Dynamic, Tagless Cache Coherence Architecture in Chip Multiprocessor

Mian Lou and Jianqing Xiao

Abstract Chip multiprocessor (CMP) systems rely on a cache coherence protocol to maintain data coherence between local caches and main memory. The traditional protocols adopted are based either on data invalidation or on data update policies. However, these strategies do not consider the changes in the data access patterns at runtime. This paper explores a novel dynamic hybrid cache coherence protocol, with the combined use of the two typical protocols. To automatically adapt functioning mode of the protocol to application behavior, an efficient protocol algorithm is also presented. Moreover, by keeping a copy of all L1 tags, an original tagless structure that costs less area replaces the traditional full bit-vector directory. The simulation results show appreciable reductions in overall cache area and power consumption, with significant reduction in entire execution time.

Keywords Cache coherence · Dynamic · Tagless directory · Area · Power

1 Introduction

The huge number of transistors that are currently supplied in a single die has made major microprocessor vendors to shift toward multicore architectures in which several processor cores are integrated on a single chip. Chip multiprocessors (CMPs) have important advantages over very wide-issue out-of-order superscalar processors. In particular, they provide benefits such as low communication latency, a well-understood programming model, and a decentralized topology well suited for heterogeneous processing cores.

M. Lou (✉) · J. Xiao
Xi'an Microelectronics Technology Institute, Xi'an, China
e-mail: loumian2008@hotmail.com

Z. Wen and T. Li (eds.), *Foundations of Intelligent Systems*,
Advances in Intelligent Systems and Computing 277,
DOI: 10.1007/978-3-642-54924-3_19,

Most prevalent CMPs (for example, the IBM Power5 [1]) have a relatively small number of cores (2–8) connected through an on-chip shared bus or crossbar, each one with at least one level of local cache to alleviate the performance degradation caused by available bandwidth. This replication of L1 caches introduces the probability of data pollution if data are being shared among cores, since any modification of private data in one cache may leave other cores with stale versions which should no longer be considered valid. Generally speaking, two families of protocols are used. The main objective of the invalidation protocol is to transmit invalidation message to sharer caches that contain the modified data. At the opposite, under the control of the update protocol, the address and value of the modified block are sent to all other sharers. Chtioui et al. [2] have demonstrated that the use of a unique protocol (invalidation or update) does not take into account the patterns of data accesses executed by processors. Hence, it is necessary to explore the dynamic hybrid cache coherence protocol, which deals with increased cache accesses and related cache misses due to the running of more complex and irregular application set. Bournoutian and Orailoglu utilize a fine-grained dynamic approach with shift counter for each L1 line, to mitigate multicore processor power consumption. However, with the increment of the number of integrated cores, it is complex to realize a global state diagram to manage diverse coherence protocols possessed by each core [3]. In [2], a hybrid policy based on a completely hardware solution and using a full bit-vector directory can significantly reduce both cache misses and unnecessary updates. Although conventional directory bandwidth utilization scales more graciously, it introduces large structures that consume precious on-chip area and leakage power due to complicated state switch comparatively [4, 5].

In this paper, we investigate a new scheme by taking advantage of the two protocols, which can trace changes in the data access patterns at runtime and automatically switch to another more efficient protocol if necessary. Specifically, we propose a novel directory structure that obviously lightens the area overhead and further mitigates energy waste.

2 Hybrid Protocol Proposed

In [2], a hybrid protocol named ESIO is able to help in choosing the appropriate policy (validation or update) for a given shared memory block in a given application phase. As depicted in Fig. 1a, the Invalid (I) state shows that no valid data are stored in each private cache. The Exclusive (E) state denotes that the corresponding data are the only correct version within the localized caches and in the shared memory. The Shared (S) state implies that the valid data may also be located in other private caches [6]. The special state “O” (invalidated by other) has played an important role in the dynamic protocol. It distinguishes blocks that have not been yet loaded from blocks that have been invalidated by another host.

