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 and exploits parallelism at both medium- and coarse-grain levels. 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.

1 Introduction

The shared-memory shared-bus architecture has been prevalent in multiprocessor design in recent years due to its hardware simplicity and simple programming model. However, the shared-memory shared-bus architecture also suffers a serious disadvantage of very limited scalability, generally up to tens of CPUs, due to limited bandwidth of the shared bus. Motivated by this observation, computer architects have been actively investigating novel multiprocessor architectures that feature higher degree of scalability than the shared-memory shared-bus architecture [1,2]. In this paper, we will present our approach concerning this active issue. We named the prototype machine we have been developing the M^2 hierarchical multiprocessor.

The architecture of the M^2 is derived with acknowledging a pragmatic correspondence between the granularity of parallel processing and the level of resource sharing among parallel hardware units. Figure 1 illustrates the observed pragmatic correspondence. Based on this observation, the M^2 is designed with two levels of multiprocessing hierarchy. At the first level of the hierarchy, the shared-memory shared-bus scheme 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 design achieves three major goals:

1. Derive a multiprocessor architecture that features much higher degree of scalability than the shared-memory shared-bus architecture and exploits parallelism at both medium- and coarse-grain levels.

2. Effectively exploit the high degree of integration capacity made available by recent advances in VLSI and packaging technologies.

3. Match the architecture of modern multiprocessor

*This research was sponsored by National Science Council of R.O.C. under grant NSC 81-0408-E-002-17

0-8186-2760-3/92 $03.00 © 1992 IEEE

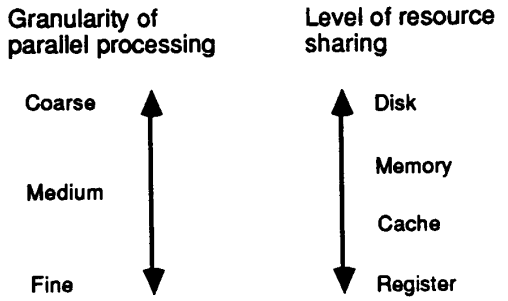


Figure 1: Pragmatic correspondence between the granularity of parallel processing and the level of resource sharing among parallel hardware units

operating systems.

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 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. 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.

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 [3,4], 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. If a shared memory scheme was adopted at the second level of the hierarchy, no matter whether the memory is physically distributed or not, then every CPU in the system would have to be notified with every page-swapping event occurring in the shared memory so that the CPU would flush the cache blocks that are cached in its private cache 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 was determined that a physically distributed with no remote access memory scheme should be employed at the second level of the M^2 hierarchy.

Nevertheless, the presence of the page-swapping-induced inter-CPU interference does not means a shared-memory design should not be used in any case. The shared-memory architecture is still a favorite scheme up to certain extent because the sever-

