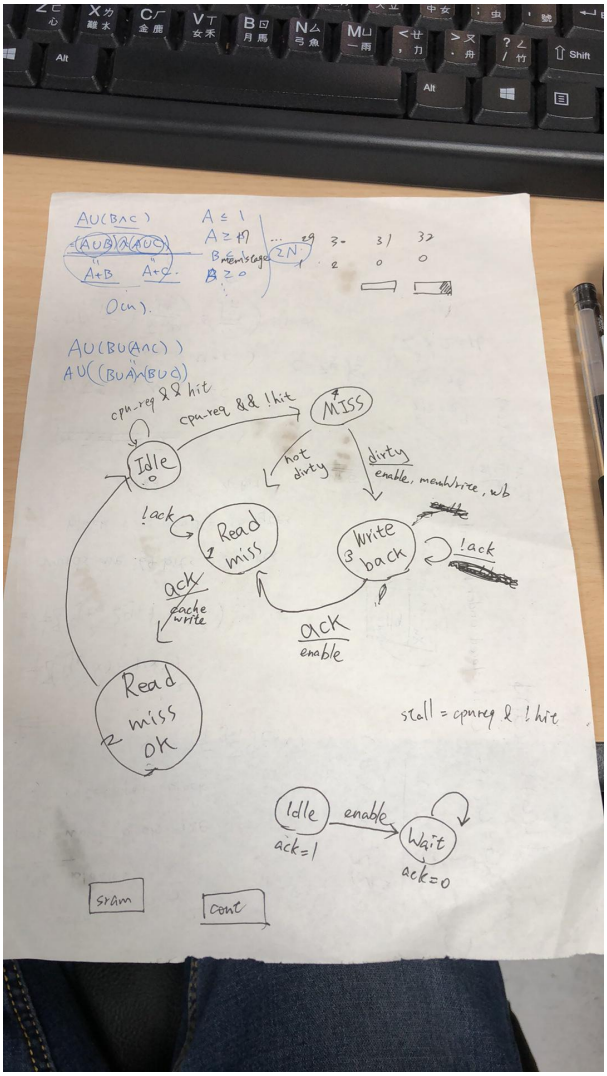
I’m Pasta 紅茶🧃 Project 2 Report

Module 說明



- dcache_controller
  dcache_control 主要用來控制CPU, SRAM 和 DRAM之間資料的移動。在本次Project中使用Write back和Write allocate來處理Wrtie hit和Write miss，因此 control 的運作機制如下:
  - Read hit or Write hit:正常運作由SRAM提資料。
  - Read miss:第一步先判斷SRAM空位是否dirty，若dirty則需寫回DRAM接著由DRAM提取資料後先放回SRAM再讀出傳回CPU。

    - Write miss:和Read miss一樣第一步先判斷SRAM是否dirty，若dirty則需寫回DRAM再由DRAM提取資料後放回SRAM，此時不同之處是在DRAM取回資料後在control會先將要寫入的資料寫好並將dirty bit設1後再寫入SRAM。
    - 實作:實作上主要透過設定在不同state下的signal來達成上方所敘述的運作模式，詳細的設定如上方圖所示，比較需要注意的地方是mem_enable這個signal在設定後的下一個cycle開始時DRAM就會開始計時，所以在STATE_IDLE到STATE_MISS的時候就要設為1 (不然的話會慢一個cycle，然而其實這不會影響答案)。
- dcache_sram
    - dcache_sram負責sram裡存放的data，此次project實作的是2-way associative cache，因此按照spec裡給的圖例，判斷兩邊是否有其中一邊hit，以及維護LRU的正確性。
    - 實作上，需判斷該cycle是否需要動到sram(enable_i)，以及判斷是read還是write(write_i)，比對傳進來的tag是否和該index上的tag一樣及valid bit是否為1，如果兩邊有一邊比對成功則為hit。當其需要從sram read的時候,如是hit的話則回傳hit到的data及tag。當其需要write進sram的時候，如果hit則把傳進來的tag及data寫入hit到的一邊，如果miss則需踢掉LRU紀錄的block並寫入，也就是踢掉比較早使用的那一邊。對於sram的tag及data的output而言，如是miss的情況都是回傳根據LRU所紀錄的來回傳，否則都是回傳hit到的一邊。
    - 至於LRU的實作，由於為2-way associative cache，一個index會對應到兩個位置，所以開一個陣列紀錄每個block中，兩邊哪一邊最早被存取。如有hit的狀況，則代表沒被hit到的那一邊是比較早被存取的，如是miss且是write，則LRU的值須反轉，因為miss時根據LRU踢掉且寫入的block會變成最近存取的，其餘則LRU維持原本的紀錄。
- CPU
    - memory造成的stall是「每個」pipeline stage都要stall，跟原本data hazard和control hazard的stall只有IF/ID要stall不同，因此要做區別
    - cpu要負責做cache controller和dram的橋梁，把訊號和資料pass thru

組員與工作分配

1. dcache_sram
2. dcache_controller
3. Data_Memory
4. CPU (Stall && Merge && Test)
5. Report

{1, 3} {2} {4}

謝獻沅 (學號B07902049)：{1, 3}
黃昱翔 (學號B07502159)：{2}
陳正康 (學號B07902143)：{4}
三人合作：{5}

遇到的問題

- 在SRAM裡面的TAG共24bit包涵了valid和dirty bit，注意要判斷有沒有hit只要看[22:0]就可以了
- LRU要針對每個block都使用一個bit來決定下一次要踢掉哪個，原本是16個block共用一個，結果跑出來怪怪的

執行環境

主要iverilog 10.3 on Ubuntu 20.04
有些debug是在ModelSIM 10.4a on Windows 10

Repository

Link (https://github.com/I-m-Pasta-Black-Tea/junk-cpu/tree/memory-hier)