한국산학기술학회논문지
Vol. 9, No. 4, pp. 945-951, 2008

High Performance Data Cache Memory Architecture

Hong-Sik Kim[1] and Cheong-Ghil Kim[2*]

고성능 데이터 캐시 메모리 구조

김홍식[1], 김정길[2*]

Abstract In this paper, a new high performance data cache scheme that improves exploitation of both the spatial and temporal locality is proposed. The proposed data cache consists of a hardware prefetch unit and two sub-caches such as a direct-mapped (DM) cache with a large block size and a fully associative buffer with a small block size. Spatial locality is exploited by fetching and storing large blocks into a direct mapped cache, and is enhanced by prefetching a neighboring block when a DM cache hit occurs. Temporal locality is exploited by storing small blocks from the DM cache in the fully associative buffer according to their activity in the DM cache when they are replaced. Experimental results on Spec2000 programs show that the proposed scheme can reduce the average miss ratio by 12.53%~23.62% and the AMAT by 14.67%~18.60% compared to the previous schemes such as direct mapped cache, 4-way set associative cache and SMI(selective mode intelligent) cache[8].

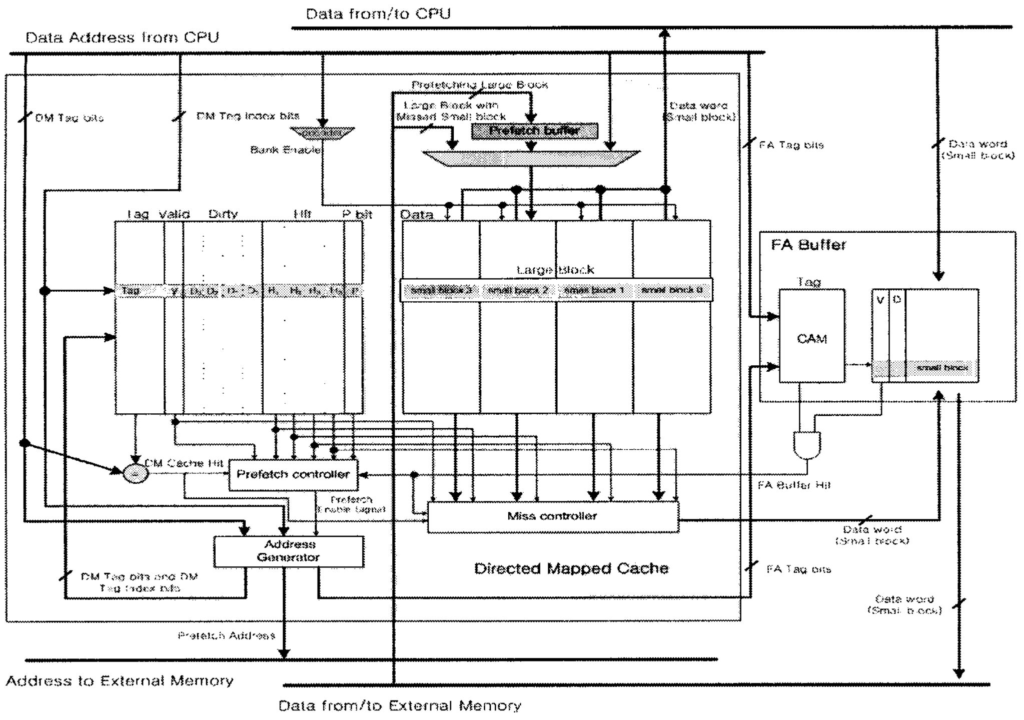
Key Words : data cache, temporal locality, spatial locality, and prefetch, AMAT(average memory access time).

