A Cache Design to Optimize Memory Performance in Mobile Processor

Su In Kim, Chang Min Eun, Hyun Hak Cho, and Ok Hyun Jeong
Department of Electronic Engineering, Sogang University
Sogang Univ., Sinsu-dong, Mapo-gu, Seoul 121-742, Korea
E-mail: suinkim@sogang.ac.kr

Abstract: In recent years, the processor performance has been improved rapidly, but the enhancement speed of memory performance is slow. Due to this gap, the improvement of memory performance is significant, and enhancing the cache performance can assist that. For this reason, we designed the cache that optimizes memory performance in the target processor, Cortex-A57. To evaluate memory and processor performance, Average Memory Access Time (AMAT) and Cycles Per Instruction (CPI) were used, and the simulation was conducted by using SimpleScalar3.0 and benchmarks from SPEC CPU2000. As a result, the performance was enhanced by 34.93% in AMAT and 10.05% in CPI with proposed design which is optimized to target processor. It is contributed to performance improvement of mobile processor as well.

Keywords—**Cache, Memory Performance, AMAT**

1. Introduction

Lately, according to development of mobile processor, high performed mobile devices have been released. As using high clock frequency and multi-core processor, mobile processor performance has been advanced. On the other hand, memory system performance is slow in progress. There are some reasons why the disparity exists. The prime reason is different aims of microprocessor and memory fields. The microprocessor field has pursued high speed, but the memory field has increased in capacity not a speed. The result of this two different aims lead to the processor-memory performance gap. The improvement rate of microprocessor is 60%/year, while the memory system performance has been enhancing at less than 10%/year, from 1980 to 2000 [1]. The processor-memory performance gap grows exponentially. After 2000, it still has been increasing, and it will continually increase in the next few years. The continuous growing gap between processor and memory speeds is an important drawback in the overall computer performance. For this reason, improving the memory performance is significant.

In the computer memory hierarchy, there are caches between CPU and main memory like Figure 1. The cache memory consists of the Static Random Access Memory (SRAM) which is faster than Dynamic Random Access Memory (DRAM) that make the main memory. This memory is typically integrated directly with the CPU chip or placed on a separate chip that has buses to interconnect with the CPU. The cache memory acts as a high speed buffer between CPU and main memory. It stored all the recent instructions and temporary data that the CPU may frequently require. Because of these role of the cache memory, the processing speed is increased by making necessary data and instructions available in the cache.

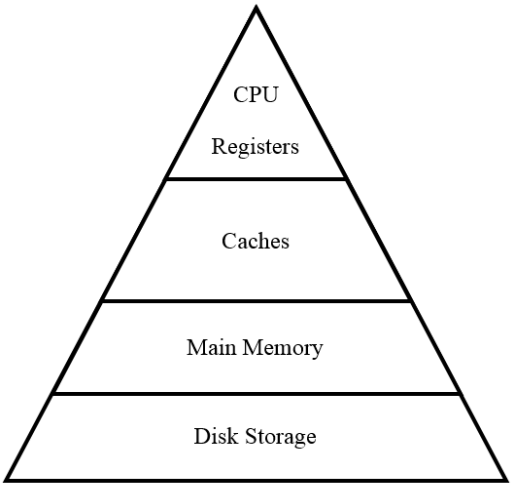

Figure 1. Computer memory hierarchy

Since the importance of cache in memory system, the improvement performance of cache can make the memory system performance enhance.

In the cache, there are parameters that classify the kinds of cache; capacity, associativity and line size. The capacity is overall size of the cache. The associativity is the number of space that can store data which have same tag address. The line size, like its name, is the size of line. The line is so called block, and it is an unit of data size that move into cache from main memory. In this paper, all together these, They are called configuration of cache. When the configuration of cache have different patterns, the performance of cache is also changed. To find out the relationship between cache and memory system, we studied the effects on the memory performance depending on the capacity, associativity and line size of cache [2]. Not only increasing the capacity of cache but also changing the associativity and line size of cache can improve the memory performance. Varing the configuration of cache shows more efficient performance enhancement rather than only enlarging the capacity of cache. Thus, for the improvement of memory performance, we designed the cache which optimizes the memory performance of target processor, not merely increasing the capacity.

In order to design the cache, the memory performances were evaluated when changing each parameter of the cache configuration. To compare the performance of conventional cache and proposed cache, these were assessed in processor performance and memory performance aspect respectively. For evaluation of the memory performance, Average Memory Access Time (AMAT) was selected as a criterion. Also, for the performance of processor, Cycles Per Instruction (CPI) was used; which is typically used for assessing the processor performance.