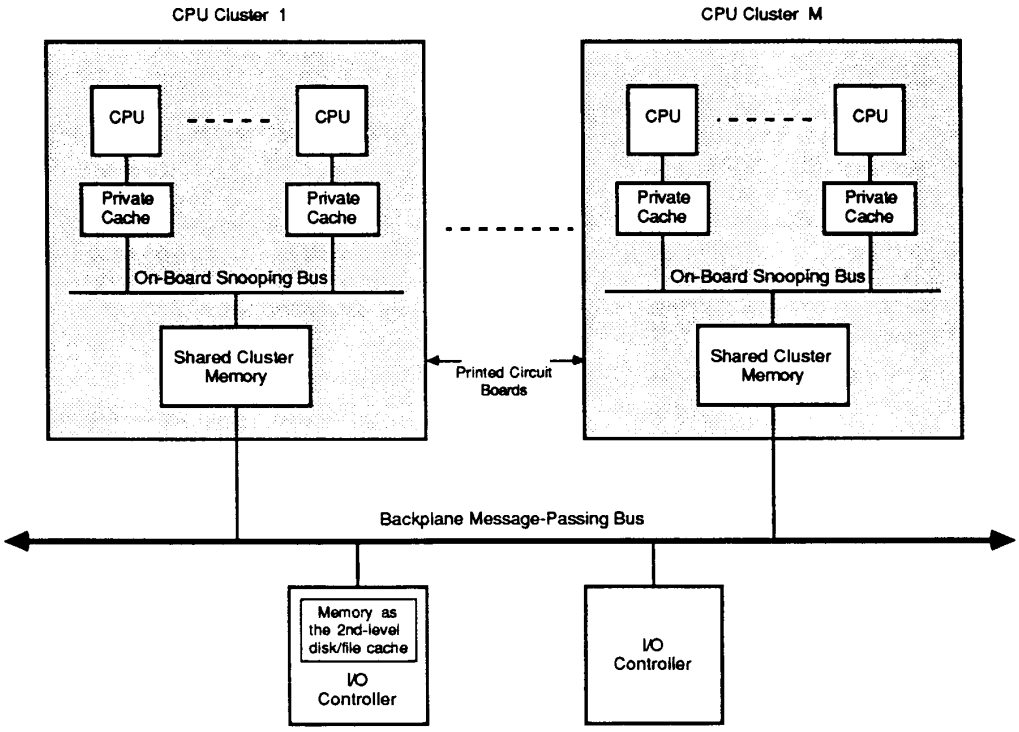


Figure 2: Block diagram of the M^2 hierarchical multiprocessor

ity of the page-swapping-induced interference is a linear function of the number of CPUs that share memory. Therefore, it was 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 a printed-circuit board. This is aimed at effectively exploiting the high degree of integration capacity made available by recent advances in VLSI and packaging technologies. As of today, 2 to 4 CPUs can be incorporated in one CPU cluster. With continuous improving of VLSI and packaging technologies, a typical-size printed-circuit board will eventually be able to accommodate 10 to 20 CPUs. It is unlikely that the number will go much further due to the limitation imposed by the bandwidth of the shared bus.

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 smaller than the bandwidth of the Futurebus+.

As far as the scalability 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. Nowa-

days, a typical message-passing backplane bus, e.g. the Multibus II [5] and Futurebus [6], 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 degree of integration capacity.

2. The second limiting factor is the bandwidth of the backplane bus. In order to determine its effect, one must first figure out the average amount of traffic a CPU would introduce on the backplane bus. If it is assumed that the number of I/O transactions issued by a CPU per unit of time grows linearly with the CPU processing power [7], then, according to the statistical data collected by Smith [8], a 50-MIPS CPU would issue about 1000 I/O transactions per second. If it is further assumed that 60% to 80% of I/O transactions hit the file cache implemented in the cluster memory, which is a typical ratio according to studies on file cache behavior [9,10], and that the file cache uses 16-KByte 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 with respect to the limitation imposed by the bandwidth of the backplane bus.

The analyses above suggest that a M^2 system can accommodate 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 [11]. 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 memory, provides a good execution platform for multiple-thread tasks. At the higher level, Mach tasks, the heavy-weight parallel entities, can be dispatched to M^2 CPU clusters for parallel execution.

3 Development of a Prototype M^2

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

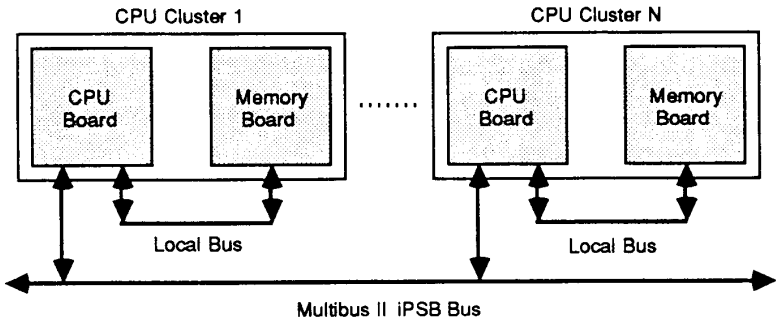
3.1 Hardware Design

Figure 3 shows the hardware block diagram of the prototype M^2 system. In the prototype machine, the Multibus II [5] is employed as the backplane message-passing bus and each CPU cluster comprises 2 Sparc CPUs [12] 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, and message-passing control logic. The memory board comprises the memory controller and memory chips. The CPU board and the memory board are connected through a local bus separate from the Multibus II.

