Future Generation Computer Systems 9 (1993) 235–240
North-Holland

The M^2 hierarchical multiprocessor

Yen-Jen Oyang *, David Jinsung Sheu, Chih-Yuan Cheng and Cheng-Zen Yang

Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan

Abstract

This paper discusses the design and development of a bus-based hierarchical multiprocessor named M^2. The primary design goal of the M^2 is to derive a multiprocessor architecture that features much higher degree of scalability than the shared-memory shared-bus architecture to meet the ever increasing processing power demanded by large database/knowledgebase computing and transaction processing. If compared with other hierarchical multiprocessors, the M^2 is distinctive in its memory configuration, which is aimed at avoiding severe inter-CPU interference due to page-swapping events. If compared with a group of multiprocessors connected by a local area network, the M^2 enjoys higher scalability due to higher bandwidth of the backplane bus.

Keywords. Hierarchical multiprocessor, scalability; shared-memory shared-bus multiprocessor; message-passing; distributed memory.

1. Introduction

In recent years, designers of high-performance workstations and file servers have started turning to multiprocessors for their new generation products. In this development, the shared-memory shared-bus configuration prevails due to its hardware simplicity and cost-effectiveness. However, the shared-memory shared-bus configuration also suffers a serious deficiency of very limited scalability, generally up to 10 to 20 CPUs, due to limited bandwidth of the shared bus. Since ever-increasing processing power is in demand for large database/knowledgebase computing and transaction processing, design of highly scalable multiprocessors is of great significance and interest to computer architects [1]. Motivated by this observation, we started the M^2 hierarchical multiprocessor project in Spring 1991.

In the design of the M^2, several guidelines were cautiously observed. These guidelines are:

(1) The machine is intended to run database/knowledgebase and transaction processing applications. Hence, the design should be optimized for that kinds of workloads. Typical workloads comprise a large number of parallel tasks that are either weakly coupled or even completely independent. The parallel tasks could come from a number of requesters that take action at the same time or from a complicated job that are decomposed into a number of pieces. Each task may further exhibit some degree of lower level parallelism.
(2) The architecture design should exploit latest advances in VLSI and packaging technologies so that the machine is superior in respect to cost/effectiveness.
(3) The design should accord with the pragmatic correspondence illustrated in *Fig. 1* between the granularity of parallel processing and the level of resource sharing among parallel hardware units.

With these three guidelines, the M^2 architecture is designed with two levels of multiprocessing hierarchy. At the first level of the hierarchy, the shared-memory shared-bus structure is employed. At the second level of the hierarchy, a physically distributed with no remote access memory organization is employed. Through the employing of the hierarchical structure and memory organizations, the M^2 features a much higher degree of

* *Corresponding author.*

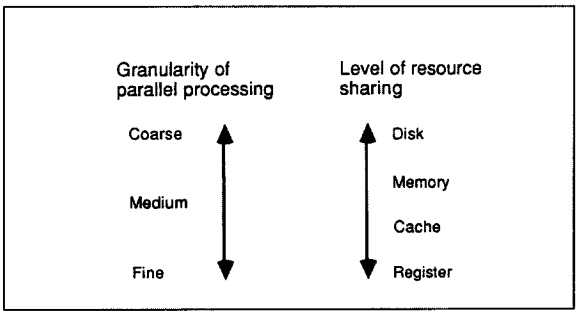


Fig. 1. Pragmatic correspondence between the granularity of parallel processing and the level of resource sharing among parallel hardware units.

scalability than the shared-memory shared-bus architecture and a good performance/cost ratio.

In the following part of this paper, we will elaborate the architecture and design decisions of the M^2 in Section 2. Then, in Section 3, we will describe a prototype M^2 that we have been developing. Finally, we will conclude our discussion in Section 4.

