A Virtual-Physical On-Chip Cache for Shared Memory Multiprocessors

Dongwook Kim and Joonwon Lee

Computer Architecture Lab., Computer Science Department,
Korea Advanced Institute of Science and Technology,
373-1 Kusung-dong Yusung-ku Taejon 305-701, South Korea.

1 Introduction

Two-level caches are popular in high performance multiprocessors because the cache at the first-level provides data very fast while the one at the second-level enables high hit ratios[1, 2]. The cache at the first-level, L1, is small and close to the processor usually on-chip for the fast access time, while the cache at the second-level, L2, is large in its capacity. Since the virtual cache makes the cache access time fast, it is adequate for the design of the L1 cache. When a virtual cache is used as the L1 cache, the synonym problem can be handled properly if the inclusion property is enforced[2]. An inclusion property means that all the data in the lower level cache is always in the higher level cache. However, the cache coherence problem in shared memory multiprocessors imposes another difficulty in designing two-level cache. Since physical addresses are used in the L2 cache, it necessitates the use of pointers between two levels to keep track of the mappings between virtual cache and physical cache for the coherence[5]. This limitation makes a write policy of the virtual cache more complex than that of the physical cache. This complexity will reduce the potential of the virtual cache to match the timing requirements of fast processors.

In this paper, we propose a novel multi-level cache architecture, where the L1 cache is decomposed into on-chip two-level virtual-physical caches.

2 The Architecture of a Virtual-Physical On-chip Cache

The main idea of our scheme is to overcome the drawbacks of a virtual cache by classifying memory requests into instruction, private data and shared data, and then storing them in separate caches according to the classified state. A simple hardware block diagram of a virtual-physical on-chip two-level cache is shown in Fig. 1(a).

The L1 virtual caches are accessed via virtual addresses which are also forwarded to the TLB at the shared data cache, and thus address translation can proceed concurrently with the access to the virtual caches. If there is a valid hit in the virtual caches, the translation and the access to the physical cache is aborted. Otherwise, the L1 physical cache is accessed by the translated address. The L2 cache would be accessed when the L1 caches cannot service the memory requests from a processor. The private data cache exclusively loads private data,

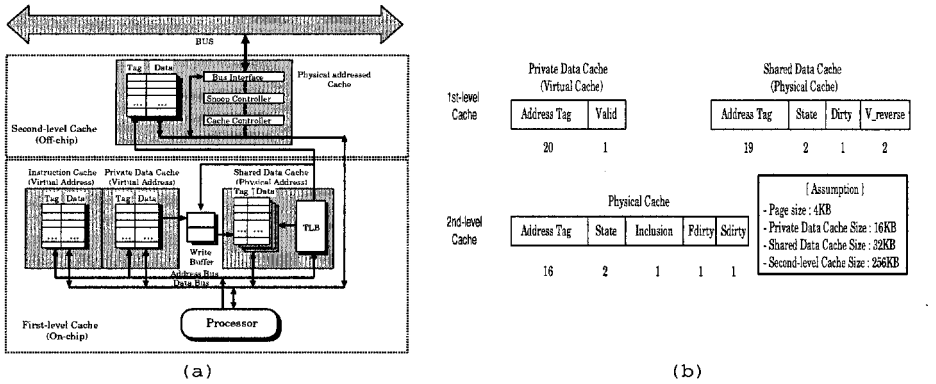

Fig. 1. Virtual-physical on-chip cache architecture

while the shared data cache can load both private and shared data. According to the state of the fetched cache line, it can be loaded in either cache. If the state is exclusive, the fetched cache line can be loaded in the private data cache and the shared data cache. Otherwise, it can be loaded in the shared data cache. The write buffer between the private data cache and the shared data cache can hide the memory latencies due to subsequent writes.

Fig. 1(b) shows the tags and control bits which are associated with each cache line. In the shared data cache, each tag entry contains a tag, state bits, a dirty bit and a v_reverse bits. The state bits indicate status of a line and it would be used for cache coherence control with other physical caches and classifying cache data into private or shared. A dirty bit indicates the cache line is updated. The v_reverse bits are reverse address pointer and are used to detect synonyms. The L2 cache is responsible for shielding cache coherence interference from other processors. The fdirty bit indicates if the corresponding cache line in the L1 cache is updated or not. It is used for the cache coherence protocol for multiprocessors. The sdirty bit indicates that the corresponding cache line in the L2 cache is updated. We assume that our scheme uses an invalidation cache coherence protocol for simplicity, although it will also work for other protocols as well.

