ICOT Technical Report: TR-732

TR-732

Evaluation of the Lock Mechanism in a Snooping Cache

by
T. Tarui, T. Nakagawa, N. Ido, M. Asaie
& M. Sugie (Hitachi)

January, 1992

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

Evaluation of the Lock Mechanism in a Snooping Cache

Toshiaki Tarui, Takayuki Nakagawa, Noriyasu Ido,
Machiko Asaie and Mamoru Sugie

Central Research Laboratory, Hitachi, Ltd.
Higashi Koigakubo, Kokubunji, Tokyo 185, Japan

Abstract

This paper discusses the design concepts of a lock mechanism for a Parallel Inference Machine (the PIM/c prototype) and investigates its performance in detail.

The lock mechanism is implemented by slightly modifying a PIM snooping cache mechanism which uses invalidation to maintain cache coherence. Since lock contention is infrequent during normal memory usage of the PIM, the lock mechanism is designed so as to minimize the lock overhead time in the case of no contentions. This is done by using an invalidation lock mechanism, which utilizes the exclusive state of the snooping cache and in which the locked address is not broadcast.

Experimental results demonstrate the benefits of the lock mechanism in regions of low lock contention. They also confirm that, in most cases, the lock mechanism works well on the PIM. However, the mechanism causes performance degradation when a locked address is accessed by multiple processing elements (PEs) in the TCMP (Tightly-Coupled Multi-Processor). This is because the flags for inter-PE communication in the PIM, such as the load-requesting flag, which are shared by all the PEs, may be accessed by multiple PEs at the same time, thus generating heavy contention. This paper also shows that combining a register-based broadcasting facility with the proposed lock mechanism can solve the above problem.

1. Introduction

Shared-memory parallel architectures show the most promise for the development of high-performance supercomputer systems. There are two main types of parallel architectures, SIMD (Single Instruction Multiple Data-stream) and MIMD (Multiple Instruction Multiple Data-stream). An SIMD system can achieve high parallelism efficiently because no complex control is necessary. However, SIMD machines are not suited to all applications, and they cannot execute non-uniform, non-deterministic programs. For this reason, an MIMD architecture is usually recommended for developing general-purpose parallel machines. In MIMD systems which use fine-grain parallel operations, all the processors operate independently. Therefore, an efficient lock mechanism (such as compare-and-swap) is essential to protect shared addresses and to ensure correct computation results when many PEs access the same memory address at the same time.

Japan's Fifth Generation Computer Project [1] has been driven by ICOT (the Institute for new generation COmputer Technology). The main goal of this project is to develop a knowledge and information processing system, utilizing the logic programming language KL1 [2], which has a stream-AND parallel feature and inter-process communication/synchronization capabilities. A parallel machine customized for this language, the parallel inference machine (PIM), is one of the most important research themes of the project. The PIM project [3], now under development, is designed to exploit these features for the efficient parallel execution of KL1 programs. PIM model c (PIM/c) [4] is a machine developed by Hitachi consisting of 256 PEs (processing elements). In the course of this project, although the PIM is dedicated to knowledge and information processing, many new ideas relevant to all MIMD machines have been developed. The TCMP lock mechanism is one such example.

PIM/c is hierarchically organized, being composed of loosely-coupled clusters of TCMP (Tightly-Coupled Multi-Processor). This structure enables the

reduction of parallel processing overhead, especially the communication/synchronization overhead, by utilizing the locality of programs. In large-scale parallel machines, high speed communication between the PEs is limited by the interconnection distance. This problem is aggravated in a flat structure. In contrast, PEs can communicate at high speed inside a local cluster composed of a limited number of PEs. Also, a convenient shared memory can be introduced into the cluster.

A snooping cache [5] is employed in PIM/c clusters to support efficient communications. During the execution of KL1 programs, logical variables are shared among the PEs. Therefore, fine-grain communications between processors occur frequently, causing a cache coherence problem. From the viewpoint of overhead reduction, as many PEs as possible should be installed in a cluster because the communication overhead is much higher between clusters than within a single cluster. Therefore, inter-PE traffic through the common bus should be minimized. A snooping cache solves this cache coherence problem with a low overhead and, at the same time, reduces the common-bus traffic. The "five-state snooping cache protocol" [6] was proposed to support fine PIM data accesses efficiently.

Synchronization, queue management and load dispatching are essential operations in supercomputers with a shared memory parallel architecture. The KL1 language generates frequent lock accesses because synchronization through the logical variables is extremely frequent. Queue management and load dispatching operations, which are required in parallel processing, also require lock operations. Thus, the overhead reduction brought about by the lock mechanism is a very important issue in the design of the PIM/c. The lock mechanism must be designed so that it works efficiently with KL1 programs. Since lock contention is uncommon in normal memory accesses of KL1 programs, the mechanism is designed to minimize the lock overhead in the case of no contentions. Therefore, an invalidation lock mechanism, which utilizes the exclusive state of the snooping cache, is used. In addition, the lock address register has a "with waiter" state [7], which indicates that another PE is waiting

for the address to be unlocked, in order to eliminate unnecessary common-bus commands.

Since the TCMP of the PIM/c uses a snooping cache, the lock mechanism should use the snooping cache hardware in order to reduce the system hardware requirements and consequently the clock cycle. The proposed lock mechanism is thus implemented by slightly modifying the PIM snooping cache mechanism. We have completed the development of a small-scale PIM/c prototype in which the snooping cache/lock mechanism is integrated in a single LSI chip.

In this paper, the basic design concepts of the PIM/c lock mechanism are described first. The performance is next evaluated on the prototype. Finally, its merits and drawbacks are evaluated in detail.

During the initial stages of the PIM's development, the lock mechanism was evaluated using a function-level simulation which exhibited its macroscopic behavior. After these investigations, the present PIM lock protocol was developed [8]. The lock behavior is determined by many parameters. Because the common bus and its control mechanism are common to both the snooping cache controller and the lock controller, the performance of the lock cannot be evaluated independently of the snooping cache mechanism. Furthermore, detailed structures, such as common-bus arbitration, common-bus timing or resolution of cache access conflicts, may also influence its behavior. Until now, the behavior of the PIM/c lock mechanism has not been simulated in detail as a very complicated parallel cache simulator would be required. Having completed the development of the real lock mechanism hardware in a single LSI chip, it is now necessary and possible to evaluate its performance.

Since an investigation of the fundamental lock mechanism characteristics is of prime importance, benchmark programs of specific applications were not used. This was motivated by the fact that benchmark programs can give us only limited data. Also the bare performance of the hardware cannot be measured, because the characteristics of the application may affect the performance. Furthermore, it is difficult to obtain a wide range of access parameters with such benchmark programs. Instead, artificial access patterns

were used, in which the lock access parameters, such as the lock ratio, could be controlled.

2. The PIM/c snooping cache

2.1. The PIM/c prototype architecture

Figure 2.1 shows the organization of the PIM/c prototype, in which the TCMPs are connected hierarchically. A loosely-connected network connects 32 clusters which each consist of a nine-processor TCMP.

Inside each cluster, eight PEs, one CC (cluster controller) and the main memory are connected with a two-way interleaved common bus. Each PE consists of a processor used for local KL1 execution, and the CC processor is used for controlling network communications. The PEs and the CC use snooping caches to support efficient communications. A two-way interleaved bus is used to increase throughput. As a result, the bus, the main memory and the caches are all interleaved.

So far, a 16-PE model (2 clusters × 8 PEs) has been developed, and a 256-PE model is under way.

2.2. The PIM snooping cache protocol

KL1 programs require frequent inter-process communication. This includes memory cell allocation, a suspension/resumption mechanism for synchronization, and incremental garbage collection. The basic characteristics of PIM cache accesses are as follows:

- Fine-grain communication

  The basic data sizes in logic programming are single words (logical variables) and double words (list cells).

- High write ratio

The KL1 system requires frequent synchronization operations, and consequently, the synchronization variables are often written to.

- High lock ratio

  Since the memory cells are shared by all the PEs, synchronization and queue/flag management must include a lock operation.

- High locality

  KL1 programs often make frequent accesses to the same address. Tail-recursion optimization can reduce the working set size and thus a high cache hit ratio can be achieved.

- Frequent inter-cache communications

  Since logical variables are referred to by both producer and consumer processes, most of the inter-process communications can be accomplished by inter-cache data transfers.

The PIM snooping cache mechanism should match the PIM cache access characteristics. These are summarized below:

- Invalidation scheme

  Based on the high access locality and the high write ratios of KL1 programs, invalidation is used to solve cache coherence problems in the PIM. In this scheme, data in other caches is invalidated at the first write access, and all successive write accesses can be accomplished locally, without common-bus transactions.

- Write-back

  With high write ratios, a write-back policy is used to reduce common-bus/memory traffic.

- Inter-cache data transfer

  An inter-cache transfer mechanism is employed to enable fast data fetches because accesses to cache are faster than to main memory.

