Computer Architecture Lab 2 Report

B08902055 李英華

Modules Explanation

dcache_controller

- 取代原先data memory位置，除了data memory原先有的input, output ports外，還有另外與off-chip data memory的intereface
- 因為miss penalty需要stall cycles，所以也會連結到各個pipeline latches，作為stall input
- Hit: 當2-way的cache有其中一個的tag bits與CPU端傳入的tag相同且該cache entry是valid時成立，否則miss
- assign r_hit_data: 不論是bring from memory to cache 然後 hit，或是本來就 hit，都是從dcache_sram的 output data，所以assign sram_cache_data
- cpu_data 和 w_hit_data: cache line 256 bits，每次讀寫32 bits，所以透過 (cpu_offset >> 2) * 32 來找到對應的資料位置。其中 >> 2 再 << 2 只是為了 alignment，雖然在這個作業中不用考慮
- STATE_MISS: miss 需要 access memory，所以 mem_enable <= 1'b1。如果 dirty，需要 write back，所以 mem_write 和 write_back 皆為 1，並進入 STATE_WRITEBACK。若為 clean，則不需寫回，剩下的流程與 STATE_READMISS 相同
- STATE_READMISS: 若 ack，代表 memory data ready，可寫入 cache，所以不需要 mem_enable，但 cache_write，並進入 STATE_READMISSOK；若尚未 ack，留在這個 state
- STATE_READMISSOK: 只 data 已經存入 cache，所以不需 cache_write，並回到 STATE_IDLE
- STATE_WRITEBACK: 若 ack，則 dirty data 已經寫回 memory，不需 mem_write 和 write_back，並進入 STATE_READMISS 等待 data ready 然後存入 cache；若尚未 ack，留在這個 state

dcache_sram

- LRU policy: 設定一個陣列 (reg record[0:15]) 存取下一個要被取代的 cache block，對於 read miss，因為只有 2-way，其實只需要在每次寫入後 flip bit，就可以實現 LRU。對於 write hit，如果其中一個 hit，就把另一個設為下一次要被取代的 block
- tag_o 和 data_o: 也是在判斷 tag bits 是否和 tag_i 相同且valid (意即是哪一個 way hit)，但如果兩者都沒有 hit，必須決定取代哪一個，所以傳回舊的 tag bits 和 data，讓 controller 透過 dirty bit 決定是否寫回。

Others

除了 pipeline registers 和 PC 需要新增 mem stall input 之外，其餘跟上次作業一樣。

Difficulties Encountered and Solutions of This Lab

- dcache_controller: 比較大的難點是要理解 controller、以及 state 轉換的意思，並且因為 sample code 的變數跟自己習慣的取名方式不太一樣，一開始會容易找不到到底是需要 assign 給哪個變數或用哪個變數判斷
- dcache_ram: 一開始自己的做法只有決定每次要取代哪個 block，反而忘記考慮 write hit。後來重看 dcache_controller 之後並請教同學的作法，才有最終的結果
- 原先的 testbanch 給的 tag bits 只有 24 bits，並且沒有在 flush to memory 的時候考慮 valid bits，因此最後一個 cycle 的值都會很奇怪。並且加上這次作業會因為實作方法的不同而會 dump 出不同的 read data 結果，一開始看 cache.txt 時會不知道自己對或錯。雖然 spec 只有提到只會看 write/read miss 的順序，而並未提及 read data，但後來與同學討論與自己的思考後有了解到背後原因，算是更進一步理解這個 lab 2 背後涉及的機制與原理。

Development Environment

- OS: Windows
- Compiler: iverilog