요 약 공간적 지역성(spatial locality) 및 시간적 지역성(temporal locality)을 동시에 향상시킬 수 있는 새로운 고성능 데이터 캐시 구조를 제안한다. 제안된 캐시 메모리는 하드웨어 프리패치 유닛과 큰 블록 크기를 갖는 직접사상(DM: direct mapped) 캐시와 작은 블록 크기를 갖는 완전 사상(FA: fully associative) 캐시의 하위 캐시 유닛으로 구성된다. 공간적 지역성은 큰 블록 데이터를 패치하여 직접 사상 캐시에 저장함으로써 보장되며, DM 캐시 히트가 발생한 경우에 그 이웃 데이터 블록을 프리패치 함으로써 최적화 된다. 시간적 지역성은 작은 블록 데이터가 DM 캐시로부터 제거 될때 그 블록의 과거 기록에 따라서 중요한 데이터는 완전사상 캐시에 저장함으로써 보장된다. Spec2000 벤치마크 프로그램에 대한 실험 결과에 의하면 제안된 캐시 구조는 비슷한 크기의 직접사상 캐쉬, 4웨이 연관사상(4 way set associative cache) 및 SMI(selective-mode intelligent cache) 캐쉬 [8]등의 기존의 구조에 비해서 미스 비율(miss rate)을 평균적으로 12.53~23.62% 그리고 AMAT(average memory access time)를 평균적으로 14.67~18.60% 줄일 수 있음을 증명하였다.

1. Introduction

Cache memories can improve overall system performance since they are very effective in reducing the average memory access time (AMAT) by exploiting two types of localities - temporal and spatial - inherent in the reference stream of a typical application program. Temporal locality is due to the property that recently referenced data will be referenced again in the near future with high probability and spatial locality is due to the tendency that adjacent memory locations are referenced close together in time. However, many application programs usually have characteristics that may degrade the cache performance, such as non-unit stride property and larger vector length than the cache size [1]. The non-unit stride property of the access vector is that some of the vectors accessed by numerical application programs have a stride unequal to one, so that the spatial locality

[1]Dept. of Electrical & Electronic Engineering, Yonsei University [2]Dept. of Computer Science, Namseoul University
[*]Corresponding author : Cheong-Ghil Kim(cgkim@nsu.ac.kr)
Received July 1, 2008 Revised August 10, 2008 Accepted August 11, 2008



[Fig. 1] Proposed Data Cache Scheme

of a cache system cannot be fully exploited. The stride means the difference in address between two consecutive accesses made by an instruction. In addition to the limitation on exploitation of the spatial locality, utilization of the temporal locality can be degraded when the vector length is larger than the cache size since such a larger vector may sweep itself out. In order to address those problems, many researches have been proposed [2-14].

In this paper, a new data cache structure that improves exploitation of both the spatial and temporal locality is proposed. The proposed data cache consists of three parts - a direct-mapped cache with a large block size, a fully associative buffer with a small block size, and a hardware prefetching unit. In order to exploit both the spatial and the temporal locality, two different block sizes are used, such as a small block size to exploit temporal locality and a large block size, which is a multiple of the small block size, to exploit spatial locality. Spatial locality is exploited by fetching a large sized data block into the direct mapped cache, and is further enhanced with a hardware-based prefetching mechanism. Temporal locality is improved by selectively storing data blocks with a high probability of a repeated reference into the fully associative buffer when the data block is evicted from the direct mapped cache as a result of replacement. The proposed scheme shows that the total number of prefetch operation is reduced by 25.72% on average, compared to the prefetch scheme in [8]. In addition, experimental results show that the average miss rate and the average AMAT of the proposed scheme with 8KB direct mapped cache and 1KB temporal buffer can be reduced by around 12.53%~23.62% and by 14.67%~18.60%, respectively, compared to various cache schemes with the similar hardware area.