- Four-word cache block

  Cache blocks are only four words in length because of the fine-grain memory access characteristics of the PIM.

Figure 2.2 shows the cache block states of the PIM/c snooping cache and the action to be performed in each state. Five states are defined in each cache block to enable the following state attributes to be represented:

- exclusive/shared

  An exclusive state indicates that no other cache contains this block. Wasteful invalidation can be eliminated because invalidation commands are sent only when the block is duplicated in another cache.

- clean/modified

  A modified state indicates that data in the cache is inconsistent with the data in main memory. This state is required in order to support the write-back policy.

- valid/invalid

  An invalid state indicates that the data in the cache is stale.

3. The PIM/c lock mechanism

3.1. Design issues

During the execution of parallel logic programs, lock operations occur frequently, because such programs include frequent synchronizations through logical variables, queue management operations and events related to load dispatching. Therefore, the reduction of lock operation overheads is one of the most important design issues.

Lock contention rarely occurs on the PIM. This is because most lock operations are used to perform "compare and swap" operations on logical variables and, therefore, the corresponding lock time is not long. Furthermore, each PE accesses a variable at most once per logical reduction (approximately 100 cycles) and the number of PEs sharing a variable is not large (in most cases only two, a producer and a consumer). Normally, access contention for logical variables is detected by the failure of a compare-and-swap operation, not by

the lock contention itself. For this reason, the PIM lock mechanism should be designed such that the lock overhead can be minimized when contention does not occur.

The PIM lock mechanism has been developed around the above-mentioned access characteristics. Below, the basic design concepts of the PIM lock mechanism are presented. When a PE locks an address, copies of the corresponding cache block in the other PEs are erased using a invalidation operation and the state of the cache block which includes the locked address is changed to exclusive. This prevents other PEs from accessing the locked address in their own caches. This lock operation will be referred to as the "invalidation lock mechanism" in the rest of this paper.

Some advantages of the invalidation lock mechanism are:

- The additional hardware required to implement the lock mechanism is small, because most of it is already incorporated into the PIM snooping cache mechanism.
- The lock overhead can be reduced using the cache state. For example, if the cache block which includes the locked address is flagged as exclusive, the invalidation command which would otherwise be caused by the lock operation can be eliminated.

A disadvantage of the invalidation lock mechanism is that when lock contentions occur frequently, common-bus traffic will be high. This is because when a PE locks an address, there is no mechanism to pass the locked address on to other PEs. Therefore, other PEs still generate fetch commands to the locked address which result in failure. This overhead cannot be avoided because the invalidation lock mechanism is designed to minimize overheads when lock contention does not occur. Such a mechanism is suitable for the PIM because in this case lock contention is not frequent.

An alternative is the broadcasting lock scheme in which the locked address is broadcast to all the PEs in the cluster. This scheme reduce the lock contention overhead since, after broadcasting, all the PEs know the locked address. Thus, accesses to the address which results in failure can be performed locally, in other words, without a common-bus command. Thus, the common-bus

overhead can be reduced when frequent lock contention is present. However, in the broadcasting scheme the lock/unlock commands are always broadcast to the common bus, regardless of whether a lock contention occurs or not, and the lock/unlock overhead is high when lock contentions are rare. Thus, the broadcasting scheme is not suitable for the PIM.

In the case of supercomputers dedicated mainly to arithmetic operations, lock contention is not frequent during normal data accesses. Thus, the use of an invalidation lock scheme is also appropriate in this case. However, the control variables for process synchronization may sometimes cause lock contentions, for example when many processes perform a fetch-and-add operation on such a variable simultaneously. Thus, the lock contention problem could still arise.

3.2. The PIM lock hardware

Figure 3.1 shows the block diagram of a PIM/c snooping cache which includes the lock mechanism. The cache mechanism incorporates a snooping cache control IC, which includes the CAA (Cache Address Array) RAM, a bus snooping circuit and address/data selectors/buffer. This is linked to an external CDA (Cache Data Array) RAM. The lock mechanism is composed of the following components, which are all implemented in the cache control IC:

- Lock directory

  The lock address and state are stored in the lock directory. There are three lock states.

L (Lock)	An L state indicates that the address in the lock directory is locked.
LW (Lock with Waiter)	An LW state indicates that the address is locked and another PE is waiting for it to be unlocked. This state is required to eliminate unlock commands when there are no waiting PEs.
E (Empty)	An E state indicates that the lock directory is not used.

- Control circuits

  The lock control circuit executes CPU commands and at the same time, snoops the common-bus addresses. The bus-snooping mechanism detects lock contentions by comparing the lock address in the lock directory with the common-bus addresses generated by the fetch accesses of other PEs. While a PE is waiting for an unlock, it can also detect unlock operations by comparing the unlock addresses with the CPU address. The lock control circuits are implemented by adding a slight modification to the cache control circuits.

- Common-bus signals

  The common bus comprises address/data lines, command lines and bus arbitration signals. A round-robin scheme is used for bus arbitration, allowing all PEs to receive equal priority. The lock mechanism and cache control circuits use practically the same signals. The only additional signals required to implement the lock mechanism are an unlock signal and a lock-hit (LH) signal. The LH signal is used to inform other PEs which attempt to fetch data that the fetch address is locked. On receipt of an LH signal, the fetching PEs stop execution until an unlock command is generated, and the locking PE changes state to LW. The LH signal is common to all the PEs in the cluster.

Table 3.1 shows the cache access cycles and bus command lengths of snooping cache operations. Each value in the table is the number of cycles for the ideal case, ignoring the overheads of bus snooping time, bus arbitration time and memory waiting time. Thus, the real cycle times will be slightly longer.

3.3. The PIM lock control mechanism

In the PIM, the CPU lock operation commands are as follows:

LR (Lock Read) read data and lock the address

UW (Unlock Write) write data and unlock the address

U (Unlock) unlock the address

Using these basic operations, more complex lock operations, such as compare-and-swap, can be performed.

There are four basic common-bus commands:

F (Fetch) read the block

I (Invalidation) invalidate the block in other caches

SO (Swap-Out) write the block back to main memory

U (Unlock) unlock the address

Beside the four basic commands mentioned above, mixed commands, involving the fetch command, and other commands are used to decrease the bus traffic. During such commands, several commands can be executed in one bus operation. For example, the mixed command of F and I (fetch the contents of another cache block and invalidate them at the same time) can be executed in the same number of bus cycles as a normal fetch command. Thus, the overhead brought about by invalidation disappears when it is done in association with the fetch operation. In this way, the common-bus overhead produced by the cache coherence maintenance and the lock operation can be reduced.

Figure 3.2 shows the states of the PIM lock mechanism and the actions to be performed at each state. The invalidation lock mechanism has no need for a separate lock command on the common bus because the locking is already performed by the invalidation command, which is used for the maintenance of cache coherency. Thus, there is no need for a complex command in order to perform a lock operation. Also, when an invalidation command is required, a mixed fetch-and-invalidation command can be used to reduce the common-bus overhead.

During a lock access, the cache state and the lock state are inspected in order to eliminate unnecessary common-bus commands. When the state of the cache block which includes the locked address is exclusive, the invalidation command which would otherwise be caused during a lock operation can be eliminated. When the lock directory is in state L (but not LW) an unlock command which would otherwise be caused during the unlock operation can also be eliminated. As a result, unlock operations can be performed without bus

commands. If the lock/cache state is not available, these commands must be generated at every lock/unlock operation, and will lead to needless common-bus traffic.

The PIM lock mechanism creates these advantages at the expense of generating heavy bus traffic when frequent lock contentions occur. When many PEs lock a single address, only one PE can issue the lock access. The other PEs, which are not informed about the lock operation, fetch the address at the same time, because the lock address is not broadcast to them. As a result, most of the fetch accesses to the locked address fail because an LH response is issued by the first locking PE. After the unlock command, all the remaining fetch commands are produced again. This sequence of fetch/unlock commands leads to heavy bus traffic.

In our approach, the lock/unlock mechanism is designed to minimize the overhead when lock contentions rarely occur. Furthermore, the additional hardware required to implement the mechanism is quite small, about 5% of the total cache hardware, because most of the lock hardware is common with the snooping cache controller.

4. Evaluation of the PIM lock mechanism

4.1. The evaluation objectives

In order to demonstrate the advantages of the PIM lock mechanism, the hardware performance was measured and analyzed using the PIM/c prototype machine. The effects of the aforementioned drawbacks were also studied in order to clarify the performance limits of the lock mechanism.

In the first stages of PIM development, the lock mechanism design was evaluated by a function-level simulation of its macroscopic behavior. At that time, no detailed simulations were made, because the lock mechanism is tightly coupled with the snooping cache mechanism. This means that a very complicated cache/lock simulation is required to evaluate precisely the

performance of the real lock mechanism. So far, the detailed effects of the control mechanism produced by this common-bus/memory architecture have not been reported. On completion of the hardware development, it became appropriate to study the lock performance in detail.