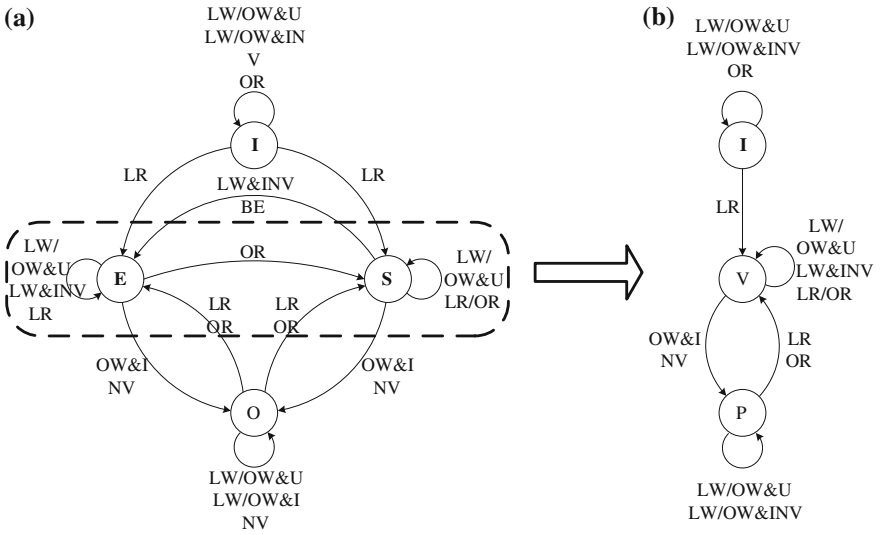

Fig. 1 **a** ESIO state diagram. **b** IVP state diagram

Table 1 Listing of abbreviations

Notation	Meaning
OR	Read by a remote processor
LR	Read by the local processor
OW	Write by a remote processor
LW	Write by the local processor
BE	Block ejection due to replacement
INV	Invalidation operation
U	Update operation

Note that this four-state (ESIO) protocol uses two bits per shared memory block. Besides, the definitions of notations on the arcs in Fig. 1 are listed in Table 1.

However, the cache coherence is mainly maintained by directory in the centralized shared memory. The motive of ESIO protocol is to identify the occurrence that the certain local cache block is invalidated by another processor. Thus, it is inessential to recognize that the source of "O" is "E" or "S" as well as the destination. Moreover, all of the data accesses sent by processors have to visit the coherence directory and further update the ESIO state according to the associated update or invalidation operation. Therefore, it is necessary to reduce the state migration among different states, leading to the significant dynamic power consumption of memory. By analyzing the detail function of each state in the above protocol, we can intelligently make modifications on the ESIO protocol in order to significantly reduce switch power overhead while minimally affecting execution performance. As demonstrated in Fig. 1b, the main contribution of our simplified IVP protocol is to combine the stale state "E" and "S" in ESIO as a single state

"V". The aim of the combination just denotes that there will be at least a valid replication in private cache. It could eliminate the unnecessary state transitions existing in the former protocol. Moreover, the state P represents the similar condition that the block is invalidated passively by another processor. The only condition that could cause state switch from P to another happened when the corresponding memory block receives a read request. The primary theory of this view is that the subsequent choice of coherence protocol depends on the write frequency for the associated data block between read operations. The detail description is demonstrated in the next section.

3 Protocol Implementation

The conventional implementations for the cache coherence mainly include two frameworks: snoop and directory. Unfortunately, snoop protocols rely on broadcasts that deteriorate bandwidth utilization and, therefore, power consumption in the interconnection, which has been constituted a significant fraction of the overall chip power [7, 8]. Directory protocols depend on a conceptually centralized structure to record duplication owners for each block. These reduce bandwidth requirements, but introduce large structures that consume precious on-chip area and leakage power. In this paper, we compare our proposal against a directory structure used in [2]. The latter one has a disadvantage in area and power which have a linear change according to the capacity of the shared memory.

3.1 Directory Structure

The proposed configuration of directory is abstractive as shown in Fig. 2, which makes original contribution as follows. For one thing, a duplication of all L1 tags integrated on the L2 cache is responsible for maintaining the coherence across all L1 caches, instead of a full bit-vector directory in [2]. The full bit-vector directory has the equivalent number of entries corresponding with the number of shared memory blocks; thus, it consumes large area on chip. However, because the total area overhead of all L1 tags is less than that of L2 tag, the proposed has a remarkable advantage to reduce area cost. On the other hand, the key to our solution is associated with each memory block a two-bit flag to determine which coherence protocol to use in the subsequent accesses. The benefit of this mechanism is the dynamic record of data access pattern. This feature could find the ideal balance between the improvement in execution performance and the reduction in memory power, along with the reasonable conversion between update and invalidation. Besides, the physical prototype of adopted directory is realized with SRAM instead of CAM used by Niagara 2 [9]. This decision is based on the fact that the CAM is so costly that is unsuitable to be employed far and wide.

