HW#3 Report
Cache Optimization

邱振源, 111550100

***Abstract*—這次的做作業主要是針對 Aquila 的 D-cache 進行分析，嘗試在固定 cache size(4KB/8KB)的情況下改變 cache way 的數量，或是改變 cache 的 replacement policy，同時分析在這些改變之後 cache 針對 read/write 的 hit rate，與 miss 之後產生的 average read/write miss latency，而做進一步的分析與探討。**

Keywords—RISC-V; Aquila; π; chcaet; profile circuit; memory request; cache way; replacement policy; FIFO; LRU; random

I. INTRODUCTION

這次的作業我主要將 cache size 設為 8KB，並透過分析 data cache controller 的 FSM 的 state 與相關的 signal，設定不同的 flag 來分析在 cache 的 read/write 資訊、read/write hit rate 及 average miss latency。

我主要的改動是 cache way 的數量與 replacement policy 的設定。在更改 cache way 的部分，我分別探討了 direct mapped、2-way、4-way 與 8-way，而在 replacement policy 的部分，則分別實作了 First In Fitst Out (FIFO) 、Least Recently Used (LRU)與 random 的 policy，探這些改變對於 cache 的效能影響。

II. CACHE PROFILING AND IMPLEMENTATION

A. Read/Write Counting

每一次 memory request 的時候，cache controller 的 state 會先回到 Idle，同時也會收到 p_strobe_i 的訊號，而當 state 在 Idle 的時候，可以藉由 p_rw_i 的值來判斷 read/write，若 p_rw_i 為 0 是 read memory request，而 p_rw_i 為 1 則是 write memory request。

由於我將 profile 的電路撰寫在 dcache.v，但在 elf 檔進入 uart 之前也會經過一些 write memory 的 instruction，為了讓量測的範圍更好的局限在執行 pi 程式的部分，我將 core_top.v 的 exe2mem_pc 也傳入 dcache 中，如此便可以藉由 pc address 判斷當前執行的部分。

B. Read/Write Hit

根據 cache controller 的 FSM，如果 state 在 Analysis 且 cache_hit 的值為 1 時代表 cache hit，由於此時的 state 已經不在 Idle，所以透過 rw 的值來判斷 read/write，rw 為 0 是 read，而 rw 為 1 則是 write。

在 hit 的時候，無論 read/write 都會有 1 個 cycle 的 hit latency，來回傳資料同時將 cache controller 的 state 回到 Idle，有關 hit latency 的部分也在後面的資料驗證。

C. Read/Write Miss

在 miss 的部分主要可以分成兩個情況，如果所需的 data 並非 dirty，cache controller FSM 的 state 會從 Analysis 跑到 Rd from Mem，反之，則需要經過 Wb to Mem 和 Wb to Mem Finish 才到 Rd from Mem。

但無論 data 是否為 dirty，cache controller 在回到 Idle 之前都需要先經過 Rd from Mem Finish 的 state，因此我在這個 state 的地方判斷 cache miss 的次數，read/write 的部分和 hit 一樣用 rw 判斷，而 read/write 的 miss latency 則是 cache controller 從 Analysis 到 Rd form Mem Finish 的 cycle 數。

D. Cache Way

在修改 cache way 的部分，除了要改變 N_WAYS 這個 parameter 的數值之外，也需要針對 way_hit、cache_hit 和 hit_index 的判定方式進行修改，根據 way 的值調整 assign 的布林運算式或 assign 到的值。

E. Replacement Policy

在修改 repalcement policy 的部分我主要實作了兩個 policy，第一個是 Least Recently Used (LRU)，第二個則是 random。在觀察 FIFO 的寫法之後，我發現如果要時做出這兩個 replacement policy，主要要修改的地方是 FIFO_cnt 的統計方式和 victim_sel 的判斷，因此我也從這個部分下手，以下適時做的細節。

a) LRU

新增一個大小為 N_LINES*N_WAYS 個 WAY_BITS 大小的 2-D array (LRU_cnt)，和 FIFO 一樣用 line_index 來取得第幾個 line，而 ways 的部分則是記錄不同 way 被使用的情況，放在 0 是最久沒用的，way 的 index 越大代表他越近被使用過。因此 victim_sel 固定選取 LRU_cnt[line_index][0]的地方，就是 least recently used。