Conventionally, the performance of computer systems is evaluated by benchmark programs. However, with benchmark programs, only limited data can be measured, and global features covering a wide range of applications cannot be wholly clarified. Furthermore, it might be difficult to evaluate the performance with a wide range of access parameters which are known to cause problems with the design. The proposed lock mechanism shows promise for a wide range of applications, although its behavior has not yet been sufficiently clarified. At present, the global features are more important than the performance in specific applications.

Since this paper focuses on fundamental characteristics which are general over a wide range of applications, artificial access patterns were used, in which the cache access parameters could span a reasonable range.

4.2. Experimental evaluation

The three access parameters of major importance to lock behavior are the lock ratio, the sharing number and the lock contention ratio. Consequently, the effects of these parameters on lock performance were evaluated. The default values of these parameters were chosen to reflect the PIM access characteristics discussed in section 3.1:

- Lock ratio

  The lock ratio is the ratio of the number of lock accesses to the number of total cache accesses. The effect of lock operation on the overall system overhead was measured by varying this ratio. The default ratio was unity, since lock ratios are high in the PIM.

- Sharing number

This is defined as the number of PEs which share a single cache block. The effect on system performance of multiple PEs all locking the same address was measured by varying the sharing number. The default value was set to 3, because there is some locality in PIM cache accesses.

- Lock contention ratio

  Lock contention is frequent if many accesses of the same address are made in a short period. The effects of lock contention on the system performance were measured by varying this ratio. However, unlike the other parameters, the lock contention ratio could not be controlled directly, since it depends on the order of bus commands, which the software cannot control. In this experiment, it was controlled by changing the working set size of the access pattern. With a small working set, each address was accessed many times by several PEs and, consequently, lock contentions were frequent. The default value of the working set was 16M words (the whole memory space), because lock contentions are uncommon on the PIM.

Table 4.1 summarizes the lock access parameters and their default values.

In the investigation of the lock mechanism, result were obtained by the following procedure:

(1) Generate a cache access pattern on a workstation, focusing on the specific lock access parameters.
(2) Download the access pattern to the local memory of each PE.
(3) Execute the cache accesses by means of a microprogram.
(4) Collect statistical data using a hardware monitor.

In the experiments, 40,000 instances of cache access were issued by each PE. In order to avoid deadlocks, the lock accesses had to be followed by unlock accesses to the same address, no other cache accesses being issued between the two. All processors in the cluster, including the CC, had their own cache, and consequently nine caches were connected to the two-way interleaved common bus. Since the access pattern stored in the local memory of each PE was fetched

by the microprogram, the effect of local memory access was eliminated after the data had been measured. The end state of the same access pattern was used as the initial cache state since the focus was on the stable state of the snooping cache. An empty cache could have been used as the initial state, but this causes an excessively high memory access ratio.

In PIM/c, each PE was fitted with a hardware monitor incorporating a 40-bit counter and circuitry to select the event to be counted. This facility made it possible to collect various statistical data such as the number of common-bus commands, the number of lock contentions, and the bus snooping waiting time, without any additional overheads.

4.4. Results

4.4.1. Verification of the lock mechanism

The lock performance was evaluated on the basis that lock contention is not frequent, in order to prove the advantages of the PIM lock mechanism. Figure 4.1 shows the relationship between the lock ratio and the cache throughput efficiency, and also the bus utilization and number of bus commands. The cache throughput efficiency is defined as the ratio of the real data throughput between the CPU and the memory system to the ideal throughput (with no cache overheads). When the lock ratio increases, the cache throughput efficiency increases only slightly. This is because at high lock ratios, the fetch-and-invalidate mixed commands generated by the lock operation, reduced the common-bus overhead.

Figure 4.2 presents the time diagram for common-bus accesses when read and write accesses are issued to the same address in sequence. If the lock ratio is low, most of the cache accesses are normal read/write accesses. In this case, an F (fetch) command is issued for the first read access and an I (invalidation) command is issued for the subsequent write access. Therefore, two common-bus commands are required. On the other hand, if the lock ratio is high, pairs of

LR (lock read) and UW (unlock write) accesses consume most of the cache access time. In this case, fetch and invalidate commands are performed at the first LR access by the mixed command and the UW accesses can then be done without common-bus commands. Therefore, only one common-bus command is issued. Since the FI (fetch and invalidate) command can be performed in the same bus cycle as the F command, the invalidation overhead can be reduced. The decrease in common-bus commands in Figure 4.1 proves this assumption. As the lock ratio increases, the number of normal F and I commands decreases and the number of mixed FI commands increases but by a relatively smaller amount. Overall, the number of bus commands decreases.

Figure 4.3 shows the effects of the sharing number on cache throughput efficiency, bus utilization, external hit ratio and the number of bus commands. The external hit ratio is the probability with which inter-cache data transfers (as opposed to memory access) occur in cache miss cases. A high sharing number causes frequent fetch/invalidation commands between caches and tends to decrease the throughput efficiency. However, the external hit ratio increases as the sharing number increases, and consequently, at high hit ratios, the data delay time in cache miss cases can be reduced by inter-cache data transfers, because the cache operates much faster than main memory. At high sharing numbers, inter-cache data transfer prevents a significant degradation in throughput efficiency which would otherwise be caused.

Consequently, the system performance is largely independent of the lock ratio and of the sharing number, and is the same as when no lock accesses occur. As a result, it is concluded that, when lock contention is infrequent, the lock overhead associated with the invalidation lock mechanism is very small.

4.4.2. Lock overhead

In order to study the aforementioned drawbacks of the PIM lock mechanism, the hardware-level overhead of the lock mechanism was measured. Figure 4.4 shows the relationship between the sharing number and

the number of lock/cache overhead cycles. The bus waiting overhead represents 40% of the total execution time, because bus utilization was high (about 60%) in the access pattern.

The bus command, bus snooping, and memory waiting overheads comprised about 60% of the total lock overhead, independent of the sharing number. These overhead contributions were due only to fetch accesses caused by LR (lock read) commands. If a PE locks an address, an FI command is used to fetch and invalidate the cache block which includes the address. Therefore, four words (one block) of data are transferred, even when only one word is required for the lock access. This copying overhead was the most significant limitation of the invalidation lock mechanism. Furthermore, this data copying may cause a so-called "ping-pong effect" when several PEs lock an address alternately. This drawback is discussed in section 5.

4.4.3. Lock contention

Figure 4.5 shows the relationship between the lock contention ratio (controlled by the working set size) and the cache throughput efficiency. The cache throughput efficiency was measured using the following three types of lock access:

(1) Access with lock and with a round-robin bus arbiter

This is the most common type of lock access.

(2) Access without lock and with a round-robin bus arbiter

In this case, the same addresses as in (1) are accessed, but LR and UW commands are translated to normal write commands. Because the LR command invalidates the contents of other caches, it is translated to a write command instead, so as to enable the pure lock overhead to be evaluated. The performance degradation due to lock contention was measured, comparing the result with (1).

(3) Access with lock and with a fixed-priority bus arbiter

In this case, the same addresses as in (1) were accessed, but the common-bus priority was fixed. The effects of common-bus arbitration on lock performance was also measured, comparing the result with (1).

In Figure 4.5, the cache throughput efficiency ratios (1)/(2) and (1)/(3) are shown.

Figure 4.6 shows the relationship between the lock contention ratio and the bus utilization and the composition of bus operation time. The overhead produced by the following bus operations was evaluated:

- Data transfer due to LR operations
- Failure of fetch access due to lock contention with an LH (lock hit) response
- Unlock commands

The performance degradation caused by lock contention can be investigated in detail.

The effect of lock contention on cache throughput efficiency

Figure 4.5 shows that no lock overhead is observed at low lock contention, when the working set size is larger than 100. When lock contention occurs more frequently, at a working set size of less than 100, the lock performance decreases quickly as the working set size decreases. As a result, it is confirmed that the invalidation lock mechanism will have no lock overhead if lock contention is rare. The lock contention ratio observed in the access pattern itself was about 5% for working sets with a size of 100 words, as a result of nine PEs accessing the two-way interleaved bus. (It should be noted that lock contention observed on the real machine was more significant because common-bus contentions delay the lock accesses.) Thus, up to 5% lock contention can be tolerated by the PIM invalidation lock mechanism, as it only leads to a small lock overhead.

Figure 4.6 shows that when lock contention occurs frequently, the fetch failure overhead increases rapidly as the lock contention increases, and more than 80% of the total bus cycles are consumed by this overhead when lock

contention is extremely frequent. The overhead caused by the unlock operation is small, thus when lock contention occurs frequently, most degradation is caused by fetch command failures due to the LH response from the locking PE. As the invalidation lock mechanism is employed specifically to minimize the lock overhead in the region of low lock contention, this overhead cannot be avoided.

On the other hand, at low lock contention rates, data transfer due to LR operations consumes almost all the bus cycles, and the system performance is limited by bus throughput. This type of data transfer leads to the "ping-pong effect" mentioned in section 4.4.2.