Table 1. The conventional configuration of cache on Cortex-A57

	L1I	L1D	L2
Capacity	48KB	32KB	512KB~2MB (configurable)
Associativity	3-way	2-way	16-way
Line size	64B	64B	64B
Replacement Policy	LRU	LRU	Random

L1I : Level-1 Instruction Cache
L1D : Level-1 Data Cache
L2 : Level-2 Unified Cache
LRU : Least Recently Used

2. Experimental Environment

2. 1 Experimental Instrument

The ARM Cortex-A57 was selected as a target processor because it is commonly used as a mobile processor [3]. Table 1 indicates the conventional configuration of cache on Cortex-A57. To assess the performance, we used the sim-outorder simulator of SimpleScalar3.0 and benchmarks from SPEC CPU2000 [4], [5]. SimpleScalar is a well-known simulator for evaluating the processor, and it has various input options of the processor configuration. The sim-outorder simulator execute the processor with regardless of the instruction orders. SPEC CPU2000 is considered as a standard tool of benchmarks, and it is used to evaluate the general-purpose processor. In the SPEC CPU2000, we selected four integer benchmarks and four floating-point benchmarks that have different characteristics in access patterns and memory footprints respectively. They are indicated in Table 2. The evaluation of performance was conducted via the AMAT and the CPI. These were calculated respectively as the average value of the eight benchmarks.

2. 2 Experimental Method

For cache design to optimize the memory performance, the AMAT was used as below [6]. It is assumed that two-level cache architecture.

$$\begin{aligned}\text{(Average memory access time)} &= (\text{Hit time of L1 cache}) + (\text{Miss rate of L1 cache}) \\ &\times [(\text{Hit time of L2 cache}) \\ &+ \{(\text{Miss rate of L2 cache}) \times (\text{Miss penalty of L2 cache})\}]\end{aligned}$$

In this equation, there are hit time, miss rate and miss penalty of cache. The hit time of cache means how long does it take to cache when cache hit situation. The miss rate of cache is the ratio of cache misses to total cache accesses. The miss penalty of cache means all times to access to memory when cache miss situation. According to this equation, the miss penalty of L2 cache and miss rate of L2 cache have the least influence on AMAT. The hit time of L1 cache has the most influence on AMAT in the same vein. Because of this, the order of priority to experiment was determined. First, the L1 cache configuration was defined, and then, the L2 cache configuration was defined.

Table 2. The selected benchmarks of SPEC CPU2000

Benchmark Name		Category
Integer Benchmarks	gcc	C programming Language Compiler
	parser	Word Processing
	vortex	Object-oriented Database
	twolf	Place and Route Simulator
Floating Point Benchmarks	galgel	Computational Fluid Dynamics
	ammp	Computational Chemistry
	apsi	Meteorology: Pollutant Distribution
	mesa	3D Computer Graphics and GPGPU

The experiment was conducted by varying each parameter of the cache configuration; the capacity, associativity, and line size of cache. The capacity is the prime parameter of cache configuration, so it was defined first. After than, the associativity and the line size of cache were defined. In summary of the experiment order, the first simulation was conducted when varying the L1 cache configuration in order of importance; the capacity, associativity, and line size of L1 cache. The next simulation was conducted, on condition of fixing the L1 cache configuration by proposed one, when varying the configuration of L2 cache in the same order as mentioned.

As previously stated, the AMAT was used to evaluate the memory performance, so the diverse AMATs of all simulations are compared. The lower AMAT shows higher performance, so its cache configuration is regarded as better than the others. As a result of that, there are optimal values of each parameter, and they became the components of the proposed cache configuration. The optimal value was selected by considering the saturation or the minimum AMAT in graph.

3. Simulation and Analysis

3. 1 Simulation Conditions

In the two-level cache architecture, the line size of L2 cache must be equal to or bigger than the line size of L1 cache. For this reason, the simulation was carried out in only this condition. In the first simulation, the line size of L1 cache was defined as 64B, so the next simulation that define the line size of L2 cache was conducted after 64B. As mentioned earlier, the AMAT and the CPI were calculated respectivley as the average value of the eight benchmarks. It means that the experiment in each case was conducted eight times with different eight benchmarks. In every case of simulations, the integer benchmarks have the lower AMAT than floatin-point benchmarks. The benchmarks which have the lowest AMAT are mostly the gcc of integer benchmarks and the mesa of floating-point benchmarks. The benchmark which has the highest AMAT is the ammp of floating-point benchmarks.