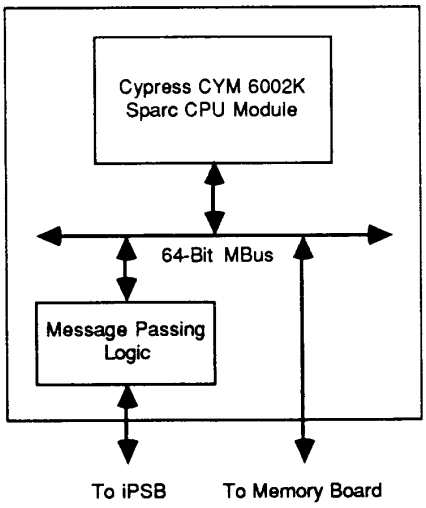
Figure 3(b) shows the block diagram of the CPU board. The major functional blocks on the CPU board are designed around an on-board 64-bit MBus [13]. Placed on the upper half of the board is the Cypress CYM6002K CPU module [14]. The CYM6002K consists of 2 Sparc CPUs along with their floating-point coprocessors, cache controllers, and cache memories. Placed on the lower half of the board is the message-passing control logic. The message-passing control logic is mainly made up of three microcontrollers, an Intel 82389 message-passing coprocessor (MPC) [15], 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 [15].

3.2 Operating System

The prototype M^2 will run the Mach operating system [11]. On the prototype M^2, Mach tasks are dis-



(a) System Configuration



(b) The CPU Board

Figure 3: Hardware Block Diagram of the prototype M^2

patched 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 discussed the major architectural features of the M^2 hierarchical multiprocessor and the reasons behind the design decisions. The design of the M^2 achieves three major goals:

1. Derive a multiprocessor architecture that features much higher degree of scalability than the shared-memory shared-bus architecture and exploits parallelism at both medium- and coarse-grain levels.
2. Effectively exploit the high degree of integration capacity made available by recent advances in VLSI and packaging technologies.
3. Match the architecture of modern multiprocessor operating systems.

References

[1] P. Stenstrom, "A Survey of Cache Coherence Schemes for Multiprocessors", IEEE Computer, June, 1990.

[2] S. Thakkar etal., "New Directions in Scalable Shared-Memory Multiprocessor Architectures", IEEE Computer, June, 1990.

[3] A. W. Wilson, "Hierarchical Cache/Bus Architecture for Shared Memory Multiprocessors", Proc. of the 14th Annual International Symposium on Computer Architecture, 1987.

[4] D. Cheriton, H. A. Goosen, and P. D. Boyle, "Paradigm: A Highly Scalable Shared-Memory Multicomputer Architecture", IEEE Computer, Feb., 1991.

[5] Intel Corporation, *Multibus II Bus Architecture Specification Handbook*, Intel Corporation, 1984.

[6] IEEE, *IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Futurebus*, IEEE Standard 896.1, 1987.

[7] G. M. Amdahl, "Storage and IO Parameters and System Potential", Proc. of the IEEE Computer Group Conference, 1970.

[8] "Disk Cache - Miss Ratio Analysis and Design Considerations", ACM Trans. on Computer Systems, Vol. 3, No. 3, Aug., 1985.

[9] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann Publishers, San Mateo, California, 1990.

[10] M. G. Baker, J. H. Hartman, M. D. Kupfer, K. W. Shirriff, and J. K. Ousterhout, "Measurements of a Distributed File System", Proc. of the 13th ACM Symposium on Operating System Principles, Pacific Grove, California, October, 1991.

[11] 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.

[12] Cypress Semiconductor, *Sparc RISC User's Guide*, Cypress Semiconductor, 1990.

[13] Cypress Semiconductor, *Sparc MBus Interface Specification*, Cypress Semiconductor, 1991.

[14] Cypress Semiconductor, *CYM6002K Dual CPU SparcCore Module*, Cypress Semiconductor, 1991.

[15] Intel Corporation, *MPC User's Manual*, Intel Corporation, 1986.