2. The M^2 architecture and design decisions

2.1. Overview of the M^2 architecture

Figure 2 depicts the block diagram of the M^2 hierarchical multiprocessor. The M^2 architecture consists of two levels of multiprocessing hierarchy. At the first level of the hierarchy, multiple CPUs, each with a private cache, and a shared cluster memory are placed on a printed-circuit board and connected through an on-board snooping bus to form a CPU cluster. In the M^2, the shared cluster memory in a CPU cluster serves as the main memory to the CPUs in the cluster. Therefore, structure-wise, there is no difference

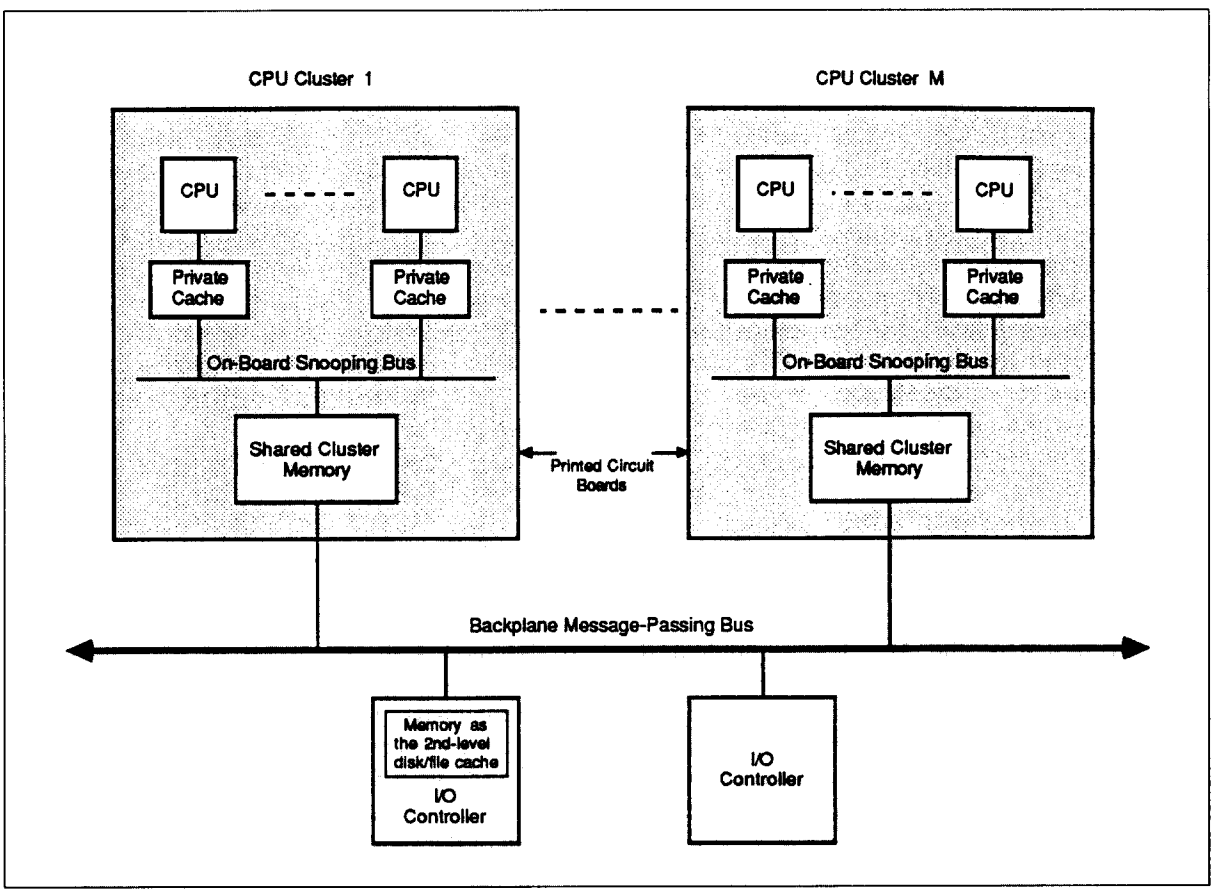


Fig. 2. Block diagram of the M^2 hierarchical multiprocessor.

between a M^2 CPU cluster and a conventional shared-memory shared-bus multiprocessor.