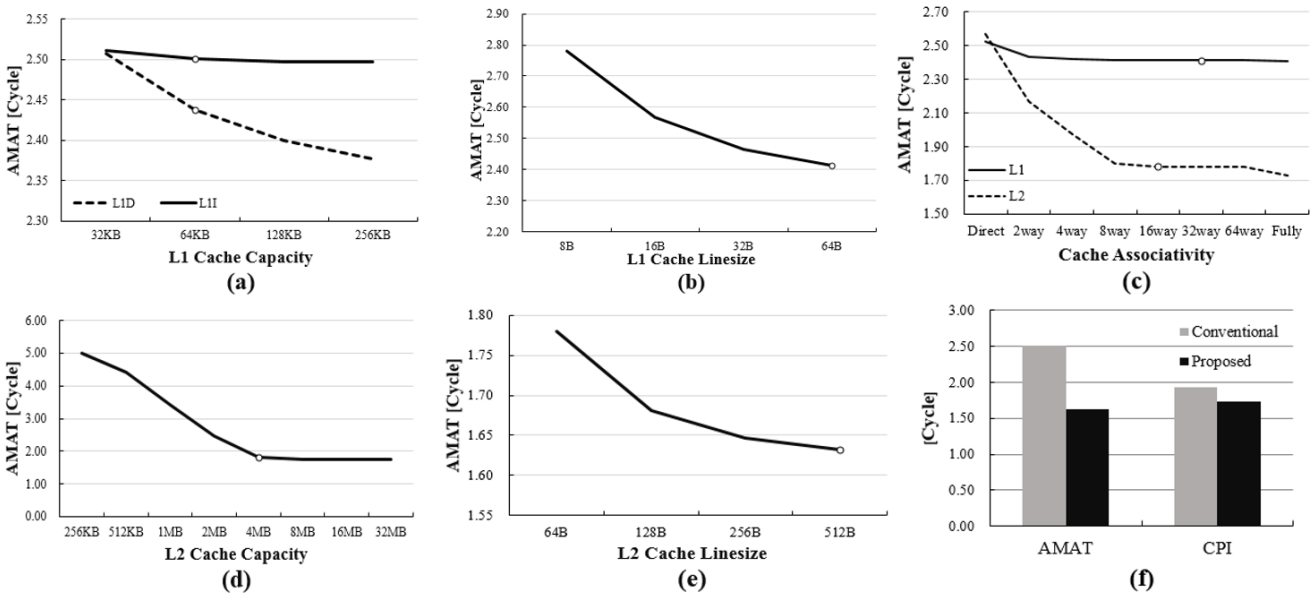

Figure 2. (a) AMAT values with capacity of L1 cache. (b) AMAT values with line size of L1 cache. (c) AMAT values with associativity of L1 and L2 caches. (d) AMAT values with capacity of L2 cache. (e) AMAT values with line size of L2 cache. (f) AMAT and CPI values with comparison of conventional cache and proposed cache.

3. 2 Simulation Analysis

The simulation results when changing the configuration of L1cache and L2 cache are shown in Figure 2.

Figure 2(a) shows the AMAT changes of the processor when L1 cache capacity is increased from 32KB to 256KB. Even though it is not easy to increase the capacity of cache, if the improvement of performance is regarded as better than sacrificing cost, we made a decision that the increased one is better in performance aspect. In this graph, after 64KB point, L1I capacity line is saturated, and its AMAT is 2.50 cycles. Thus, 64KB is selected as the capacity of L1I cache. The conventional value of L1I capacity is 48KB, and its AMAT is 2.51 cycles. By changing the capacity of L1I cache from 48KB to 64KB, the memory performance is increased by 0.71%. In L1D line, although it doesn't saturated, after 64KB point, the performance improvement rates are getting smaller. Thus, 64KB is selected as the capacity of L1D cache, and its AMAT is 2.44 cycles. The conventional value of L1D capacity is 32KB, and its AMAT is 2.51 cycles. By changing the capacity of L1D cache from 32KB to 64KB, the memory performance is increased by 6.92%.

Figure 2(b) shows the AMAT changes of the processor when L1 cache line size is increased from 8B to 64B. In this graph, the L1 line size line doesn't saturated, and the 64B point has the minimum value. Thus, the optimal value of L1 cache line size is 64B, and its AMAT is 2.41 cycles. The conventional value of L1 line size is also 64B, so there is no memory performance gain in this case.

Figure 2(c) shows the AMAT changes of the processor when L1 and L2 cache associativity are changed from direct-mapped to 64-way, and fully associative. In the L1 cache associativity graph, the L1 associativity line is saturated after 32-way point, but the 32-way and fully associative points have the same minimum value. The both points indicate 2.41 cycles. However, there is a disadvantage in cost aspect at fully associative due to complex hardware. In order to make the fully associative cache, the comparators are required because there is no tag addresses in fully associative architecture. To store or load data in this cache, the comparators compare all addresses in cache, so it is complex than the other associative cache. Due to this reason, if the both cases show the same memory performance, the 32-way associative is regarded as the better than fully associative in the cost aspect. Thus, the optimal value of L1 cache associativity is 32-way. The conventional value of L1 associativity is 2-way, and its AMAT is 2.43 cycles. By changing the associativity of L1 cache from 2-way to 32-way, the memory performance is increased by 0.82%. In the L2 cache associativity graph, the line is saturated after 16-way associative point. Thus, it can be the optimal value, and its AMAT is 1.78 cycles. The conventional value of L2 associativity is also 16-way, so there is no memory performance gain in this case.