The effect of common-bus arbitration on cache throughput efficiency

Figure 4.5 shows that, at high lock contention rates, the performance obtained using fixed-priority bus arbitration is much better than that obtained with round-robin bus arbitration. This is because when round-robin bus arbitration is used, fetch failures due to lock contentions are more frequent than that obtained with fixed-priority bus arbitration. For example, if all nine PEs lock one address, they produce an FI command on the common bus, allowing the PE with the highest priority to access the address. With a round-robin bus arbiter, the priority of the locking PE drops to the lowest level after the lock operation. As a result, fetch commands generated by other PEs, which fail because of a lock contention, are produced before the unlock operation and cause wasted bus traffic. The experimental results show that about seven fetch failures are produced during one lock operation when all the PEs in a cluster access one address.

On the other hand, with a fixed-priority bus arbiter, the locking PE has a relatively high priority and can produce the unlock command soon after the lock operation. In this case, the wastage of bus cycles due to fetch failure can be eliminated.

In this way, when lock contention occurs frequently, the lock performance is affected greatly by common-bus control operations such as bus arbitration. However, in the PIM, as the lock contention is not frequent, a round-robin bus

arbitration is suitable, because an equal bus overhead can be achieved for each PE.

5. Discussion

In this section, the advantages and disadvantages of the PIM invalidation lock mechanism are studied using the experimental results presented in the previous sections.

As discussed in section 4.4.3, no lock overhead is observed when lock contention is infrequent, because of the following features of the lock mechanism:

- A lock operation can be performed by a local register write and there is no need for lock address broadcasting when the locked address is in the exclusive state.
- Utilizing the mixed fetch-and-invalidate command, invalidation for lock operations can be performed without any common-bus overhead.
- The lock state can be used to reduce unnecessary unlock commands on the common bus when a lock contention does not occur.

Once lock contention occurs, common-bus overheads arise due to fetch failure and unlock operations. After a fetch failure, the PE accessing the locked address stops execution until an unlock command is generated. This feature can reduce the common-bus traffic as the PE waiting for the unlock operation will not be using the bus. However, when multiple PEs access a locked address, multiple fetch failures at the same address occur for all the PEs involved. Thus, when lock contentions are extremely frequent (more than 5% of the total access pattern), frequent fetch failures would raise the bus traffic significantly and the common bus would become a bottleneck.

In the PIM, lock contentions are not frequent in the course of normal inference operation. Thus, the problem mentioned above rarely occurs. However, when flags and queues are shared by several PEs in the cluster, lock contention may occur, because all the PEs sharing the addresses access them

once per logical reduction (approximately 100 cycles). In that case, for example, nine block data transfers (four words each) occur in a single logical reduction. These data transfers take 63 cycles (an inter-cache data transfer takes 7 cycles) and consume more than half of the total bus cycles of a logical reduction. These nine data transfers can be replaced by broadcasting. However, broadcasting cache protocols have heavy drawbacks in normal cache operations.

When a flag is shared by all the PEs and is accessed by each PE once per logical reduction, a significant performance degradation arises. This is because the invalidation cache/lock mechanism used in the PIM will cause heavy bus traffic due to frequent lock contentions and data transfers as discussed in 4.4.2 (the so-called "ping-pong effect"). Such a situation might be generated, for example, by the flag for the load dispatching operation. Any PE requesting the load sets the corresponding flag and all the other PEs read this flag once per logical reduction. If the flag is ON, the PE reading the flag resets it and dispatches a job. If the flag is OFF, no operation is necessary.

In this case, frequent one-to-any communications are performed using lock operations. When the flag is ON and load dispatching is required, the lock mechanism overhead is not significant because a load dispatching operation takes many cycles. On the other hand, when the flag is OFF, these flag-fetching operations cause a significant degradation, because no other flag operations are performed. Furthermore, most of the PEs access an OFF flag because load dispatching is not very frequent. Because of this, an additional communication mechanism is required for this kind of operation.

In order to solve this problem, the PIM employs a software interruption mechanism, "slit-check" [9]. In this mechanism, each PE has a special register of 16 bit flags and the value of each flag can be broadcast among the PEs. In each PE, the contents of this register can be checked by the CPU. Because this register is independent of the cache/lock mechanism, the cache/lock protocol is not altered. As the slit-check mechanism does not have a lock feature, it must use the lock mechanism associated with the cache. The slit-check flag is checked for a particular event, and if the flag is ON, real operations are performed using

the data in the cache. Using the slit-check mechanism with the invalidation cache/lock mechanism, the above-mentioned flag operations, which would otherwise cause heavy overheads, can be performed with a very small overhead. The hardware needed to implement the slit-check is about 5% of the total cache/lock controller.

The combination of the lock mechanism with an invalidation protocol and the slit-check mechanism with a broadcast facility can help in implementing efficient TCMP systems.

6. Conclusions

In this paper, the design concepts for the PIM/c lock mechanism were described. This lock mechanism has now been integrated in a single IC with the snooping cache controller. Several fundamental characteristics of the lock mechanism have been investigated using a small-scale prototype of the PIM/c, which is composed of nine-PE TCMPs.

The main results obtained through our investigation are as follows:

(1) When lock contention is infrequent, no lock overhead is observed because of the following characteristics of the PIM lock mechanism:

- The lock operation can be performed with just a local register write and an invalidation.
- If the state of the locked address is exclusive, the associated invalidation operation can be eliminated and no common-bus command is required.
- The mixed fetch-and-invalidate command can reduce the invalidation overheads of the lock.
- Using the lock state stored in the lock directory, unlock operations can be eliminated when no lock contentions occur.

(2) Frequent lock contentions cause the lock overhead to increase quickly with increasing contention, because fetch commands fail much more

frequently. The fetch failure increases as the number of locking PEs increases.

(3) Frequent lock contentions also cause the lock performance to be affected greatly by the common-bus control (e.g. by bus arbitration). In the PIM, a fixed-priority bus arbitration reduces the fetch failure rate and improves the performance.

(4) The upper limit of the lock contention ratio (discussed at (1)) as determined from the access pattern (the address traces of the PEs) is about 5%. The lock mechanism is suitable for the PIM because normal memory accesses remain in this area.

(5) The lock contention problem (discussed in (2)) may occur at a flag shared by multiple PEs in the cluster, because the access frequency is extremely high. We are convinced that the combination of the cache/lock mechanism with an invalidation protocol and the slit-check mechanism with a register-broadcasting facility can help realize efficient TCMP systems.

Acknowledgements

The authors would like to thank Dr. Shun'ichi Uchida, manager of the research department of ICOT, for his guidance and support, and Dr. Kazuo Taki, chief of 1st ICOT Laboratory, for his helpful discussions. This research was sponsored by ICOT.

References

[1] S. Uchida, K. Taki, K. Nakajima, A. Goto and T. Chikayama, "Research and Development of the Parallel Inference System in the Intermediate Stage of the FGCS Project," Proceedings of the International Conference on Fifth Generation Computer Systems 1988, pp. 3-15, 1988.

[2] K. Ueda and T. Chikayama, "Design of the Kernel Language for the Parallel Inference Machine," Computer Journal, Vol. 33, No. 6, pp. 494-500, 1990.

[3] A. Goto, M. Sato, K. Nakajima, K. Taki and A. Matsumoto, "Overview of the Parallel Inference Machine Architecture," Proceedings of the International Conference on Fifth Generation Computer Systems 1988, pp. 208-229, 1988.

[4] A. Goto, K. Taki, T. Nakagawa and M. Sugie, "Parallel Inference Machine PIM/c - Global Structure," Proceedings of the 40th Annual Convention IPS Japan, 2L-1, 1990 (in Japanese).