The second level of the M^2 hierarchy is made up of multiple CPU clusters connected through a backplane message-passing bus. At this level of hierarchy, memory is distributed in both physical and logical senses. That is, the memory of a cluster is accessible only to the cluster itself and is not accessible to other clusters. Communication between clusters is carried out through passing messages.

In the M^2, also connected to the backplane message-passing bus are I/O controllers. The I/O controllers and the CPU clusters operate under a client-server model [3]. Some I/O controllers, e.g. disk controllers, are associated with a large memory which serves as the disk/file cache. In such cases, the memory in the CPU cluster and the memory associated with the I/O controller constitute a two-level disk/file cache. Data consistence between the memories in the CPU clusters and I/O controllers is maintained through executing a directory-based coherence protocol [4].

2.2. *Architectural features and design considerations*

This subsection elaborates the main features and design considerations of the M^2 architecture. If compared with other hierarchical multiprocessors [5,6], the M^2 is distinctive in its memory organization at the second level of the hierarchy. In the M^2, a physically distributed with no remote access memory organization is employed. This design is aimed at avoiding severe page-swapping-induced inter-CPU interference. The page-swapping-induced inter-CPU interference, as illustrated in *Fig. 3*, is an inheritance of memory sharing among CPUs. For a group of CPUs that share memory, regardless of if the memory is physically distributed or not, each CPU must be notified with the page-swapping events occurring in the shared memory so that the CPU would flush the blocks that are cached in its private caches from the page being swapped out and update its TLB (Translation Lookaside Buffer) contents accordingly. Since the page-swapping-induced inter-CPU interference grows linearly with the number of CPUs that share memory, it is inappropriate to employ the shared-memory approach beyond a certain extent. Therefore, it was determined that a physically distributed with no remote access memory scheme should be employed at the second level of the M^2 hierarchy.

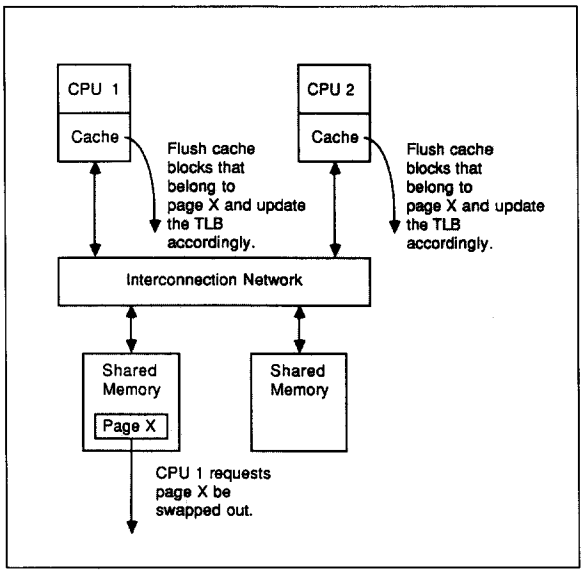


Fig. 3. Illustration of inter-CPU interference caused by page-swapping events.

Nevertheless, the presence of the page-swapping-induced inter-CPU interference does not imply that a shared-memory design should not be used in any case. The shared-memory approach is still favorable up to certain extent due to its hardware simplicity and cost-effectiveness. This is the reason why VLSI chip sets that implement shared-memory shared-bus multiprocessors have become popular lately. Aiming to take advantage of this development, we decided to employ the shared-memory shared-bus structure at the first level of the M^2 hierarchy.

In the M^2, a CPU cluster is to be built on one printed-circuit board. This is intended to effectively exploit the high degree of integration capacity made available by recent advances in VLSI and packaging technologies. As of today, a typical-size board can accommodate 2 to 4 CPUs. With continuous advance in VLSI and packaging technologies, a typical-size board will eventually be able to accommodate 10 to 20 CPUs before the limit imposed by the bandwidth of the shared bus is reached.