為了保持 LRU_cnt 的狀態，在 initial 的時候我將各個 line 的不同 ways 設為 0 到 N_WAYS-1。而每次 hit 的時候就將 hit 的 way 移到 N_WAYS-1 的地方，其他則依序往前移一格，這樣一直沒有 hit 的 way 就會被移到 0 的位置，而 victim_sel 就能順利選到 least recently used。而 miss 的時候，因為 victim_sel 會將[line_index][0]的地方換掉，所以 state 在 Rd from Mem Finish 的時候要記得將[0]的地方移到[N_WAYS-1]。

b) Random

random 的實作部分就比較簡單，主要是將 victim_sel 的隨機附值，同時用 N_WAYS 對該值取餘數，如此一來就可以確保 victim_sel 的範圍不會超過 ways 的數量。

III. Result And Analysis

A. Discussion of Cache Size

TABLE I. Result Of 4KB FIFO Cache In Different Way Number

Item	Dir. Map	2-way	4-way	8-way
Total Cycle	1566142944	1555061943	1555028025	1555028800
Read Rate	58.15%	58.15%	58.15%	58.15%
Read Hit Rate	77.01%	77.57%	77.57%	77.57%
Avg Read Hit Lat	1	1	1	1
Read Miss Rate	22.99%	22.43%	22.43%	22.43%
Avg Read Miss Lat	55.61	55.61	55.61	55.61
Write Rate	41.85%	41.85%	41.85%	41.85%
Write Hit Rate	83.30%	83.30%	83.30%	83.30%
Avg Write Hit Lat	1	1	1	1
Write Miss Rate	16.70%	16.70%	16.70%	16.70%
Avg Write Miss Lat	55.53	55.53	55.53	55.53
Cache Hit Rate	79.64%	79.97%	79.97%	79.97%
Cache Miss Rate	20.36%	20.03%	20.03%	20.03%
Wb Miss Lat Rate	48.25%	47.88%	47.88%	47.88%
!Wb Miss Lat Rate	32.00%	31.77%	31.77%	44.67%

TABLE II. Result Of 8KB FIFO Cache In Different Way Number

Item	Dir. Map	2-way	4-way	8-way
Counting msec	25293	25445	29076	29848
Total Cycle	1293841010	1301452591	1482996390	1521583877
Read Rate	58.15%	58.15%	58.15%	58.15%
Read Hit Rate	84.61%	83.14%	80.58%	79.21%
Avg Read Hit Lat	1	1	1	1
Read Miss Rate	15.39%	16.86%	19.42%	20.79%
Avg Read Miss Lat	55.57	55.61	55.61	55.52
Write Rate	41.85%	41.85%	41.85%	41.85%
Write Hit Rate	91.83%	93.36%	84.17%	83.30%
Avg Write Hit Lat	1	1	1	1
Write Miss Rate	8.17%	6.64%	15.83%	16.70%
Avg Write Miss Lat	55.52	55.50	55.50	55.48
Cache Hit Rate	87.64%	87.42%	82.08%	80.92%
Cache Miss Rate	12.36%	12.58%	17.92%	19.08%
Wb Miss Lat Rate	37.35%	37.72%	45.34%	46.73%
!Wb Miss Lat Rate	25.40%	25.63%	30.24%	31.58%

一開始我將 cache size 設為 4KB，並將結果整理成 TABLE I，然而，根據 TABLE I，雖然在 read 的部分，cache hit/miss 的數值有些微的不同，然而，在其他部分，無論我們修改 cache way 的數量，數值皆沒有明顯的變化。因此，我重新將 cache size 設為 8KB，並將結果整理成 TABKE II，可以發現數值上的變化較為明顯，有關後面的討論我也都將 cache size 設為 8KB，如此一來，優化的效果與數值上的差距也會更方便討論。