2. Proposed Cache Memory Scheme

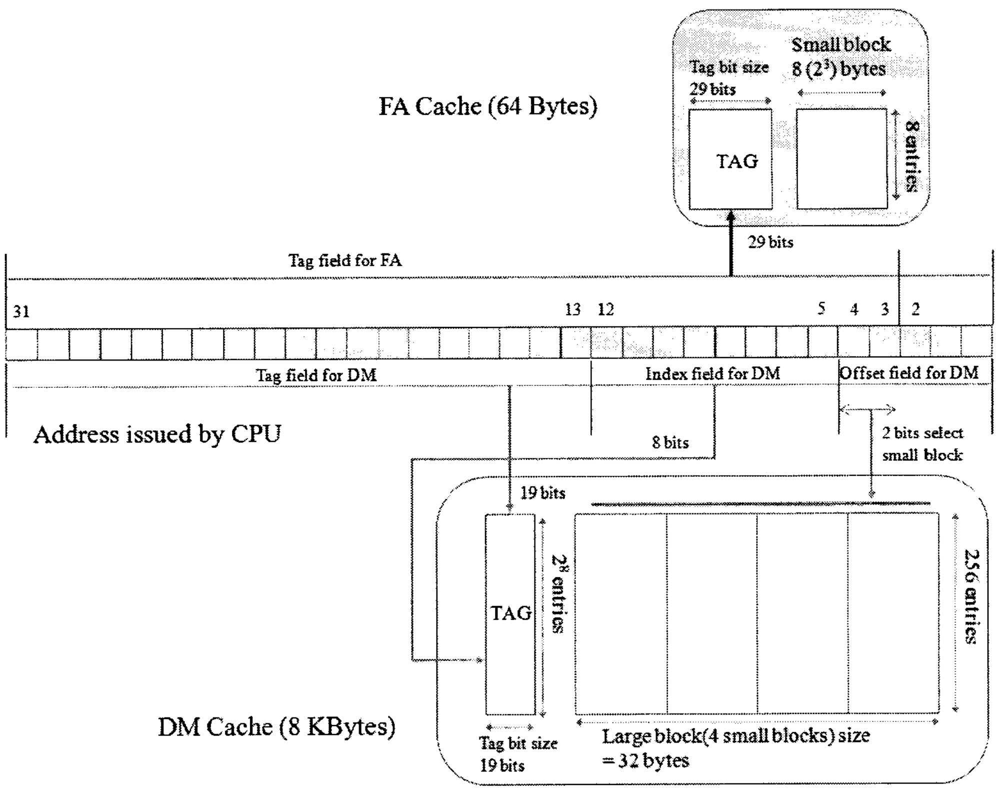
The proposed cache structure is shown in Fig. 1. The proposed data caching scheme consists of three parts, such as a prefetch control unit, a direct mapped cache, and a fully associative buffer. The direct-mapped cache is the main cache and its organization is similar to a traditional direct-mapped cache, but for its unique organization of data entries and additional flag registers for each data entry. Each data entry of the direct mapped cache is divided into several banks, each of which is the size of one block in the fully associative buffer. The power consumption can be reduced by using the most significant two bits of the large block offset to activate just one of the banks in the direct mapped cache. When the CPU issues a memory reference, the direct-mapped cache and the fully associative buffer are searched in parallel. If a reference hits in the direct-mapped cache, but misses in the fully associative buffer, its corresponding small block is simply fetched from the direct mapped cache and the hit bit register for the small block is set. The prefetch controller generates a prefetch enable signal when a large block in the direct mapped cache is accessed with multiple hit bit register set.

The prefetch controller generates a prefetch enable signal when a large block in the direct mapped cache is accessed with multiple hit bit register set. If a prefetch enable signal is issued after i-th large block hits, both the tags of the direct mapped cache and the tags of the fully associative buffer are searched to detect whether the (i+1)-th large block is already present. The address for (i+1)-th large block is generated by the address generator. One cycle search overhead occurs, but its overhead is negligible because prefetching is initiated in response to only about 2.67%~3.17% of the total number of addresse references on average, according to the experiments of the proposed scheme with various setassociative buffer sizes on SPEC2000 benchmarks. If the (i+1)-th large block does not exist in either the direct mapped cache nor the fully associative buffer, then the (i+1)-th large block is fetched into a prefetch buffer and the P bit in the i-th large block is set to prevent the prefetch control unit from generating further prefetches for the i-th large block. The number of prefetch buffer entries is assumed to be one. In [8], tags of only the fully associative buffer are searched when prefetch signal is set, so that the incoherency copies may occur. Since the proposed scheme searches both the tags of all the caches blocks, such a pollution can be prevented. When a miss occurs in both of the direct-mapped cache and the fully associative buffer, the cache controller initiates miss handling in order to fetch the missing large block into the direct mapped cache and moves small blocks evicted from the direct mapped cache into the fully associative buffer. Also, the large block data content in the prefetch buffer is moved into the direct mapped cache. The time of data transfer from the prefetch buffer to the cache is hidden because much more cycles are required for miss handling than the transfer cycles. In this paper, 19 clock cycles are assumed for the miss handling overhead. The missed block, however, may exist in the prefetch buffer. Therefore, when the block in the prefetch buffer is transferred into the direct mapped cache, the tag value in the prefetch buffer is simultaneously compared with the miss address. If the comparator shows a match, the data content in the prefetch buffer is delivered not only to the spatial buffer and to the CPU at the same time. The ongoing miss handling is canceled by the cache controller.