One important observation on the structure of the M^2 is that it is basically the same as a group

of multiprocessors connected through a local area network. However, the M^2 is superior in system scalability since a backplane bus offers much higher communication bandwidth than a local area network. For example, a 256-bit Futurebus + can transfer up to 3.2 gigabytes, equivalent to 25.6 gigabits, of data per second. On the other hand, a FDDI network, as of today, can transfer 100 megabits of data per second and may be upgraded to 200 megabits per second in the near future, which is still orders of magnitude lower than the bandwidth of the Futurebus + .

As far as the scalabillty of the M^2 architecture is concerned, there are two limiting factors as discussed in the following.

(1) The first limiting factor is the physical dimension of the message-passing backplane bus. Nowadays, a typical message-passing backplane bus, e.g. the Multibus II [7] and Futurebus [8], can accommodate 20 or so printed-circuit boards. If each CPU cluster, which is to be implemented on a single printed-circuit board, contains 10 to 20 CPUs, then the total number of CPUs that a M^2 system can accommodate could be as high as 200 to 400 CPUs. Here, the assumption of incorporating 10 to 20 CPUs on one printed-circuit board will be feasible in next several years as the new generation of VLSI and packaging technologies offers higher integration capacity.

(2) The second limiting factor is the bandwidth of the backplane bus. In order to determine this effect, one must first figure out the average amount of traffic a CPU would introduce on the backplane bus. Since the amount of I/O traffic varies largely with the nature of the application software, we will base our analysis here on the data collected by Allen Smith [10] for the benchmark set he used. If we assume that the number of I/O transactions issued by a CPU per unit of time grows linearly with the CPU processing power [9], then, according to the statistical data collected by Smith [10], a 50-MIPS CPU would issue about 1000 I/O transactions per second. If we further assume that 60% to 80% of I/O transactions hit the disk/file cache implemented in the cluster memory, which is a typical ratio according to studies on file cache behavior [11,12], and that the disk/file cache uses 16-Kb blocks, then each CPU would introduce 3.2 to 12.8 megabytes of traffic on the backplane bus per second. Given this value and that a 256-bit Futurebus + can transfer up to 3.2 gigabytes of data per second, one can expect that the scale of a M^2 system can be up to hundreds of CPUs as far as the limitation imposed by the bandwidth of the backplane bus is concerned.

The discussion above concludes that the maximum scale of a M^2 system is in the order of hundreds of CPUs, which is an order of magnitude larger than the typical scale of a shared-memory shared-bus multiprocessor.

The last note on the M^2 architecture is that there is a natural match between the M^2 architecture and the architecture of the Mach operating system [2]. In the Mach, threads within a task are sharing-resource light-weight parallel entities. In the M^2, the CPU cluster, with multiple CPUs and a shared cluster memory, provides a good execution platform for multiple-thread tasks. At the higher level, Mach tasks, the heavy-weight