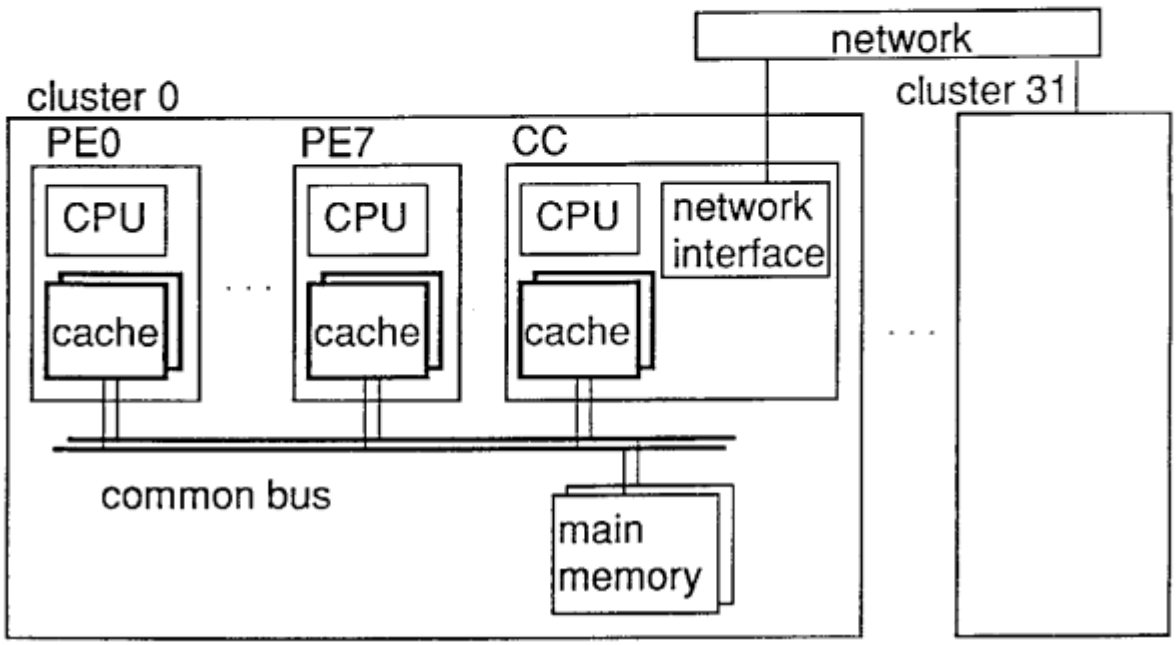
[5] J. Archibald and J. Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model," ACM Transaction of Computer Systems, Vol. 4, No. 4, pp. 273-296, 1986.

[6] A. Goto, A. Matsumoto and E. Tick, "Design and Performance of a Coherent Cache for Parallel Logic Programming Architectures," Proceedings of the 16th ISCA, pp. 25-33, 1989.

[7] P. Bitar and A. M. Despain, "Multiprocessor Cache Synchronization Issues, Innovations, Evolution," Proceedings of the 13th ISCA, pp. 424-433, 1986.

[8] A. Matsumoto, T. Nakagawa and A. Goto, "Parallel Inference Machine: PIM - On Parallel Cache and Lock Mechanism," Proceedings of the 34th Annual Convention IPS Japan, 2P-6, 1986 (in Japanese).

[9] T. Nakagawa, A. Goto and T. Chikayama, "Slit-Check Features to Speed Up Interprocessor Software Interruption Handling," IPSJ SIG Reports of the Computer Architecture IPS Japan, 77-3, 1989 (in Japanese).



PE: Processing Element
CC: Cluster Controller

Fig. 2.1. Organization of the PIM/c prototype.

(PIM/c: Parallel Inference Machine model C)

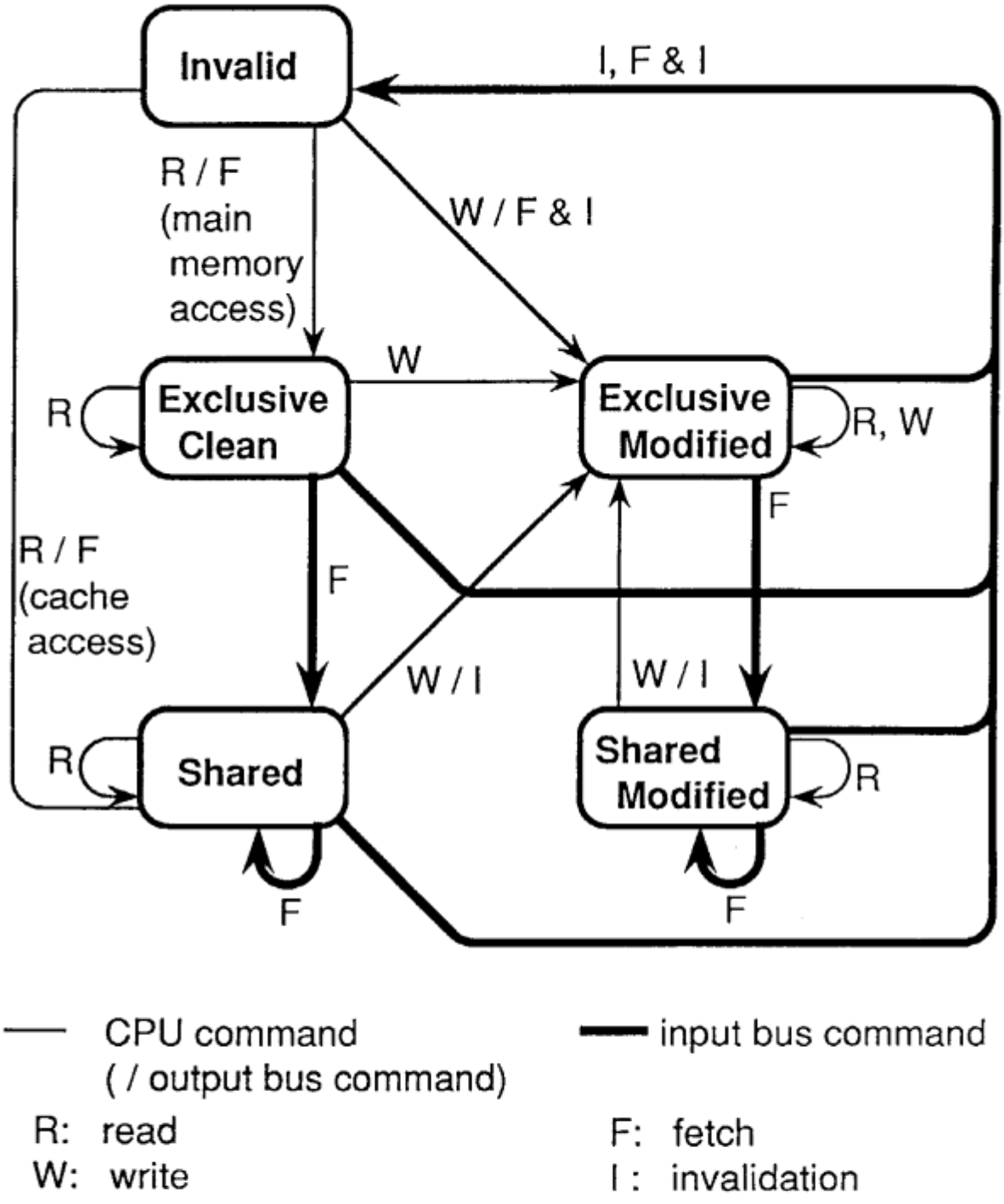


Fig. 2.2. PIM/c snooping cache state diagram.

