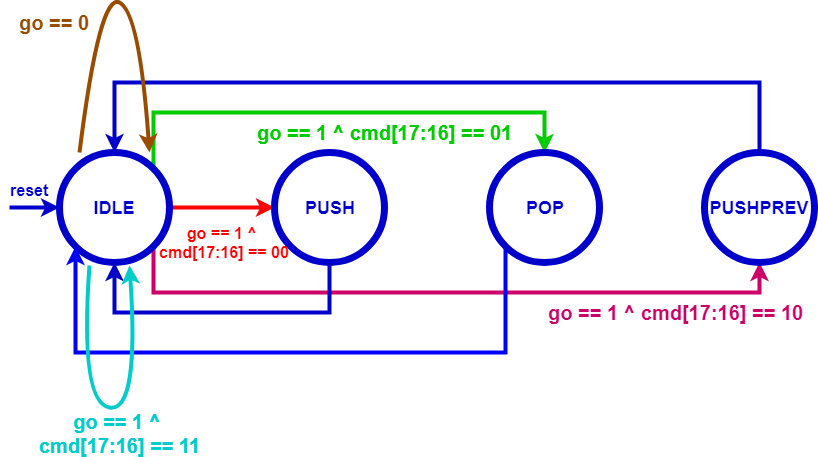
**106030012 廖昱瑋 工院21 Lab4**

1. FSM



圖(一) block diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **IDLE** | **PUSH** | **POP** | **PUSHPREV** |
| **ready** | 1 | 0 | 0 | 0 |
| **w\_en** | 0 | 1 | 0 | 1 |
| **r\_en** | 0 | 0 | 1 | 0 |

combinational circuit共有四個state，分別是IDLE、PUSH、POP、PUSHPREV，reste後進入IDLE，go == 0時一直待在IDLE，接著go==1時，依據cmd前兩個bits決定要進入哪個state，進入PUSH、POP、PUSHPREV運算完後，再回到IDLE準備接收下一個資訊。

1. 設計過程

主要為兩個always block，一個為flip-flop，另一個是combinational circuit，combinational circuit有四個state，分別是IDLE、PUSH、POP、PUSHPREV。reset==1時，reset所有值進入IDLE，直到go==1時，再進入其他cmd要求的state，而每次到clk的posedge，就把combinational circuit運算完的所有next\_變數，傳給主要變數。

而為了避免還沒運算完成，就被testbench檢查變數，ready的值就擔任很重要的角色，在IDLE state時ready設為1，其他三個state ready為0，而每次進到PUSH、POP、PUSHPREV後，next\_state設為IDLE，這樣代表運算時ready會是0，運算完後ready又變回1，這樣就可以在運算後變數才被檢查，而且運算完，可以隨時準備再讀取資料。

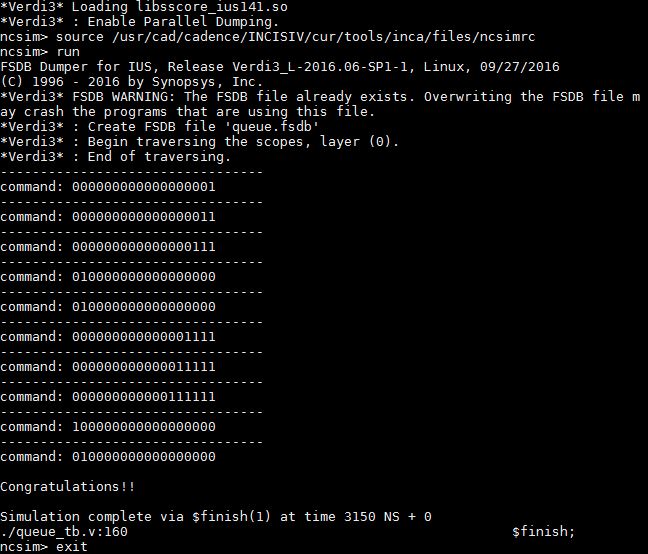
1. 問題及解決方法

最一開始我打完的時候發現語法錯誤，而且錯誤很奇怪，一直抓不出來，後來花了一個小時才發現是在define的地方不需要打分號，可見對語法熟悉是麼重要的事。

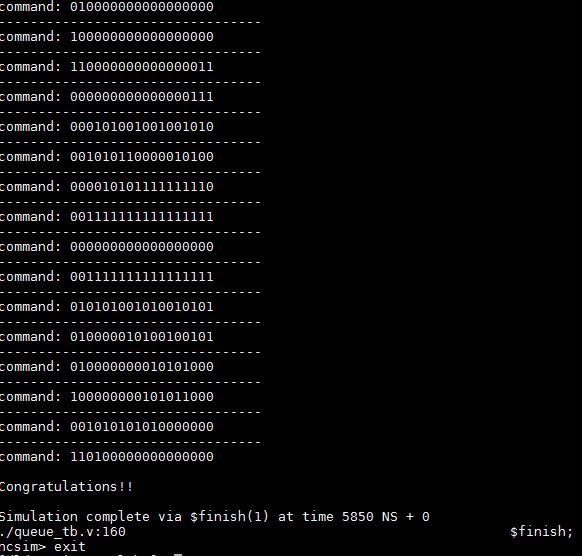
一開始combinational circuit部分我分成太多個always block寫，不方便管理，一直出錯，ready的地方一直保持是1，後來我把它合併為一個always block，整個思緒就變清晰許多，比較好debug。