3. Operational Model

The process for the management of the proposed caching scheme is described in more detail. For simplicity, we use as an example an 8KB direct-mapped cache with a large block size of 32-bytes and a 64 byte spatial buffer with a small block size of 8-bytes without loss of generality. The address map for each sub-cache blocks of the proposed scheme is shown in Fig 2. In addition, let assume that a 32-bit memory address, such as FFFFFF00, is generated by the CPU. In case of the direct-mapped cache, the tag field is 19-bits (A: 7FFFF), the index field is 8-bits (B:F8), and the offset field is 5-bits. The most significant two bits of the offset field, the small block offset bits, are used to select one of the four banks in the direct mapped cache. In this example, since the value of the small block offset bits is 2-bits (C: 00), the first small block (small block0) is selected. In the fully associative buffer, the tag field is 29-bits (D: 1FFFFFF0) and the offset field is 3-bits.

When a direct mapped cache hit occurs or the large block data corresponding the referenced data address (whose tag value A and index bit value is B) exists in the direct mapped cache, the small block data whose small block offset is C (00) are sent to the CPU and the hit bit for referenced small block (small block0) is set to identify it as a referenced one. In this case, if P bit for the large block entry is in the reset state, a prefetch operation is



[Fig. 2] An Example of address map for two sub-cache memories of the proposed scheme

triggered if more than one hit bits for the corresponding large block are already set. At the same time, the tags of both the direct mapped cache and the fully associative buffer are searched for the prefetch address which is generated by the address generator in order to check whether it already exists in the cache memories. The address generator generates the prefetch address by incrementing the index field of the referenced address. In this example, the prefetch address shall be FFFFFF20. Therefore, the tag value and index value of the prefetch address for the direct mapped cache search are 7FFFF and F9, respectively. In addition, the tag address of the prefetch address for the fully associative buffer is 1FFFFFF4. If the data corresponding to the prefetch address does not exit in the cache memories, the prefetch operation is continued and the P bit for the corresponding large block is set in order to prevent repetition of the prefetch operation for the referenced large block. Otherwise, the prefetch operation is cancelled.

If a read access to the fully associative buffer hits, then the requested data block is transferred to the CPU. If a write access to the fully associative buffer is a hit, then the write operation is performed and the dirty bit is set.

When a miss occurs in both caches, a large block including the missed small block is brought into the direct mapped cache from the next level memory or data corresponding to the tag value of the large block (A) are fetched. If the valid bit for the large block is set, the fetched data will replace the previous data in the corresponding direct mapped cache entry so that some small blocks with hit bit register set shall be transferred into the fully associative buffer in order to exploit the temporal locality of the small blocks which have been previously hit. If the hit bit, Hi, of the small block i ($0 \leq i \leq 3$) is set, the index bit field (B:F8) and the two bit small bit offset field (C:i) are attached to the tag value of the direct mapped cache (A) by the address generator. The two-bit small block offsets corresponding to the four small blocks are '00', '01', '10', and '11', separately. Therefore the tag address of (ABC) is generated for the

[Table 1] Experimental results on SPEC 2000 benchmark programs for various cache schemes