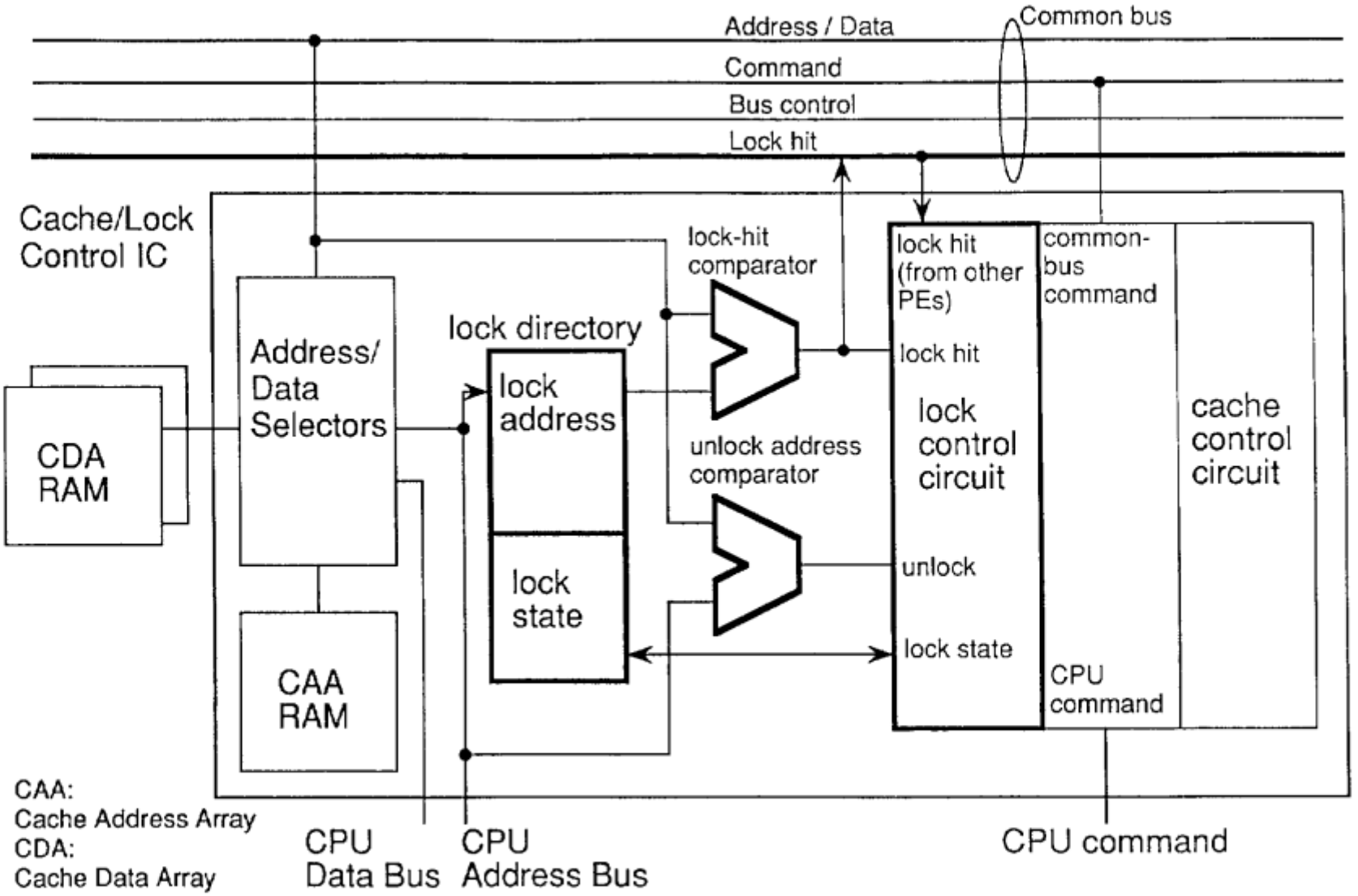


Fig. 3.1. Block diagram of PIM/c snooping cache.

Table 3.1.
Cache Access Cycles and Bus Command Length

Com-mand	Conditions		Cache Access Time	Bus Command Length
Read Write	Cache Hit	No Invalidation	2	
		Invalidation	4	3
	Cache Miss	No Swap-out	10	7
		Swap-out	13	10
Unlock	No Lock waiting		2	
	Lock waiting		4	3

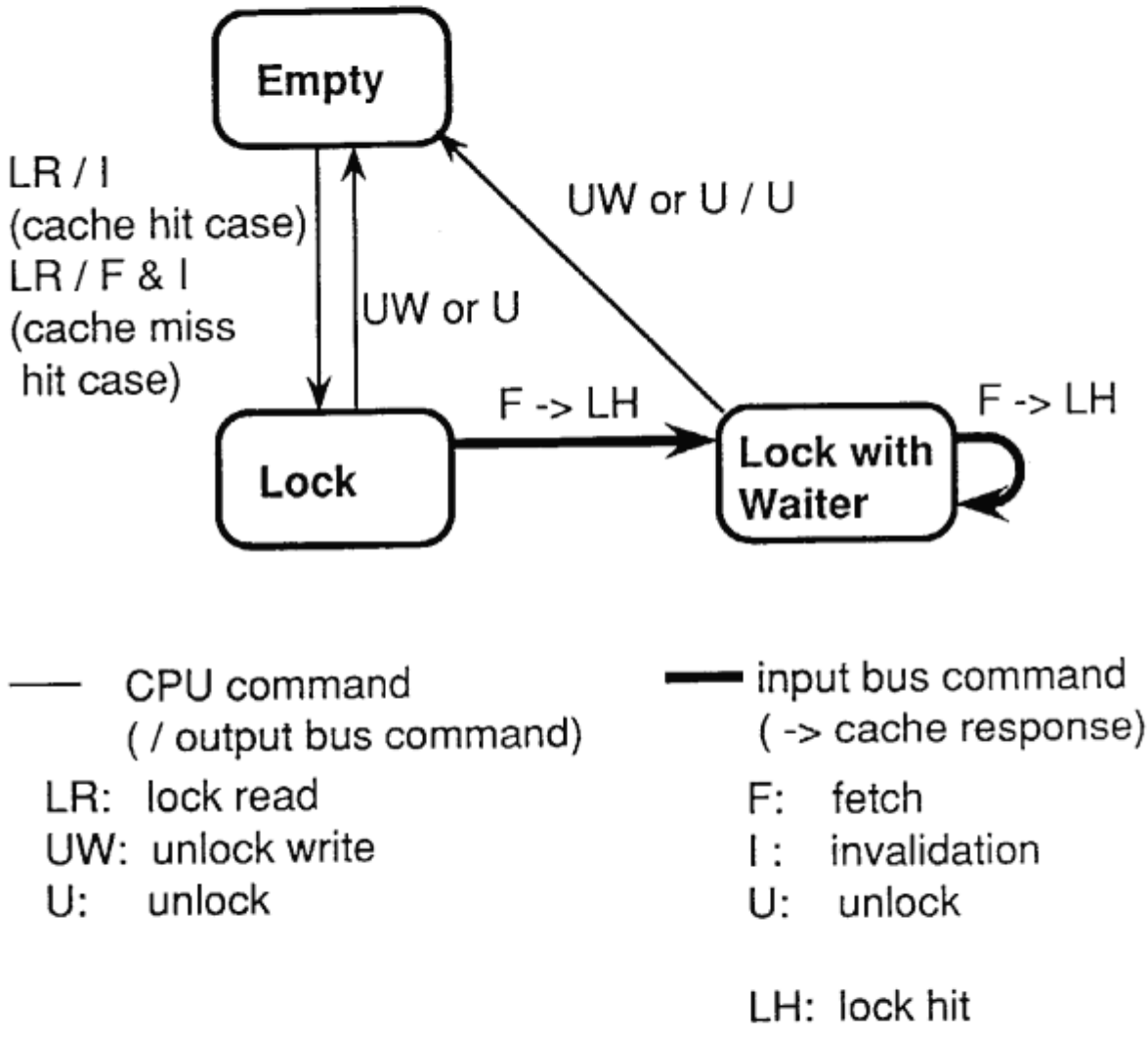


Fig. 3.2. States and actions of the PIM lock mechanism.

Table 4.1.
Lock Access Parameters

Parameter	Range	Default Value	Notes
Lock ratio	0.0 - 1.0	1.0	$\frac{\text{Number of lock accesses}}{\text{Total number of cache accesses}}$
Sharing number	2 - 9	3	Number of PEs sharing a cache block
Working set size	1 - 16M(words)	16M(words)	Controls lock contention

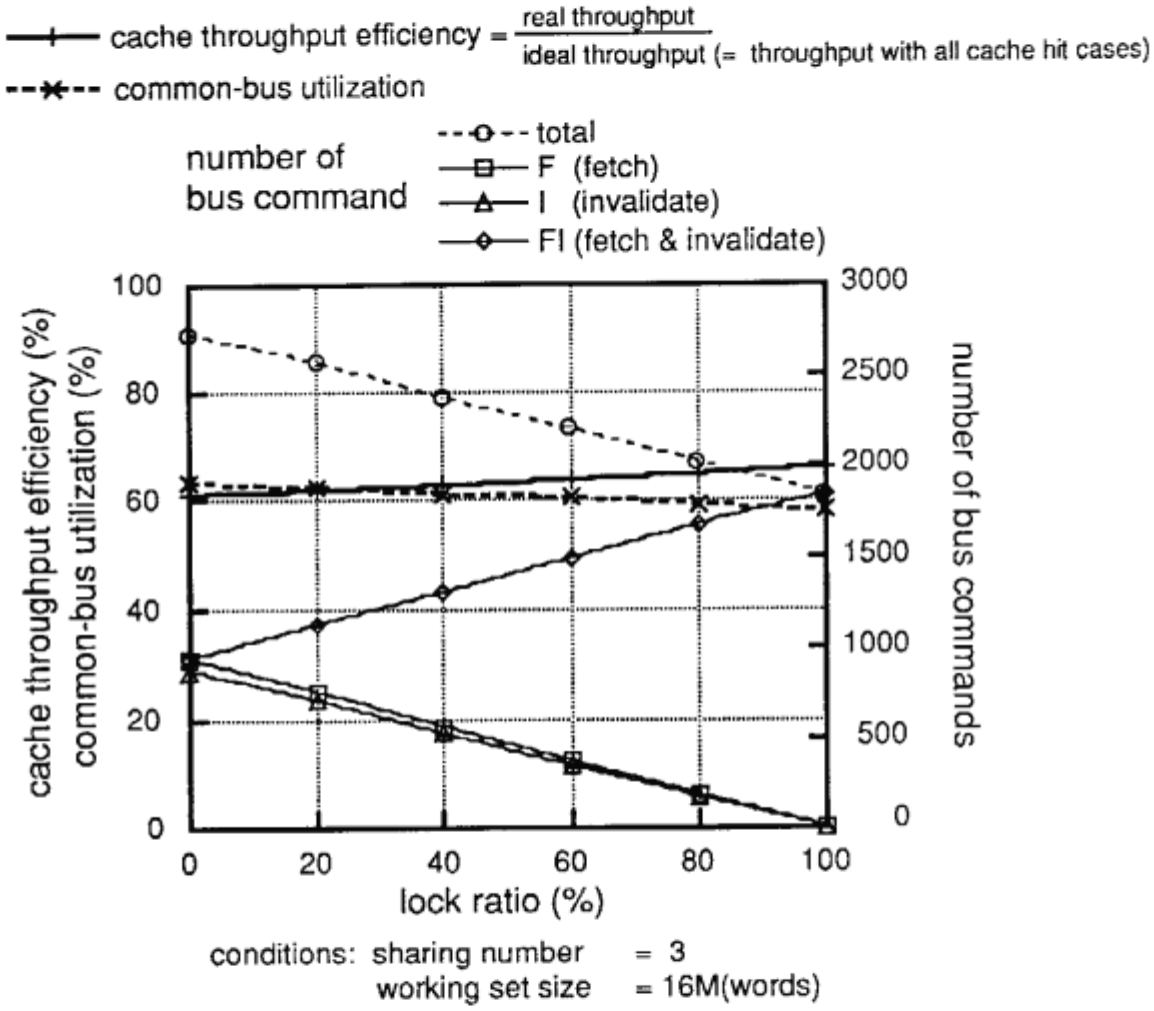
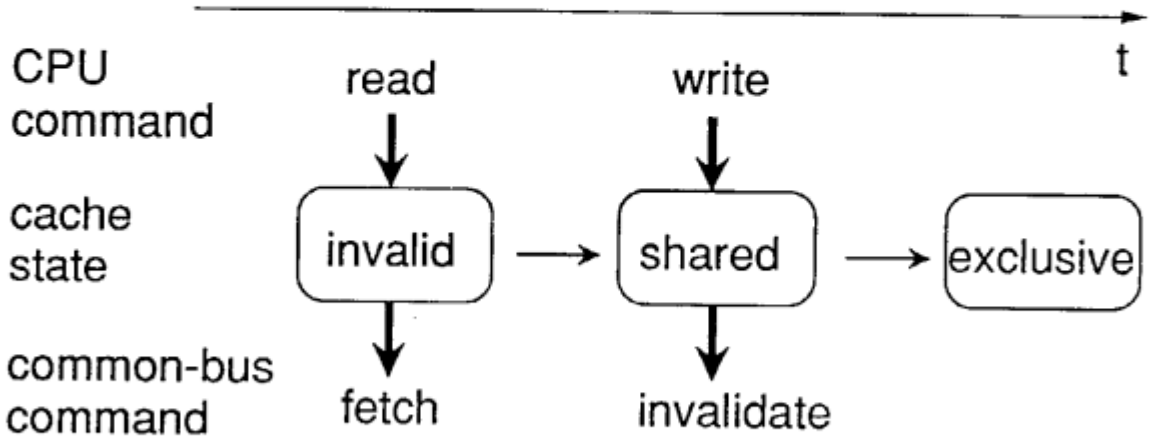


