CA HW5 Report

Q1: General specification of the cache unit

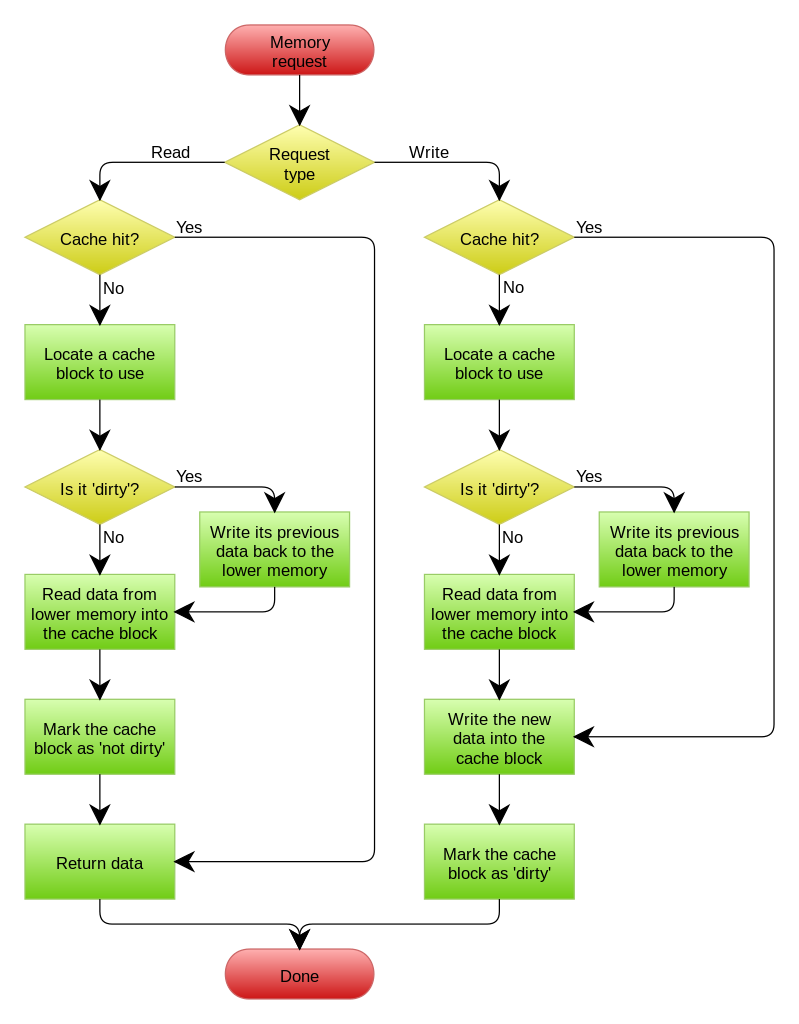
A1:這次作業是設計cache，那我是用二維的reg[154:0]CACHE[7:0]來當作direct-maped cache，其中[154]是valid bit、[153]是dirty bit、[152:128]cache tag、[127:0]則是裝四個word的data，並透過reg [127:0] block來做運算，等到每個posedge clk來的時候再跟新值回去CACHE，一開始會先判定是proc\_read or proc\_write 如果都不是，那就是在最一開始的地方，把所有控制訊號設為0，proc\_stall設為1，那如果是read的話，就先判定是否為valid，若不為valid，先從memory把data拿來放入block，接下來再判定是否為hit，hit的話就直接讀值，miss的話就要看dirty bit是不是1，如果是0，就從memory拿data放進CACHE再做讀，如果dirty bit是1，那就把資料先write back回memory再locate新的資料回CACHE，並做讀值，還要把dirty bit改回0，proc\_stall也會改回0。如果是write 的話也很類似，但是沒有先判斷是不是valid，而是直接看是hit or miss，如果是hit就是改值並且把dirty it改為1，如果miss的話，要看dirty bit是不是1，如果是0，就去memory拿值（改tag）並把新的值寫入，dirty bit改為1，若dirty bit是1就要先write back回記憶體，再從記憶體拿新的值回來CACHE並寫新的值進去CACHE，然後把