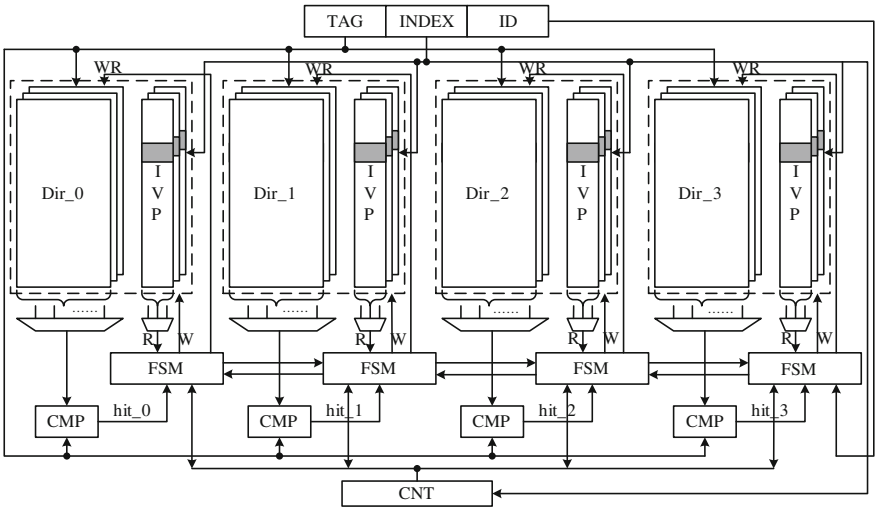

Fig. 2 The proposed configuration of directory

In practice, the flow of operation is processed as follows. Firstly, it should be parallel to search all entries existing in different directories depicted as the above structure. All the possible entries can be located with the index segment of the access address from L1 cache directly. Secondly, by comparing the content of selected entries with the tag segment of the former address, it is accurate to find out which processors have the correct duplications. At the same time, the associated IVP state can also be obtained. If there is a read operation, the corresponding protocol of memory block may turn into the opposite one, according to a following smart algorithm. In addition, the content with regard to IVP state shall be updated under the control of the algorithm.

3.2 Protocol Algorithm

For clarity, Fig. 3 demonstrates the above-mentioned algorithm. The algorithm takes into account the dynamic behavior of the program, based on a special counter noted as "cnt". When there is a read access to the corresponding block with "P" state, "cnt" is tested and two branches are possible: (1) cnt exceeds δ. In this situation, it is approximate to show that the shared data have not been used frequently. Consequently, an impression that invalidation protocol (cp = 0) may give better performance than the update protocol (cp = 1) is made. Simultaneity, another temporary register called "T" is added in order to determine dynamically suitable threshold value for update. Once the counter reaches 0, the protocol will return to the invalidation. (2) cnt is less than δ. To a certain extent, it is explained a

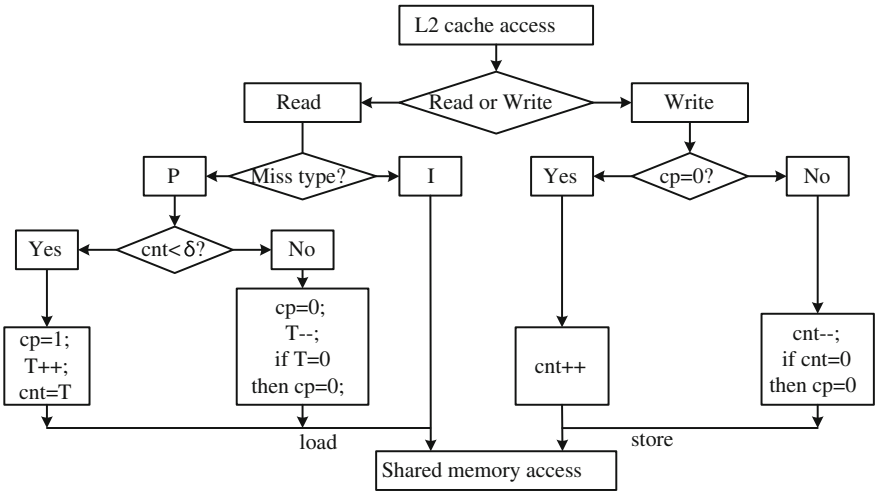

Fig. 3 Algorithm of the dynamic protocol

phenomenon that multiprocessors start to employ the shared data continually, and then, the protocol must switch to update. Under this condition, the temporary counter T is increased after each read operation which is employed to predict the probability that the same data can be modified in succession. Note that the protocol to use initially is invalidation.