Fig. 4.1. Cache throughput efficiency, common-bus utilization and number of bus commands as a function of lock ratio.

(1) Normal read / write access



(2) Lock read / write access

(Number of bus commands can be reduced)

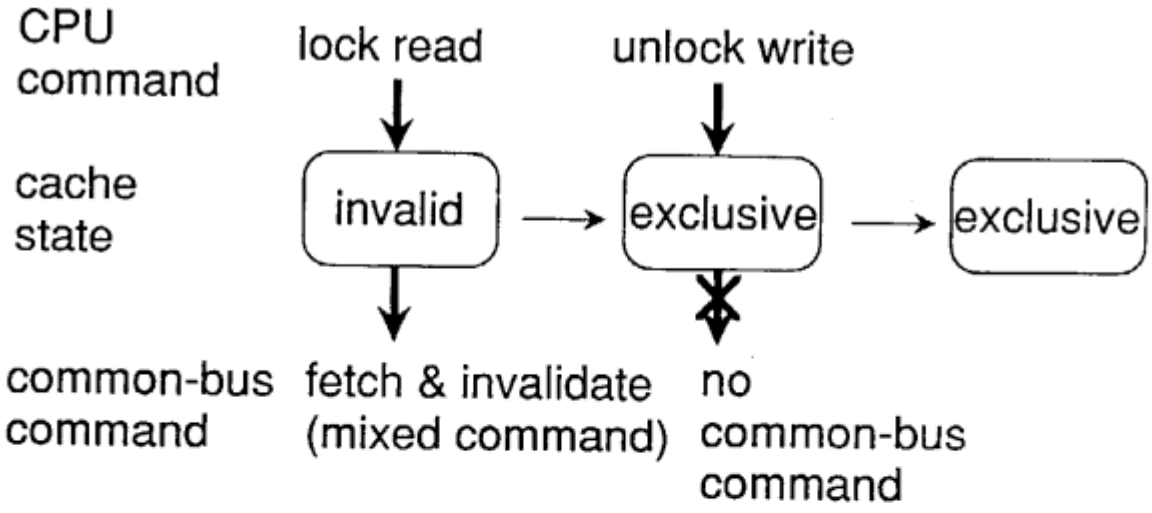


Fig. 4.2. Time diagram for common-bus accesses.

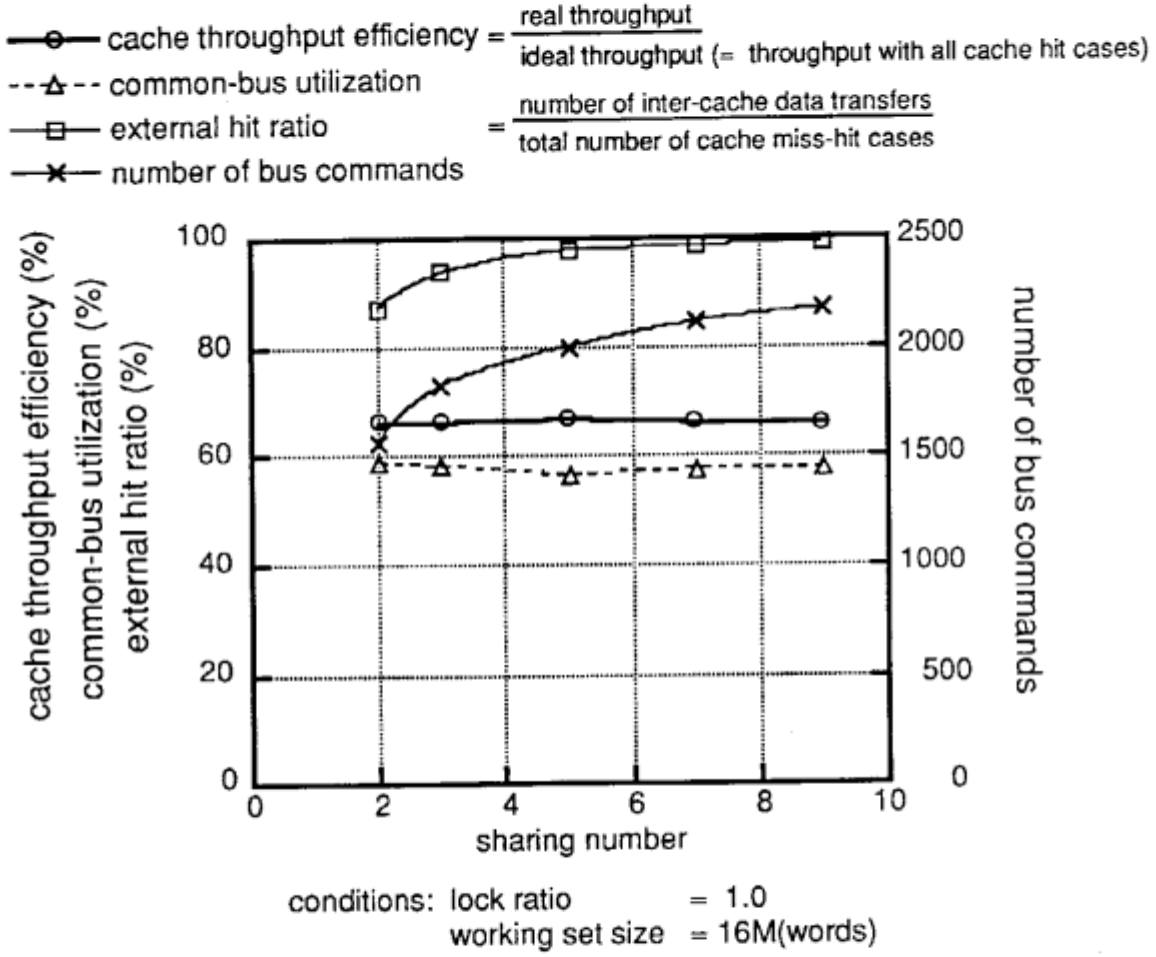


Fig. 4.3. Cache throughput efficiency, bus utilization, external hit ratio and number of bus commands as a function of sharing number.