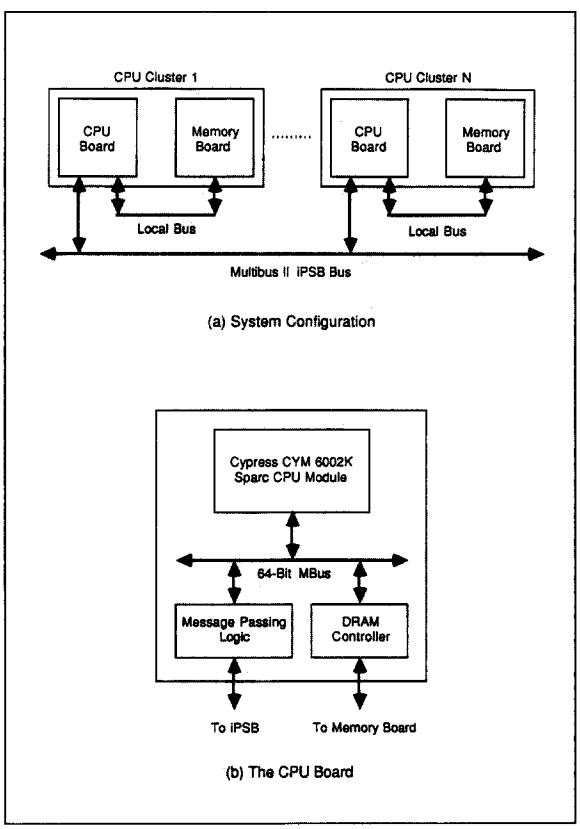


Fig. 4. Hardware block diagram of the prototype M^2.

parallel entities, can be dispatched to M^2 CPU clusters for parallel execution.

3. Development of a prototype M^2

This section describes a prototype M^2 system under development. The hardware design of the system is presented in 3.1 while the operating system issues are elaborated in 3.2.

3.1. Hardware design

Figure 4 shows the hardware block diagram of the prototype M^2 system. In the prototype machine, the Multibus II [7] is employed as the backplane message-passing bus and each CPU cluster comprises 2 Sparc CPUs [14] along with a 64-megabyte cluster memory. The CPUs and the cluster memory are actually placed on two separate boards, the CPU board and the memory board. The CPU board comprises the CPUs, floating point coprocessors, cache controllers, cache memories, DRAM controller, and message-passing control logic. The memory board comprises purelly DRAM chips. The CPU board and the memory board are connected through a local bus separate from the Multibus II.

Figure 4(b) shows the block diagram of the CPU board. The major functional blocks on the CPU board are designed around an on-board 64-bit MBus [15]. Placed on the upper half of the board is the Cypress CYM6002K CPU module [16]. The CYM6002K consists of 2 Sparc CPUs along with their floating-point coprocessors, cache controllers, and cache memories. Placed on the lower half of the board is the DRAM controller and message-passing control logic. The message-passing control logic is mainly made up of three microcontrollers, an Intel 82389 message passing coprocessor (MPC) [17], an Intel 82380 DMA controller, and an Intel 8751 microcontroller. The detailed design of the message-passing control logic is elaborated in the MPC User's Manual from Intel Corporation [17].

3.2. Operating system

The prototype M^2 will run the Mach operating system [2]. On the prototype M^2, Mach tasks are dispatched to CPU clusters in their entirety. In other words, the threads within a task are dispatched only to the CPUs of the cluster that the task is dispatched to and will not spread to other CPU clusters. The reason to adopt this strategy is that, as mentioned earlier, the CPU cluster provides a natural execution platform for multiple-thread tasks. For tasks that are dispatched to different CPU clusters, the inter-task communication is carried out over the backplane message-passing facility.

4. Conclusion

In this paper, we elaborated the development of the M^2 hierarchical multiprocessor. The development of the M^2 is aimed at meeting the demand of ever increasing processing power for database/knowledgebase computing and transaction processing. The M^2 features:

- Scalable up to hundreds of CPUs, which is an order of magnitude larger than the scalability of the shared-memory shared-bus architecture.
- Effectively exploiting the high degree of integration capacity made available by recent advances in VLSI and packaging technologies.
- Matching the architecture of modern multiprocessor operating systems such as Mach.

References

[1] S. Thakkar et al., new directions in scalable shared-memory multiprocessor architectures, *IEEE Comput.* (June 1990).

[2] A. Tevanian Jr., Architecture-independent virtual memory management for parallel and distributed environments: the mach approach, Ph.D. Thesis, Dept. of Computer Science, Carnegie- Mellon University, 1987.

[3] E. Levy and A. Silberschatz, Distributed file systems: Concepts and examples, *ACM Comput. Surveys* 22 (4) (Dec. 1990).

[4] D. Chaiken, C. Fields, K. Kurihara and A. Agarwal, Directory-based cache coherence in large-scale multiprocessors, *IEEE Comput.* 23 (6) (June 1990).