Power consumption is another key factor when we take into account the entire framework of cache system initially. According to [10], for an SRAM used for cache, there are three main activities that consume energy: read, write, and idle. These three types of power can be modeled by Eqs. 1–3. All the models show that the energy costs change linearly according to the number of words (M) and number of bits per word (N), with λ and α represent, respectively, the weighting coefficient and technology parameter. Equation 4 is deduced for the whole energy consumption calculation in the SRAM, wherein n_{read}, n_{write}, and n_{idle} are, respectively, the counter values of read access, write access, and waiting cycles. Based on the above viewpoints, the tagless directory adopted in this work has a great effect on the reduction for memory in size. On the other hand, the proposed IVP makes great efforts on the optimization about the number of accesses to L2 tag and directory. Additionally, at the hardware level, the multibanks manner for each directory also causes the most parts of one memory to work as the low power mode.

$$E_{\text{read}} = (\lambda/\lambda_0)^{2\alpha} \times (R_0 + R_2 \times N) \times (R_2 + R_3 \times M) \tag{1}$$

$$E_{\text{write}} = (\lambda/\lambda_0)^{2\alpha} \times (W_0 + W_2 \times N) \times (W_2 + W_3 \times M) \tag{2}$$

$$E_{\text{idle}} = (\lambda/\lambda_0)^{2\alpha} \times (I_0 + I_2 \times N) \times (I_2 + I_3 \times M) \tag{3}$$

$$E_{\text{SRAM}} = n_{\text{read}} \times E_{\text{read}} + n_{\text{write}} \times E_{\text{write}} + n_{\text{idle}} \times E_{\text{idle}} \tag{4}$$

4 Experimental Results

In this section, we evaluate the proposed hybrid protocol during the experiments. The experiment is based on a full-system simulation using Magnusson et al. [11]. Here, we make use of the typical matrix multiplication which is parallelized onto several processors to testify our method. We choose a representative multicore system configuration, having per-core 4 kB L1 Dcache (1,024 entries, directly mapped with a 4-byte line size) for 4 cores and a shared 1-MB L2 cache.

Figure 4 gives the execution time of various configurations obtained with the given application. It proves that the proposed hybrid protocol reduced the execution time 22.2, 34.8, and 51.4 %, respectively, compared to the update, invalidation, and single core. This advantage is due to the reduction in the number of cache misses caused by invalidating read–write shared data frequently. In addition, since the bus contention caused by update protocol is serious, its performance is not as good as the proposed. The worst condition is occurred with the single core in series, which must employ the certain data used in matrix multiplication time and again.

As described in Fig. 5, the experiment uses CACTI 6.5 [12] to estimate the total cache subsystem power overhead across all four configurations. The technique presented reduces the power cost about 8.9, 15.5, and 31.1 % compared to the single core, invalidation, and update, respectively. As we can see, there is tremendous energy overhead when employing the update protocol, since a great many rewriting operations need to access cache memory without limit. With regard to the benefit of our low power scheme, there are two reasonable points to explain according to the power model. Firstly, it is feasible to avoid the unnecessary update in virtue of the dynamic protocol. Secondly, the way to utilize a tagless directory reduces the size of the entire cache.

Figure 6 shows a rough area comparison between the tagless directory and the full bit-vector directory. The latter structure is implemented generally as a matrix of m lines and p columns, where m is the number of memory blocks and p is the number of processors. From this view, the full bit-vector needs integrate an on-chip memory with 256 kb in size, matching with a shared 1M L2 cache. However, the architecture in our work needs only a small amount of extra hardware to keep a copy of all L1 tags. As the result shows, the tagless directory requires 84.3 % less area than the full bit-vector directory.