B. Disscussion of Cache Ways with FIFO

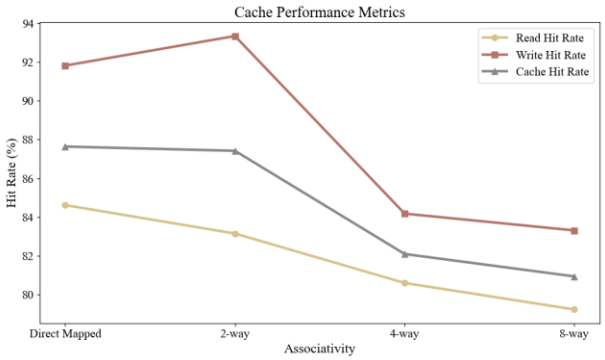

Fig. 1. Hit Rate of Different Cache Ways with FIFO

Fig. 1 是整理 TABLE II 在 cache size 為 8KB 的時候 read/write 和整個 cache 在不同 cache way 的時候的 hit rate，狀況。根據 Fig. 1 可以發現，隨著 cache associativity 提升，hit rate 下降，在觀察 pi 的 objdump 檔之後，推測可能是因為，在執行 pi 程式的時候，其 memory access 模式主要集中在連續地址範圍，因此，當 cache 為 direct mapped 形式的時候，由於 memory block 固定對應到特定的 cache 位置，減少了 cache block 被替換的頻率，因此 hit rate 較高。而在較高 associativity 的情況下，cache set 中的替換行為可能導致 hit rate 略微下降。

C. Discussion of Average Read/Write Latency

根據 TABLE II，在 cache hit 的時候，因為 cache controller 會用 1 個 cycle 去傳出 data 和回到 Idle state，所以無論 read 或 write，average hit latency 皆為 1。而無論 read/write，在 miss 之後的平均 latency 都有大約 55 個 cycle，同時 write 的平均 latency 也隨著 associativity 上升而下降，miss 的高 latency 也可以看出提升 cache hit rate 的重要性。

D. Discussion of Using LRU As Replacemeny Policy

TABLE III. Result Of 8KB LRU Cache In Different Way Number

Item	Dir. Map	2-way	4-way	8-way
Counting msec	25293	25123	25107	31059
Total Cycle	1293841010	1285336428	1284546236	1582125436
Read Rate	58.15%	58.15%	58.15%	58.15%
Read Hit Rate	84.61%	85.46%	86.41%	79.19%
Avg Read Hit Lat	1	1	1	1
Read Miss Rate	15.39%	14.54%	13.59%	23.81%
Avg Read Miss Lat	55.57	55.50	55.50	55.58
Write Rate	41.85%	41.85%	41.85%	41.85%
Write Hit Rate	91.83%	91.12%	89.93%	83.30%
Avg Write Hit Lat	1	1	1	1
Write Miss Rate	8.17%	8.78%	10.07%	16.70%
Avg Write Miss Lat	55.52	55.42	55.38	55.53
Cache Hit Rate	87.64%	87.87%	87.88%	79.17%
Cache Miss Rate	12.36%	12.13%	12.12%	20.83%
Wb Miss Lat Rate	37.35%	36.94%	36.90%	48.77%
!Wb Miss Lat Rate	25.41%	25.14%	25.11%	32.31%

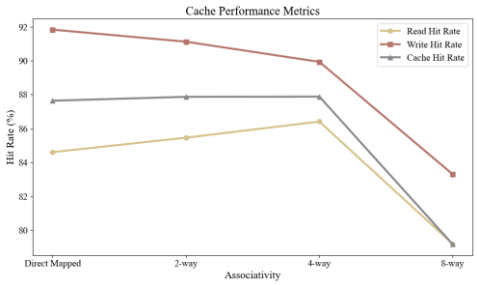

Fig. 2. Hit Rate of Different Cache Ways with LRU

TABLE III 紀錄了在不同 associativity 下用 LRU 當作 replacement policy 的結果，除了 8-way 以外可以發現 LRU 的整體 hit rate 較 FIFO 高，且執行速度較快，在觀察 Wb Miss Latency Rate 之後可以發現，LRU 在 write back 的 cycle 佔整體的比例明顯較少，推測是因為 LRU 會換掉最久沒被使用的 memory access 位置，因此在執行 pi 時一些只需讀取一次的地方就會被優先取代掉，需要較常被 request 的位置會存取在 cache，在 data 是 dirty 之後不會需要很頻繁的被寫回 memory，因此加快了執行速度。

另外 Fig. 2 也統計了 read/write 和整個 cache 的 hit rate，可以發現和 FIFO 不同，在 LRU 時，4-way cache 有較高的 hit rate，速度也相較原本提升了 16%，而在 8-way 時，hit rate 快速下降，推測可能是因為 4-way 時，LRU 可以有效的將執行 pi 時一些不常被 request 的地方優先被 replace 掉，但仍避免在連續的 memory access 時可能會都在同一個 block 而需要一直 replace 的問題，從而提升 hit rate；而 8-way 時則因為連續要求附近的 address 而常常需要 replace，使的 hit rate 降低。