Caches	Capacity (byte)	Performance		Aear (mm2)
		average Miss rate	average AMAT	
32 KB Direct Mapped Cache	32K	9.87853	2.876925	0.494700
64 KB Direct Mapped Cache	64K	8.346137	2.585766	0.970413
16KB 4-Way Set Associative Cache	16K	8.736124	2.659864	0.349832
16K Victim Cache with 1KB Buffer	17K	8.320473	2.580890	0.358371
SMI with 8KB DM and 1KB Buffer	9K	6.855545	2.538740	0.362667
SMI with 8KB DM and 2KB Buffer	10K	6.184494	2.461915	0.478407
Proposed with 8KB DM and 64B Buffer	8K	6.354995	2.283201	0.231021
Proposed with 8KB DM and 1KB Buffer	9K	5.409480	2.100770	0.382077

fully associative buffer by the address generator. For example, if H0 has logic value of '1', then the tag address for corresponding fully associative buffer entry shall be 1FFFFFE0.

Since any modified or referenced small block is transmitted to the fully associative buffer before its corresponding large block is replaced, cache write-back operation does occur only from the fully associative buffer. In case of the proposed caching scheme, the write back operation is performed only for the 8-byte small blocks with hit bit register set so that the write traffic into memory can be possibly reduced.

4. Experimental Results

In this section, the simulation environment, performance metrics, and area overhead are explained in detail. Simple Scalar/ARM processor simulator [15] is used to collect runtime information on SPEC2000 benchmark programs. Only data references are collected and used for the simulation. For the comparisons with other caches in terms of performance and area overhead, the direct-mapped cache (DM), the victim cache (VT), the 4-way set associative cache (4W) and a SMI cache are used.

The performance of the proposed caching scheme depends on the number of hit bits for prefetching and the size of the fully associative buffer. The miss rate is calculated by dividing the number of references that are not found in the cache with the number of the total memory references. Generally, the more meaningful measure to evaluate the performance of any given memory-hierarchy is the average memory access time [18]. So in this paper, miss rate and AMAT are used for the evaluation of the proposed cache memory.

Table 1 describes the comparison of the propsed scheme with the other cache memories including the direct mapped caches, the victim caches, 4-way set associative cache and SMI cache in terms of hardware area and cache performance. For the area analysis, the CACTI-4.2 simulator is used under 0.09 μm technology and 1.2 V supply voltage. According to the experimental results, the proposed scheme with 1KB buffer results in similar size as 8KB SMI cache with 1KB buffer, 16KB 4-way set associative cache, and 16KB victim cache with 1KB buffer, although it shows much higher performance gains such as smaller miss rate by (12.53~23.62%) and smaller AMAT by (14.67~18.60%) compared to those previous caching schemes. In case of the proposed scheme with 64 byte buffer, the experimental results shows that the proposed scheme can results in smaller average miss rate by (12.53~23.62%) and the smaller average AMAT by (14.67~18.60%) even with much smaller hardware area compared to all the previous schemes. Considering the performance improvement and the area overhead presented in Table 1, the proposed cache scheme can guarantee a higher performance with less hardware area compared to the previous cache systems.

5. Conclusion

A new high performance data cache structure that improves the utilization of both the spatial and temporal locality is proposed. Spatial locality is exploited by fetching and storing large blocks into a direct mapped

cache, and is further enhanced by prefetching a neighboring block when a direct mapped cache hit occurs. Temporal locality is exploited by storing small blocks from the direct-mapped cache in the fully associative temporal buffer according to their activity in the direct mapped cache when they are evicted from the direct mapped cache as a result of replacement. Experimental result based on Spec2000 benchmark programs shows that the proposed scheme can reduce the average miss ratio by 12.53%~23.62% and the average memory access time by 14.67%~18.60% compared to the previous cache schemes with the similar hardware area.