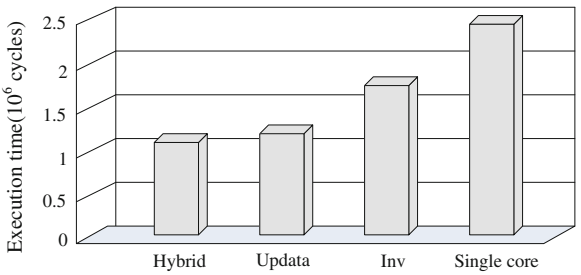

Fig. 4 Execution time of various configurations

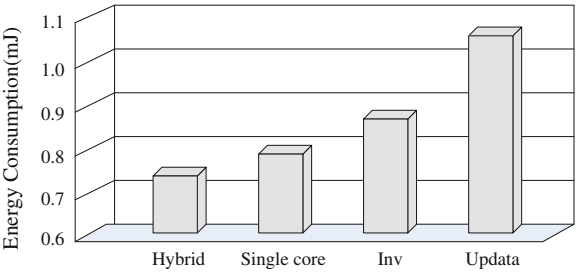

Fig. 5 Total cache subsystem power overhead

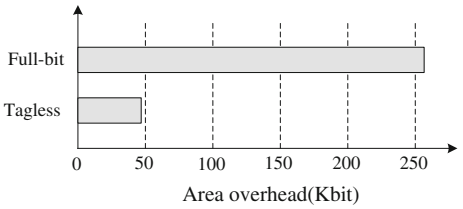

Fig. 6 Area requirements of two directory structures

5 Conclusions

In this work, we have designed a new dynamic hybrid protocol to maintain data coherence. It could enable a capability for solving performance limits due to the use of unique protocol. The protocol presented is implemented based on a new tagless directory structure. In contrast to the traditional full bit-vector directory designs that have large area overhead, this smart directory is realized with a small replication of all L1 cache tags making use of SRAM instead of CAM. Moreover, to choose the appropriate protocol dynamically for a given block, a hybrid protocol algorithm is also presented in detail. The results demonstrate that our technique could significantly reduce execution time, compared to the unique protocol. The results also prove that the proposed directory structure has an ability to reduce the energy consumption and area overhead.

References

1. Kalla R, Sinharoy B, Tendler JM (2004) IBM Power5 Chip: a dual-core multithreaded processor. Micro IEEE 24(2):40–47
2. Chtioui H, Ben Atitallah R, Niar S, Dekeyser J, Abid M (2009) A dynamic hybrid cache coherency protocol for shared-memory MPSoC. Paper presented at the 12th Euromicro conference on the digital system design, architecture, methods and tools, Patras, 27–29 Aug 2009
3. Bournoutian G, Orailoglu A (2011) Dynamic, multi-core cache coherence architecture for power-sensitive mobile processors. Paper presented at the seventh IEEE/ACM/IFIP international conference on hardware/software codesign and system synthesis, Taipei, 9–14 Oct 2011
4. Zebchuk J, Qureshi MK, Srinivasan V, Moshovos A (2009) A tagless coherence directory. Paper presented at the 42nd annual IEEE/ACM international symposium on microarchitecture, New York, 12–16 Dec 2009
5. Ros A, Acacio ME, Garcia JM (2008) DiCo-CMP: efficient cache coherency in tiled CMP architecture. Paper presented at the IEEE international symposium on parallel and distributed processing, Miami, 14–18 April 2008
6. Mark SP, Janak HP (1984) A low-overhead coherence solution for multiprocessors with private cache memories. Paper presented at the 11th annual international symposium on computer architecture, Michigan, 5–7 June 1984
7. Magen N, Kolodny A, Shamir N (2004) Interconnect-power dissipation in a microprocessor. Paper presented at the 2004 international workshop on system level interconnect prediction, Renaissance Paris Hotel Paris, France, 14–15 Feb 2004
8. Wang HS, Peh LS, Malik S (2003) Power-driven design of router microarchitectures in on-chip networks. Paper presented at the 36th annual IEEE/ACM international symposium on microarchitecture, Washington, 3–5 Dec 2003
9. Opensparc T2 system-on-chip (SoC) microarchitecture specification (2008) Redwood shore. http://www.oracle.com/technetwork.systems/opensparc/opensparc-t2-page-1446157.html. Accessed May 2008
10. Atitallah RB, Niar S, Greiner A, Meftali S, Dekeyser JL (2006) Estimating energy consumption for an MPSoC architecture exploration. Paper presented at the 19th international conference on architecture of computing systems, Frankfurt, Germany, 13–16 March 2006
11. Magnusson PS, Virtutech AB, Stockholm S et al (2002) Simics: a full system simulation platform. Computer 35(2):50–58
12. Wilton SJE, Jouppi NP (2002) An enhanced access and cycle time model for on-chip caches. Solid-State Circuits 31(5):677–688