3 Performance Evaluation

To study the performance implications, we use an event-driven simulator which runs on the MINT[4]. The test programs used in this study come from the SPLASH suite[3]. Those are FFT, Barnes, MP3D, Pthor and LU ,which are implemented for snoopy cache bus-based multiprocessors. We measure two performance metrics in the simulation : the cache coherence traffic and the average memory access time. An important advantage of the hybrid V-P cache is that the private data cache is not affected by irrelevant cache coherence interference. A processor does not stall to access the private data cache when a coherence

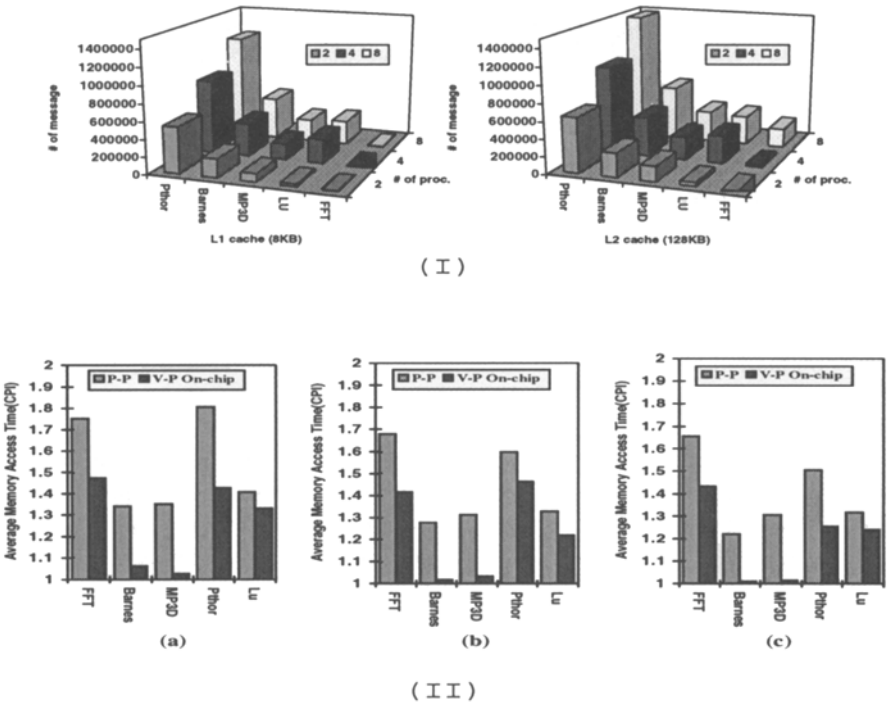

Fig. 2. (I) Number of coherence messages to the L1 cache(16K/128K) (II) Average memory access time(cycles) versus cache sizes, (a)16K/128K, (b)32K/256K, (c)64K/512K

message is issued from the bus to the L1 cache because the message affects only the shared data cache. Thus, a processor stall times can be significantly reduced by the private data cache. The results in Fig. 2(I) show that the inclusion property is essential for a two-level cache to reduce the number of coherence messages which are issued to the L1 cache. We believe that the performance gains by the private data cache will be more prominent as the number of processors increases. This is due to the fact that more bus coherence requests will be generated from a large number of processors, and the L1 cache will be disrupted more often and a processor stall time will be increased.

A higher hit ratio does not necessarily mean faster memory access time since cache access time for a cache hit is more important when the hit ratio is very high. The V-P on-chip cache provides fast cache access time for a cache hit at the private data cache. Therefore, average memory access time can be decreased in proportion to the hit ratios of the private data cache. Fig. 2(II)(a), Fig. 2(II)(b) and Fig. 2(II)(c) show average memory access times for various cache sizes. The V-P on-chip cache outperforms the P-P cache for all the tested applications.

There is a wide performance gap between the V-P on-chip cache and the P-P cache for the Barnes and Pthor, since a large proportion of data references of those applications are private data. Meanwhile, LU shows slight performance gap due to a small proportion of private data.

4 Conclusions

In this paper, we propose a new V-P on-chip cache scheme for multiprocessors. The key idea of the new scheme is that virtual caches and physical cache are incorporated at the same level as L1 caches, and the fetched data from a memory should be loaded in an exclusive cache according to whether it is shared data or private data. If a newly fetched data is a private data, it can be stored in the private data cache accessed by virtual addresses. Otherwise, it should be stored only in the physical cache called shared data cache. It is easy to optimize the circuits of a private data cache, since the logics for managing synonyms and cache coherence are not needed. It is responsible for the shared data cache to shield the private data cache either from synonyms or from irrelevant cache coherence interferences. In order to achieve this purpose, an inclusion property should be imposed to the shared data cache as like an L2 cache. Simulation results show that the V-P on-chip cache outperforms other cache structures.

Recent VLSI technology enables a very fast cycle time for a processor, and thus it becomes more difficult for the memory system to meet such a fast cycle time. In a shared memory multiprocessor, this problem is even more aggravated due to the cache coherence problem. We believe that our proposed cache scheme addresses such issues appropriately, and thus it will be a promising soultion for memory system of high performance multiprocessors.

References

1. C. Anderson. Improving performance of bus-based multiprocessors. *PhD thesis, University of Washington*, 1995.
2. J. L. Baer and W. H. Wang. On the inclusion property for multi-level cache hierarchies. *In Proceedings of the 15th Annual International Symposium on Computer Architecture*, pages 73–80, 1988.
3. J. P. Singh, W. Weber, and A. Gupta. SPLASH:Stanford Parallel Applications for Shared Memory. *Computer Architecture News*, 20(1):5–44, 1992.
4. J. E. Veenstra and R. J. Fowler. MINT tutorial and user manual. *TR452, Computer Science Department, Univ. of Rochester*, 1994.
5. W. H. Wang, J. L. Baer, and H. M. Levy. Organization and performance of a two level virtual real cache hierarchy. *In Proceedings of the 16th Annual International Symposium on Computer Architecture*, pages 140–148, 1989.