References

[1] B. Juurlink, "Unified Dual Data Caches," Proceedings of the Euromicro Symposium on Digital System Design, 2003, pp. 33-40

[2] Norman P. Jouppi, "Improving Direct-Mapped Cache Performance by the Addition of a Small Fully Associative Cache and Prefetch Buffers," Proceedings of 17th ISCA, May. 1990, pp. 364-373.

[3] D. Stiliadis and A. Varma, "Selective Victim Caching: A Method to Improve the Performance of Direct Mapped Cache," IEEE Transactions on Computers, Vol 46, No. 5, May 1997, pp. 603-610.

[4] A. Gonzalez, C. Aliagas, and M. Valero, "A Data Cache with Multiple Caching Strategies Tuned to Different Types of Locality," Proceedings of International Conference on Supercomputing, 1995, pp. 338-347.

[5] B. Juurlink, "Unified Dual Data Caches," Proc. The Euromicro Symposium on Digital System Design, 2003. pp. 33-40.

[6] V. Milutinovic, M. Tomasevic, B. Markovic, and M. Tremblay, "A New Cache Architecture Concept: The Split Temporal/Spatial Cache," 8th Mediterranean Electro-technical Conference, vol.2, 1996, pp. 1108 - 1111.

[7] J. H. Lee, J. S. Lee, and S. D. Kim, "A new cache architecture based on temporal and spatial locality," Journal of System Architecture, Vol. 46, Sep. 2000, pp. 1451 - 1467.

[8] J. H. Lee, S.-W. Jeong, S. D. Kim, and C.C.Weems, "An Intelligent Cache System with Hardware Prefetching for High Performance," IEEE Transactions on Computers, Vol. 52, No. 5, 2003, pp. 607 - 616.

[9] T. Mowry, M. S. Lam, and A. Gupta, "Design and evaluation of a compiler algorithm for prefetching," Proceedomgs of 5th International Conference on Architectural Support for programming Languages and Operating Systems, 1992, pp. 62-73.

[10] A.K. Porterfield. "Software Methods for Improvement of Cache Performance on Supercomputer Application," PhD dissertation, Rice Univ. 1989.

[11] W. Y. Chen, S. A. Mahlke, P. P. Chang, and W. M. Hwu, "Data Access Microarchitectures for Superscalar Processors with Compiler-assisted Data Prefetching," Proceedings of 24th Annual Workshop on Microprogramming and Microarchitectures, 1991.

[12] A.J. Smith, "Cache Memories," Computing Surveys, vol. 14, no. 3, 1982, pp. 473-530.

[13] J.D. Gindele, "Buffer Block Prefetching Method," IBM Technical Disclosure Bulletin, Vol. 20, no. 2, 1977, pp. 696 - 697

[14] D. Zucker, M.J. Flynn, and R. Lee, "A Comparison of Hardware Prefetcing Techniques for Multimedia Benchmark," Proceedings of IEEE Multimedia, 1996, pp. 236-244.

[15] D. Burger and T.M. Austin, The SimpleScalar tool set, version 2.0, Technical Report TR-97-1342, University of Wisconsin-Madison, 1997.

Hong-Sik Kim [Regular Member]

- 2004. 8 : Dept. of Electrical and Electronic Engineering, Yonsei Univ. (Ph.D.)
- 2004. 11 : Post Doctoral Research Fellow at Bradley Dept. of Electrical and Computer Engineering, Virginia Tech.
- 2006. 1 : Senior Engineer at System LSI Group in Samsung Electronics Co.
- 2007. 4 : Research Professor at the Dept. of Electrical & Electronic Engineering, Yonsei University.

<Area of Interest>
Design for Testability, Built-in Self Test

Cheong-Ghil Kim [Regular Member]

• 2003. 8 : Dept. of Computer Science, Yonsei Univ. (Master of Eng.)
• 2006. 8 : Dept. of Computer Science, Yonsei Univ. (Ph.D.)
• 2006. 9 : Post Doctoral Researcher and Research Professorat the Dept. of Computer Science, Yonsei Univ.
• 2008. 3 : Professor at the Dept. of Computer Science, Namseoul Univ.

<Area of Interest>
Computer Architecture, Multimedia Embedded Systems