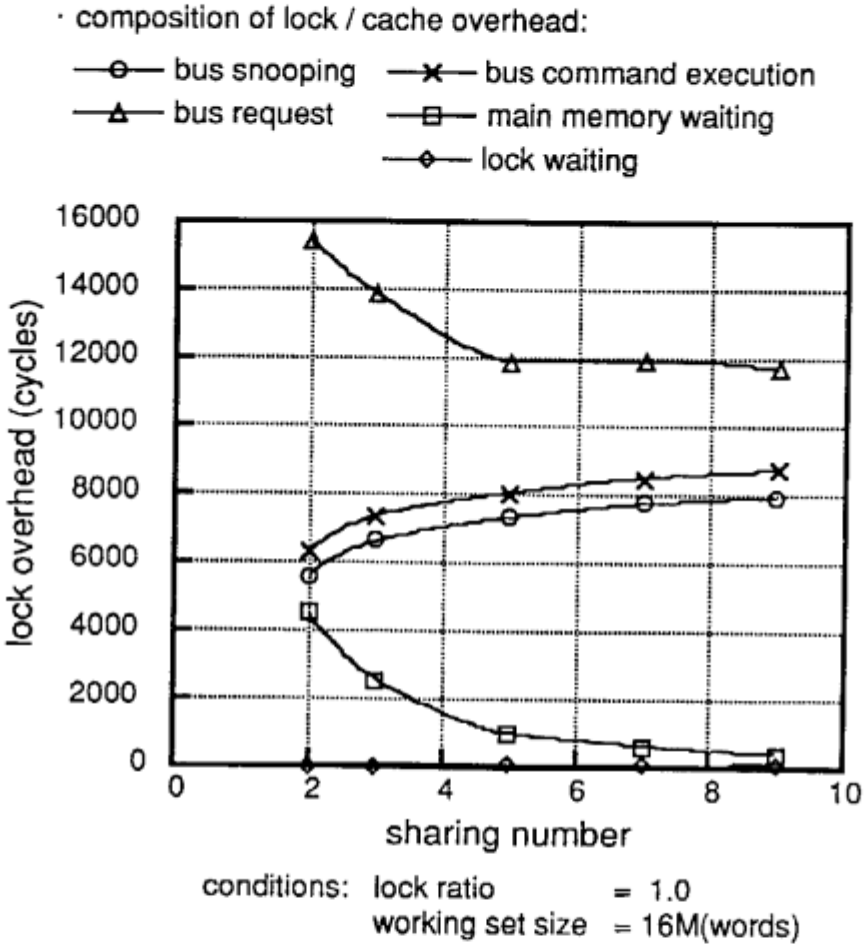


Fig. 4.4. Lock / cache overhead cycles as a function of sharing number.

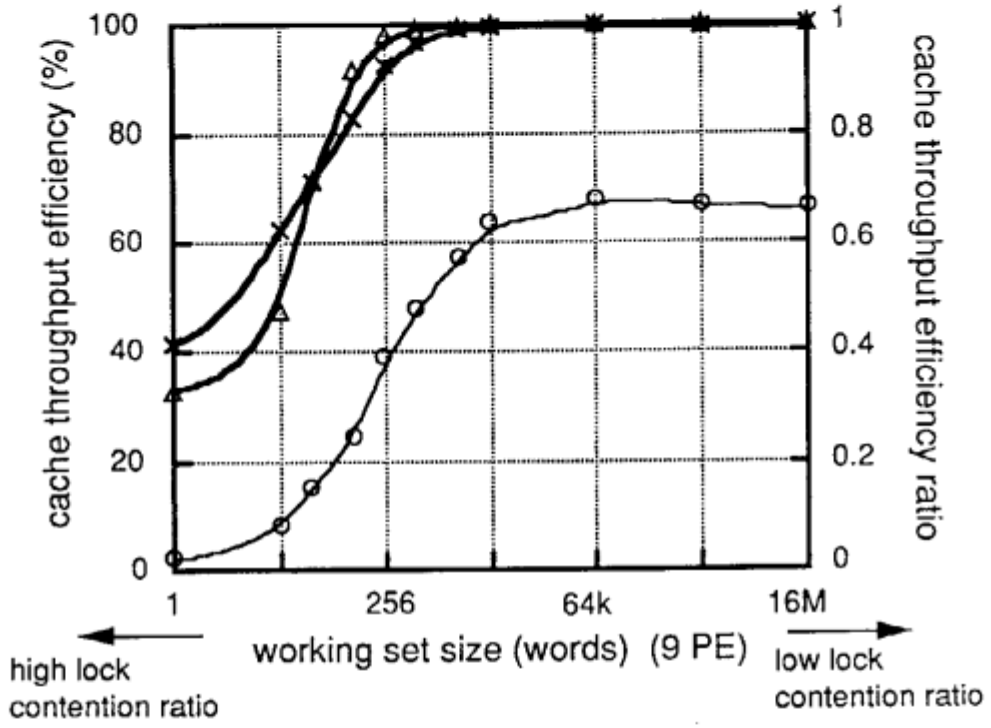
· cache throughput efficiency = $\frac{\text{real throughput}}{\text{ideal throughput (= throughput with all cache hit cases)}}$:

—o— access with lock and a round-robin bus arbiter

· cache throughput efficiency ratio:

—△— access with lock / access without lock

—×— access with a round-robin bus arbiter / a fixed priority bus arbiter



conditions: lock ratio = 1.0
sharing number = 3

Fig. 4.5. Cache throughput efficiency as a function of lock contention.

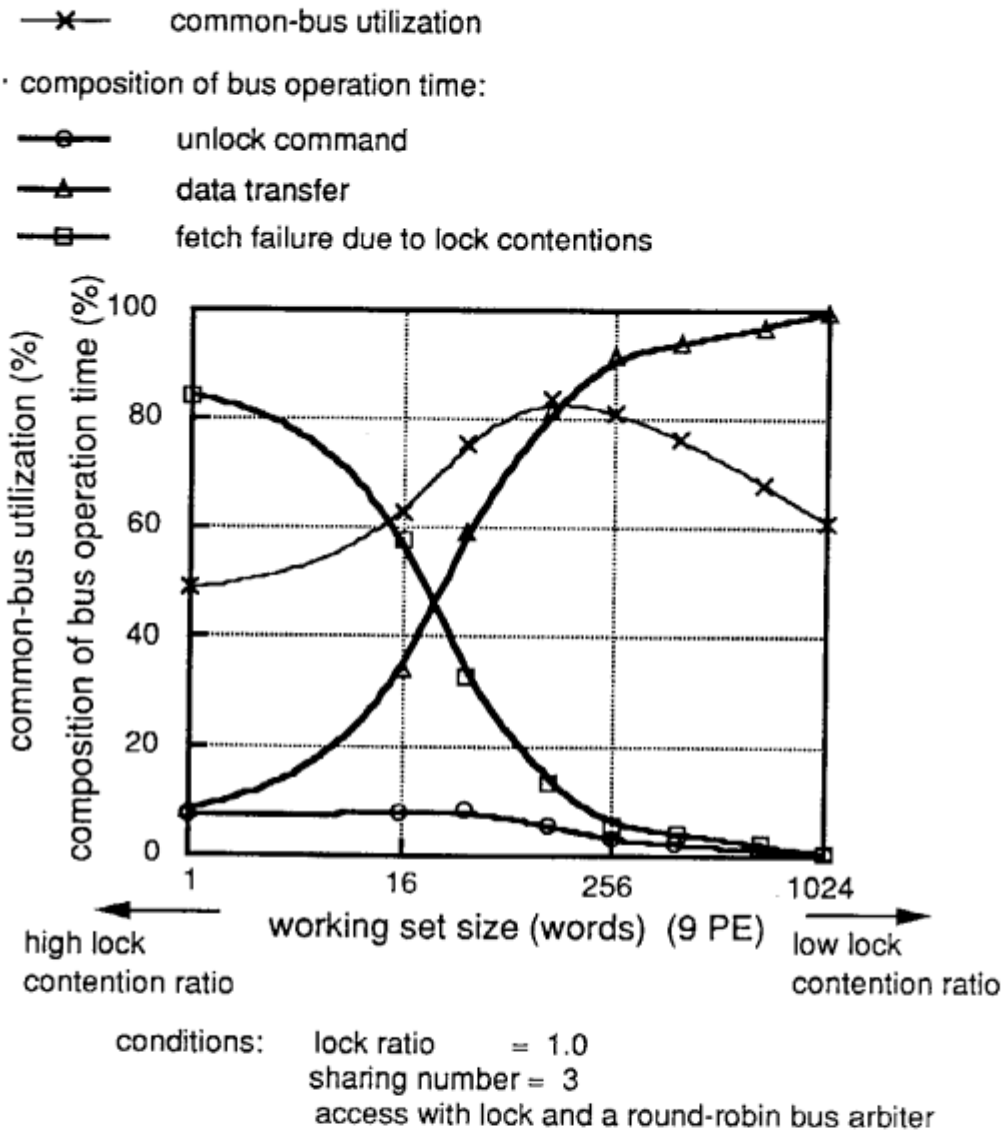


Fig. 4.6. Common-bus utilization and composition of bus operation time as a function of lock contention.