dirty bit設為1，用proc\_wdata寫入新的值後，proc\_stall都要改回0。

Q2:Read/Write Policy

分別有write through和write back兩種，write through的話不用用到dirty bit，因為他只要一發生改變就會存回memory，雖然比較慢，但是可以確保，memory和CACHE裡的值是一樣的，那如果使用write back的話，則兩邊的值可能會不一樣，但是不用一改值就送回memory，只有他要再改的時候才需要存回memory（By dirty bit）。

Q3:State Graph



Q4:

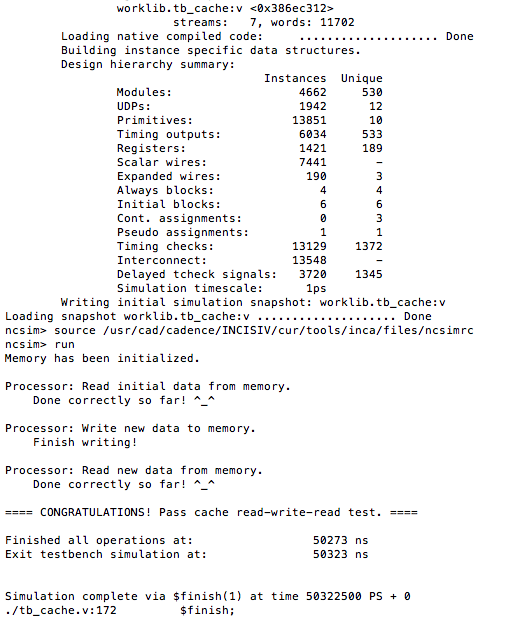
Read/Write 的miss rate應該都和block的size有關，這次要求的block size為4-word，所以一次miss會更新四個word的共同tag，miss rate 應為25%。

若為read/write operation沒有經過write back的話，會先stall 6個cycle，在每個word又1個cycle，總共10個cycle，如果需要write back，則會stall 13個cycle，在每個word又1個cycle，總共17個cycle。

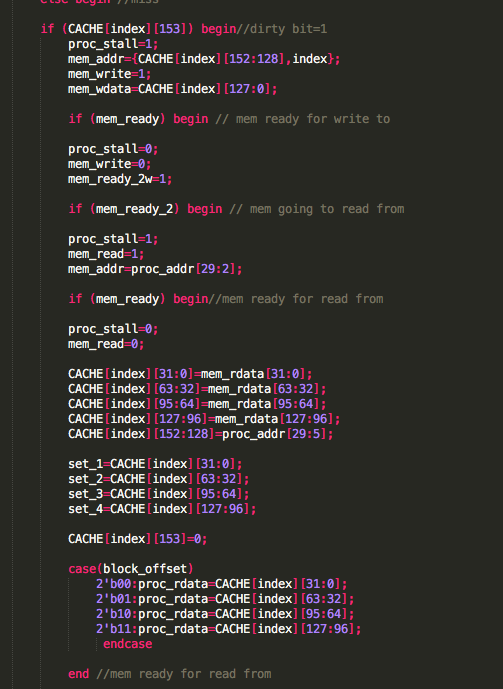
Q5:

CLOCK CYCLE TIME:5ns

TOTAL Cell Area:74020 um^2



補充：

這次在寫作業遇到最大的問題是如何在write back的時候stall兩次，一開始是這樣寫

但是mem\_write不會上升，所以也不會mem\_ready兩次，後來嘗試在這裡寫state graph：



 proc\_stall也還是不受控ＱＱＱＱ，最後是乾脆在一開始就把proc\_stall設為1，到要output才把proc\_stall調回0，才解決問題，這次nWave真是看到眼睛都快脫窗了ＱＱ