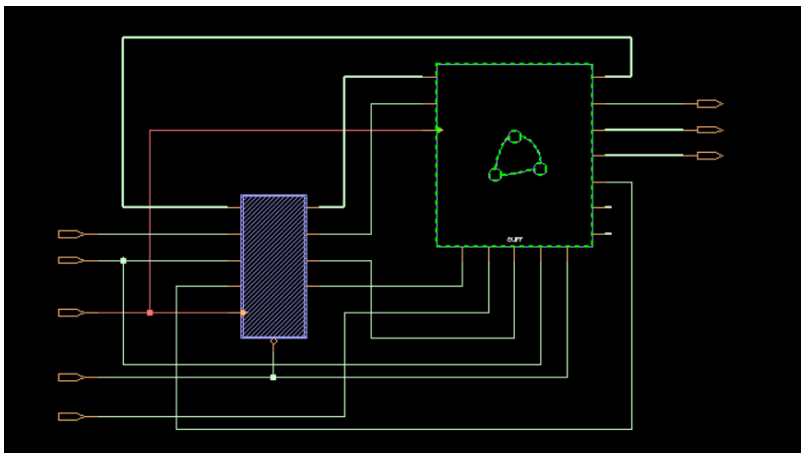
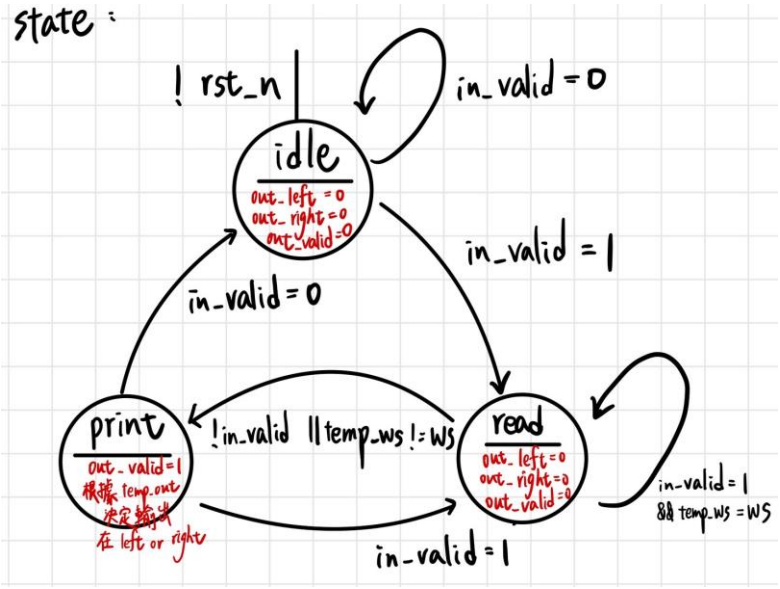
Report_dcs193 HW02 蔡東宏 110511277

1. 程式說明:

這次的 HW 為 Simplified I2S，根據 WS 的值分別將 SD(input data)存入左右道，最後在 WS 改變時，輸出存入的 data。這題我用了 3 個 state，分別為 idle、read、print，當一開始!rst_n 時，state 為 idle，當 in_valid=0 時，state 維持 idle，而當 in_valid=1 時，state 變為 read，此時開始讀資料(先將原本的資料向左 shift 1bit，再將最後一 bit 改為 SD)，當!in_valid 或 temp_ws!=WS(temp_ws 為 clk rising 時候的 WS 值)，state 變為 print，反之則 state 繼續為 read，當 state 為 print 時，out_valid=1，並且會根據上一個 WS 值決定要輸出在左邊還是右邊，而此時，當 in_valid=1 時，state 跳到 read，反之，state 則跳到 idle。

2.架構圖:





3.優化過程:

a.一開始我存資料的時候，我使用兩個 temp_shift 分別存左道的資料以及右道的資料，但後來拉 nWave 看的時候，我發現當我在某一道存資料的時候，另一道的值都會為 0，因此，我就將所有資料都存在同一個 temp_shift，要輸出的時候在看前一個 clk rising edge 的時候，WS 值是多少(我用 temp_out 存前一個 clk rising edge 的 WS)，所以 temp_out=0 的時候，輸出在左道，反之，則輸出在左道。如此一來，在合成的時候，原本需要兩個 32bit 的 flip-flop，但修改成使用一個 temp_shift，只需要一個 32bit 的 flip-flop，因此，改完之後，我的面積從 10000 多減少到 3985。

b.一開始我沒有使用 print 這個 state，在 read state 要輸出的時候，我多了一個變數存是否要輸出，然後經過 flip-flop，再下一個 clk rising edge 的時候輸出，但我後來發現，多一個 print state 雖然 state 這個 flip-flop 會多 1bit，但要多一個變數也需要一個 flip-flop，並且需要更多判斷式，會讓面積變得更大。

4.結論:

這次作業為 Simplified I2S，我用了 3 個 state 去處理此問題，雖然在第一次做出來的時候面積非常大(約 10000 多)，但在後來慢慢觀察 nWave，觀察哪一部份可以優化，最後發現左右道可以寫在同一個 temp_shift，不用分開做，如此一來，有效的減少我的面積。因為還不熟悉硬體描述語言，所以這次 HW 我花了滿久的時間做，在 debug 的時候看 nWave 成功有效讓我抓到很多小細節的 bug。在這次作業的長時間 debug 過程中，我慢慢熟悉硬體描述語言，在之後的 HW 和 LAB 應該會更快做出來。