Figure 2(d) shows the AMAT changes of the processor when L2 cache capacity is increased from 256KB to 32MB. In the L2 cache capacity graph, the line is saturated after 4MB point. According to this, the optimal value of L1 cache capacity is 4MB, and its AMAT is 1.82 cycles. The conventional one is 2MB at the most, and its AMAT is 2.46 cycles. Therefore, the memory performance is increased by 26.01% by changing the capacity of L2 cache from 2MB to 4MB.

Figure 2(e) shows the AMAT changes of the processor when L2 cache line size is increased from 64B to 512B. According to the above mentioned condition, the simulation was conducted after 64B; the proposed L1 line size. In the L2 cache line size graph, the L2 line size line doesn't saturated, and the 512B point has the minimum value. Thus,

the optimal value of L2 cache line size is 512B, and its AMAT is 1.63 cycles. The conventional one is 64B, and its AMAT is 1.78 cycles. By changing the line size of L2 cache from 64B to 512B, the memory performance is increased by 8.43%.

Figure 2(f) shows the average AMAT and CPI of the processor when using the conventional cache configuration and the proposed cache configuration. The conventional cache configuration is shown in Table 1, and the proposed cache configuration is 64KB capacity, 32-way associative, and 64B line size of L1 cache and 4MB capacity, 16-way associative, and 512B line size of L2 cache. In the AMAT graph, the conventional one is 2.51 cycles, but the proposed one is 1.63 cycles. Thus, the memory performance gain is 34.93%. In the CPI graph, the conventional one is 1.93 cycles, but the proposed one is 1.73 cycles. Therefore, the processor performance gain is 10.05%. Consquently, the cache design that optimize the memory performance can increase not only memory performance but also processor performance.

4. Conclusion

In modern mobile devices, the performance of processor has been rapidly getting higher, but the enhancement speed of memory performance is slow. Because of the different directions in processor and memory field, the processor-memory performance gap has been getting bigger. This gap between processor and memory speeds is the prime obstacle in the overall computer performance, so improving the memory performance is important. In the computer memory hierarchy, there are caches between CPU and main memory. The cache acts like a high speed buffer between CPU and main memory. Because of these role of the cache, the processing speed is increased by making necessary data and instructions available in the cache. Like this, the cache performs the significant role in memory system, so the improvement performance of cache can lead the enhancement performance of memory system. Thus, to improve the memory performance, we designed the cache that optimizes the memory performance of target processor.

To design the cache, a study of the memory performance enhancement was conducted when varying the cache configuration; the capacity, associativity and line size. The proposed cache configuration was found out by comparing the AMATs of all cases. The proposed cache configuration is 64KB capacity, 32-way associative, and 64B line size of L1 cache and 4MB capacity, 16-way associative, and 512B line size of L2 cache. It shows efficient improvement in both AMAT and CPI by 34.93% and 10.05% respectively. In short, the proposed design improve the performance of memory and processor. Therefore, in the mobile environment, the proposed one is more optimal than the conventional one.

In further work, It will be studied that the performance improvement of mobile processor when considering power consumption. The power consumption depends on the capacity of cache, so it is expected to get different results from this paper.

Acknowledgment

This research was supported by the MSIP(Ministry of Science, ICT & Future Planning), korea in the ICT R&D Program 2015 (2014-050-004-002).

References

[1] Carlos Carvalho, "The Gap betwwen Proceossor and Memory Speeds", *International Congress and Conventional Association*, pp.27-34, January 2002.

[2] Ma Hai-feng, Yao Nian-min and Fan Hong-bo, "Cache Performance Simulation and Analysis under SimpleScalar Platform", *International Conference on New Trends in Information and Service Science*, pp.607-612, July 2009.

[3] ARM Cortex-A57, http://www.arm.com/products/processors/cortex-a/cortex-a57-processor.php

[4] D.C. Burger and Todd M. Austin, "The SimpleScalar Tool Set, Version 2.0", *UW Madison Computer Science Technical Report #1342*, June 1997.

[5] Standard Performance Evaluation Corporation (SPEC) website, http://www.spec.org/

[6] David A. Patterson and John.L Hennessy, *Computer Organization and Design*, 5th Editon, ELSEVIER, 2014.