E. Discussion of Using Random As Replacemeny Policy

TABLE IV. RESULT OF 8KB RANDOM CACHE IN DIFFERENT WAYS

Item	Dir. Map	2-way	4-way	8-way
Counting msec	25293	30739	30740	31059
Total Cycle	1293841010	1566135181	156616426	1582122348
Read Rate	58.15%	58.15%	58.15%	58.15%
Read Hit Rate	84.61%	77.01%	77.01%	76.19%
Read Miss Rate	15.39%	22.99%	22.99%	23.81%
Avg Read Miss Lat	55.57	55.61	55.61	55.58
Write Rate	41.85%	41.85%	41.85%	41.85%
Write Hit Rate	91.83%	83.30%	83.30%	83.30%
Write Miss Rate	8.17%	16.70%	16.70%	16.70%
Avg Write Miss Lat	55.52	55.53	55.53	55.53
Cache Hit Rate	87.64%	79.64%	79.64%	79.17%
Cache Miss Rate	12.36%	20.36%	20.36%	20.83%
Wb Miss Lat Rate	37.35%	48.25%	48.54%	48.77%
!Wb Miss Lat Rate	25.41%	32.00%	32.19%	32.31%

TABLE IV 紀錄了在 cache 使用 random replacement policy 的結果，可以發現 read 的 hit rate 隨 associativity 上升而下降，但 write 的 hit rate 卻沒有太大的變化，推測可能是因為用 random 隨機 replace cache 的 data 可能會破壞在執行 pi 的時候，一些常見的 memory access 位置，進而使的 hit rate 下降。

另外，可以發現用 random 作為 replacement policy，Wb Miss Latency Rate 比例明顯比另外兩個多很多，可能是因為 random 會選到 pi 連續的 memory access，使這些 address 的 dirty data 常需寫回 memory 又讀出，進而拖慢執行時間。

F. Resource Usage

TABLE V. RESOURCE USAGE OF 8KB FIFO CACHE

Item	Dir. Map	2-way	4-way	8-way
Slice LUTs	884	1768	2490	2599
Slice Registers	876	1130	1134	1068
Slice	374	620	828	860
LUT as Logic	860	1736	2458	2567
LUT as Memory	24	32	32	32
Block RAM	2.5	5	10	20

TABLE V 記錄了在使用 FIFO 作為 replacement policy 的時候不同 associativity 對資源使用的區別，可以發現，隨著 cache way 的 size 提升，對資源的使用明顯成長，而 direct mapped 的方式可以用較少的資源完成 cache 的設計。

TABLE VI. RESOURCE USAGE OF 8KB 4-WAY CACHE

Item	FIFO	LRU	Random
Slice LUTs	2490	2549	1666
Slice Registers	1134	1900	872
Slice	828	1123	584
LUT as Logic	2458	2517	1364
LUT as Memory	32	32	32
Block RAM	10	10	10

再來，我分析了在同樣都是 4-way 的情況下，三種 replacement policy 對資源使用的區別，可以發現 LRU 對資源的使用較大，推測是因為我使用了 2-D array 來完成 LRU 的設計，而 FIFO 只需要 1-D array，Random 只需要直接隨機取值，少了為了 replacement policy 而需要的暫存器，使的資源的使用較低。

IV. CONCLUSION

這次的作業主要是探討在固定 cache size 為 8KB 的情況下對改變 cache way 數量和換替換策略對 cache 效能的影響。首先是 cache way，在 replacement policy 為 FIFO 時，direct mapped 的 hit rate 較高，可能是因為 pi 的 memory access 較連續，使用 direct mapped 減少了替換頻率。

再來是改變 replacement policy 的部分，LRU 相較於 FIFO 和 random 有較高的命中率，尤其是在 4-way 的時候命中率較高，其效能約相較原本提升 16%，因為其能有效保留頻繁使用的資料區塊，而 random 在執行 pi 程式的時候可能因為隨機替換破壞局部常見的 memory access address，而常需要 wb，進而影響效能。

在 latency 的部分，hit 的延遲為 1 個 cycle，一旦 cache miss 則需要約 55 個 cycle，也可以驗證老師上課說 cache 對比於 onchip memory 來說會降低效能。而在資源使用的部分，LRU 在提高命中率的同時需要更高的硬體開銷。這表明在實際設計中需要根據具體應用場景平衡效能與資源使用。