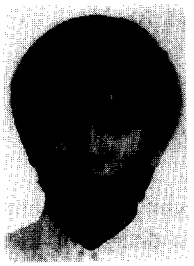
[5] A.W. Wilson, Hierarchical cache/bus architecture for shared memory multiprocessors, *Proc. 14th Annual Internat. Symp. on Computer Architecture* (1987).

[6] D. Cheriton, H.A. Goosen and P.D. Boyle, Paradigm: A highly scalable shared-memory multicomputer architecture, *IEEE Comput.* (Feb. 1991).

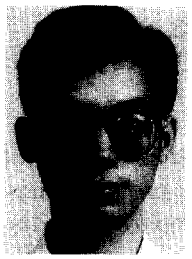
[7] Intel Corporation, *Multibus II Bus Architecture Specification Handbook* (Intel Corporation, 1984).

[8] IEEE, *IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Future-bus*, IEEE Standard 896.1 (1987).
[9] G.M. Amdahl, Storage and IO parameters and system potential, *Proc. IEEE Computer Group Conf.* (1970).
[10] Disk cache – miss ratio analysis and design considerations, *ACM Trans. Comput. Syst.* 3 (3) (Aug. 1985).
[11] J.L. Hennessy and D.A. Patterson, *Computer Architecture: A Quantitative Approach* (Morgan Kaufmann, San Mateo, CA, 1990).
[12] M.G. Baker, J.H. Hartman, M.D. Kupfer, K.W. Shirriff and J.K. Ousterhout, Measurements of a distributed file system, *Proc. 13th ACM Symp. on Operating System Principles*, Pacific Grove, CA (Oct. 1991).
[13] R.H. Katz, S.J. Eggers, D.A. Wood, C.L. Perkins and R.G. Sheldon, Implementating a cache consistency protocol, *Proc: 12th Annual Internat. Symp. on Computer Architecture* (1985).
[14] Cypress Semiconductor, *Sparc RISC User's Guide* (Cypress Semiconductor, 1990).
[15] Cypress Semiconductor, *Sparc MBus Interface Specification*, (Cypress Semiconductor, 1991).
[16] Cypress Semiconductor, *CYM6002K Dual CPU SparcCore Module* (Cypress Semiconductor, 1991).
[17] Intel Corporation, *MPC User's Manual* (Intel Corporation, 1986).

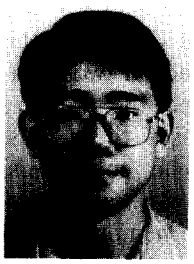
Yen-Jen Oyang received the B.S. degree in Information Engineering from National Taiwan University in 1982, the M.S. degree in Computer Science from the California Institute of Technology in 1984, and the Ph.D. degree in Electrical Engineering from Stanford University in 1988. He is currently an Associate Professor in the Department of Computer Science and Information Engineering, National Taiwan University. His research interests include computer architecture and VLSI system design.



David J. Sheu was born on September 11, 1967 in Taipei, Taiwan. He received the B.S. degree in Information and Computer Engineering from Chung-Yuan Christian University in 1990, and the M.S. degree in Computer Science and Information Engineering from National Taiwan University in 1992. His major research interest is multiprocessor architecture and hardware design.



Chih-Yuan Cheng was born on October 19, 1968 in Taipei, Taiwan, R.O.C. He received the B.S. degree in Computer Science and Information Engineering from National Taiwan University in 1991. He is currently in the master program of the same department and expected to complete his degree in 1993. His research field is superscalar architecture and compiler design.



Cheng-Zen Yang is currently a Ph.D. student at the Dept. of Computer Science and Information Engineering of National Taiwan University, Taipei, Taiwan. He received the B.S. degree in Computer Engineering from National Chiao Tung University in 1988, and the M.S. degree in Computer Science and Information Engineering from the same University in 1990. His major research interests include distributed computing system, distributed file system and Operating System.