剛開始我的make sim0、1、2都過了，但make syn0、1、2都沒過，原因是addr出錯，剛開始的PUSH應該要PUSH到ram的第0格，但我都PUSH到第1格，後來發現是因為我在combinational circuit直接把位置的值給addr，而沒有多設一個next\_addr變數，在combinational circuit部分完成next\_addr運算，於positive edge時再傳值，所以addr在一瞬間就被加1，數值變成1了，w\_num則一開始就被存到ram[1]，以後在寫sequential circuit時要更小心才是。

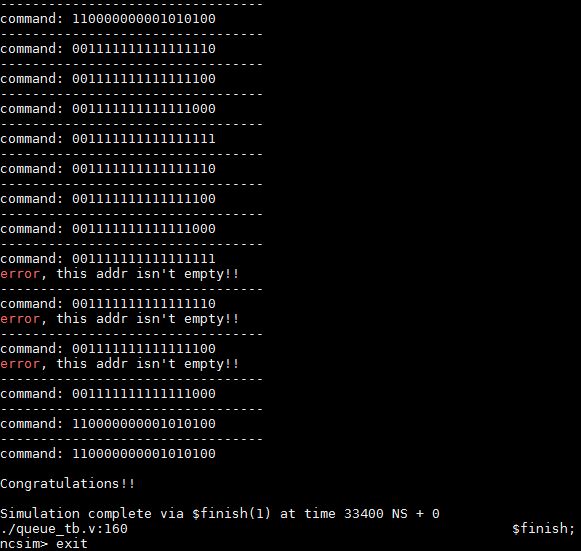
1. 模擬、合成結果



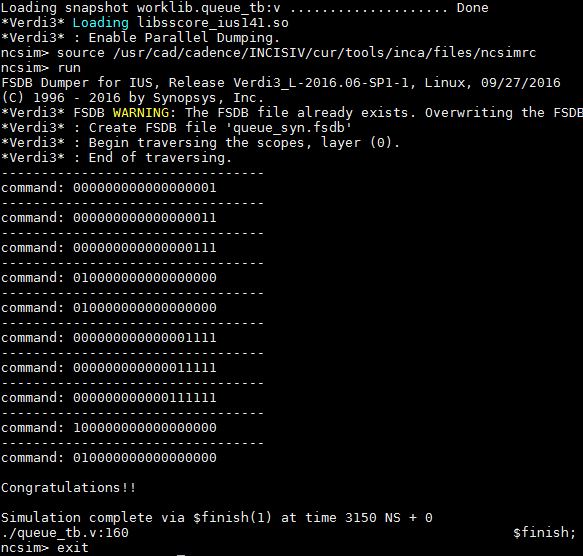
圖(二)make sim0



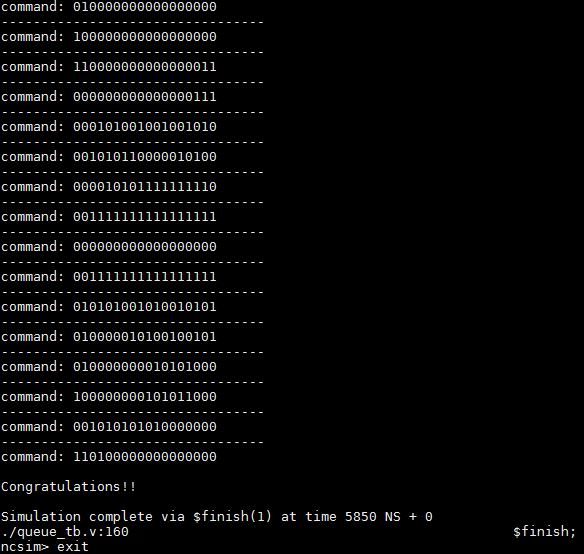
圖(三) make sim1



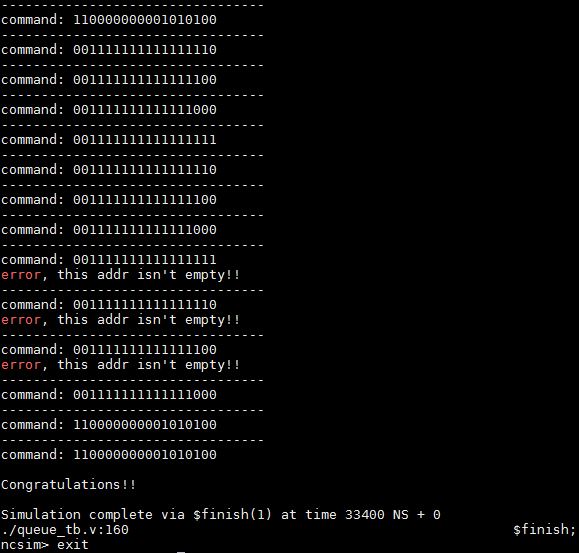
圖(四) make sim2



圖(五) make syn0



圖(六) make syn1



圖(七)make syn2