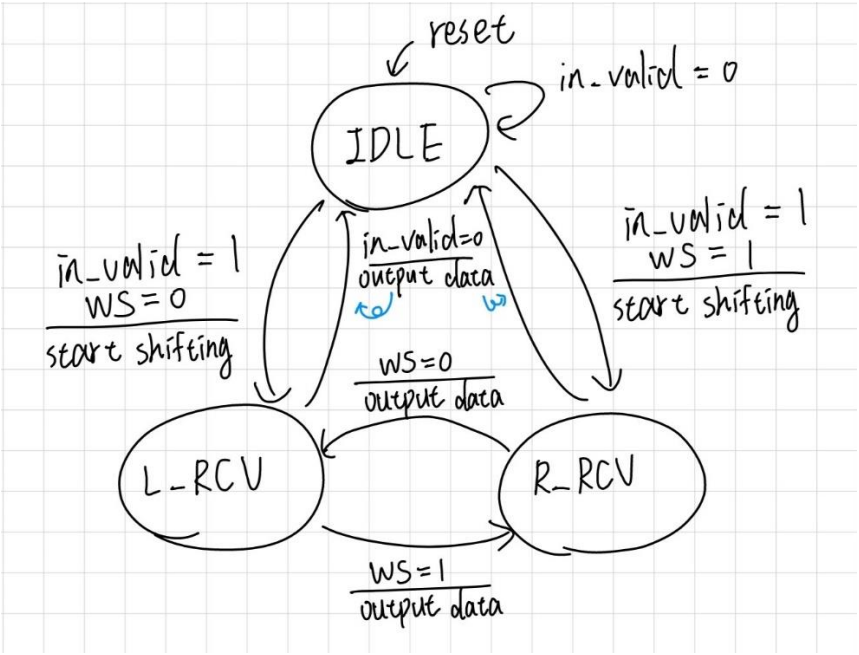
數位電路與系統 HW2 Report

110511233 李承宗

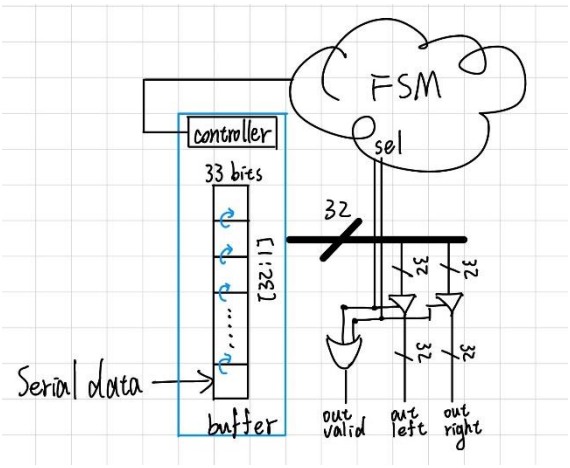
作業二是 Sequential Circuit，因此免不了設計 FSM，加上輸入輸出是 serial-in-parallel-out 的關係，所以可以先估計電路會有一組接收資料的 shift register。左右聲道的部分我打算用 demultiplexer 來選擇輸出 channel 的方式來實現，這樣就可以讓輸出共用同一組 register，減少資源用量(但可能會造成 glitch，希望這樣設計出來的電路不會對使用者造成影響)。

有了基本的構思之後，接下來就是設計的部分：從 FSM 開始，由於這次作業的電路牽涉到收發資料，因此會有閒置、接收、結束的狀態。最後我設計的 FSM 有三個狀態，分別如下：



*FSM*主要拿來控制是否接收*Serial Data*送來的資料，以及要輸出的聲道，因此我設計出來的狀態分別有：不需要接收資料的*IDLE*狀態、輸出至左聲道的*L_RCV*以及輸出至右聲道的*R_RCV*，我發現接收結束之後好像不需要其他特殊處理，因此我把*DONE*拔掉了。如此設計可以讓狀態切換的時候觸發輸出資料的事件，讓後續電路設計的難度降低。

在實現的時狀態控制的部分我還有多用一組 2-bit 的 register，用來記錄現在暫存器的資料應該輸出到哪個聲道（或不輸出），作為輸出電路的選擇線、*out_valid*的輸出。



架構圖如上，有前面設計好的*FSM*、輸出選擇線、輸出的邏輯電路以及一組33*bits*的*buffer*。其中*sel*會輸出00_2、10_2或01_2，代表著沒有輸出、左聲道輸出以及右聲道輸出，因此只要將*sel*的兩條線做*bitwise OR*就可以得到*out_valid*的訊號，*out_left*以及*out_right*的訊號則是用*demultiplexer*的概念來做輸出，但是輸出的時候選擇線只會有一條是 1，因此只需要兩組 32bits 的*AND Gate*來選擇輸出，這樣一來就不用請出*demux*了。比較難設計的部分是*buffer*，由於在左右聲道切換的當下需要輸出資料，還得將*SD*的資料讀進來變成下一次的輸出，因此*buffer*的操作模式不是單純的的移位跟*clear*：

```
buffer_nxt = out_valid ? {buffer[0], SD & in_valid} : {buffer[31:0], SD & in_valid};
```

藉由這段程式碼可以看出*buffer*會執行向左移位的動作，然後會有清除前 32bits（*buffer*為33*bits*）或是單純向左移位的動作，選擇方式為檢查當下是否有輸出（此頁右下為範例波形）：當下狀態有輸出的話，*buffer*裡面的前 32bits 就已經使用過了，因此下一個 state 就要把*buffer*裡面的東西清空。但是輸出當下以及觸發輸出的當下（即為輸出前一個 state）的資料還沒使用過，需要存起來。因此我處理的方式是留下*buffer*內的最後一個 bit 以及*SD*，這也是為什麼我的*buffer*會有 33bits 而不是 32bits。另外 shift-in 的資料是*SD* & *in_valid*而不是*SD*，是為了避免*in_valid*為 0 時*SD*還有資料輸入而做的處理。輸出取*buffer*內的前 32bits，再用*sel*的值決定輸出頻道，至此整個電路設計完成。

```
if      (sel[0]) out_right = buffer[32:1];
else if (sel[1]) out_left  = buffer[32:1];
```

sel/輸出	左聲道	右聲道
2'b00	不輸出	不輸出
2'b01	不輸出	輸出
2'b10	輸出	不輸出
2'b11	-	-