MARS Performance Evaluation with Different Interconnection Networks

Feipei Lai ,Lea-Ming Tzeng, Thom-Ling Chang and Tai-Ming Parng

Department of Electrical Engineering & Computer Science
National Taiwan University
Taipei, Taiwan, R. O. C

Abstract

MARS is a multiprocessor system designed at National Taiwan University. In this paper, we estimate the performance of MARS with shared-bus and hierarchical multiple-bus. The performance estimation was carried out through queuing models, which were further simplified by used of modeling. Not only the effects of the shared data memory accesses but also the interactions among processors were addressed in our models. We simulated four variants of shared-bus system to analyze the performance improvement resulting from write-buffer and interleaved memory under our cache coherence protocol, Phoenix. Given write-buffer for ten-processor, the system utilization can be improved 34.8%. Memory interleaving in the ten-processor case can also boost the original performance by 154.2%. In the hierarchical multiple-bus system, eight processors are grouped as a cluster and packet-switching is used to transfer data. Data consistency is achieved by keeping shared data noncacheable.An analysis on system performance sensitivity which varies with parameters is conducted. Based on this, we further propose several guidelines for the design of cache coherence protocol.

1. Introduction

Designers of computer system are often interested in predicting what a proposed machine organization will behave, before that organization is actually constructed. This preliminary performance evaluation serves several useful purposes; for example, the designer can evaluate tradeoffs in the choices of hardware, evaluate alternative in load balancing algorithms, and identify potential bottlenecks.

MARS is a multiprocessor system designed at National Taiwan University. The performance of MARS with two types of interconnection network-shared bus and and hierarchical multiple bus- is estimated in this paper.

MARS is a multiprocessing system, which has a number of processor boards [1]. Each processor board is linked together via an interconnection network as depicted in Figure 1. Inside each board, there are CPU chips, i.e., IFU, (Instruction Fetch Unit) and IPU (Integer Processing Unit) as well as special chips, FPU (Floating-point Processing Unit) and LPU (List Processing Unit), dedicated to floating point and list operations, separate Instruction Cache and Data Cache along with the Cache Controller and Memory Management Unit (CCMMU).

IFU is the buffering and control mechanism between the instruction cache and the datapath chips (IPU, FPU, LPU). It is designed to interleave instruction fetch and execution, and to achieve coordinated operations among IPU, FPU, and LPU, while IPU [2] retains the integer datapath and some control parts of a common RISC CPU [3~5], performing integer arithmetic, shift, logical operations, and address calculation for data operands of all datapath chips. LPU [6] provides hardware primitives for list processing, such as car, cdr, cons, rplaca (replacea), rplacd (replaced). CCMMU [7] is responsible for the operation of local data cache on each processor board, address translation, and data cache coherence protocol among processors. Local data cache will be as large as 64KB and data will be heuristically prefetched in the face of pointer or list.

As a tightly coupled multiprocessor system, MARS must depend on an efficient interconnection network to avoid performance degradation due to memory sharing. Single shared bus is the first consideration because of its low functional complexity and relatively low cost. We survey the multiprocessor system such as RP3 [8~10], CM*[10~11] and Cedar[10], and find that each system puts the memory module into processor or "cluster" to reduce the bus traffic. Concerning how we manage to reduce bus traffic and thus the transaction latency, we interleave memory into each processor board, instead of a lumped global shared memory. From our analysis that interleaving memory into each processor could increase performance about 3%~104%, which is dependent on the number of processors and the other system parameters. Further more, we try to attach write-buffer between processor and bus when write_back or write_to_memory occur, processor can put it to write-buffer and continue to execute without waiting any memory access. In section 3, we conclude from simulation result that adding write-buffer would improve system performance about 2%~29%.

Since MARS with single shared bus can only support 6~16 processors, another interconnection network,i.e. hierarchical multiple-bus, is proposed. In the hierarchical multiple-bus system, every 8 processors with sharing interleaved memory and cache are grouped as a "cluster", just like Cedar cluster [10]. In such interconnection network, even though we connect 64 processors, we can still keep average processor utilization at about 60%~80%.

2. MARS Model

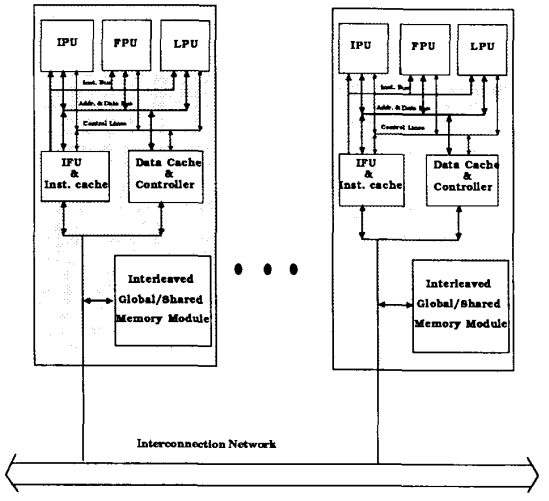
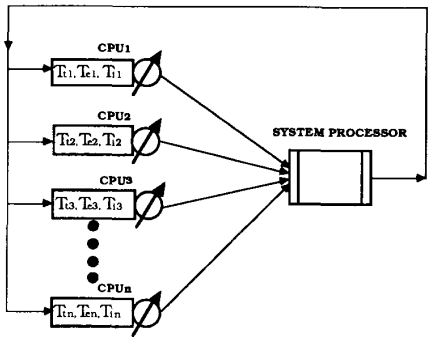


Fig. 1 The Global System Architecture of MARS



$T_{i,n}$: mean internaltime of data reference in CPUn.

$T_{e,n}$: mean internaltime of IFU prefetch instruction miss delay time in CPUn.

$T_{t,n}$: mean internaltime of IFU prefetch data miss delay time in CPUn.

CPUn : a FECS(Flow Equivalent Center Service).

SYSTEM PROCESSOR : a submodel of system , it's function is depenend on system behavier.

Fig. 2 System Level Model

2.1 MARS system Model

In this paper, all the models are described by PAWS - a queuing model simulation tool [12]. We use hierarchical modeling technique [13~15] to simplify the system model. Hierarchical modeling is the process of partitioning a large model into a number of smaller submodels. Each of these submodels is evaluated, and the individual solution is combined to obtain the solution of the original model. In the general case, there is an arbitrarily defined subsystem: *aggregate*, which interacts with the other service centers in the network, called collectively the complement or complementary network. A key step in the hierarchical approach is to replace the entire *aggregate* by a FESC (flow equivalent service centers) that reduplicates its procedures, thereby. Hierarchical modeling is the process of modeling a system by using multiple levels of submodels.

Figure 2 shows the MARS system level model. Each is an FESC of single MARS board. All CPUs are connected to SYSTEM PROCESSOR to simulate their interactions. Tt, Te and Ti stand respectively for the mean residence time of data reference, IFU prefech instruction miss and IFU prefetch data miss in CPU respectively. These residence time in CPU are calculated by our own lower model respectively.

2.2 CPU submodel

The lower model of CPU FESC is shown in Figure 3. CPU submodel contains IFU, OTHERS and IC. IFU is the model of instruction fetch unit. OTHERS represents the other element behavior in processor. IC is the instruction cache model and initialization model. We separate IFU from the other elements because its abstract behavior is different from the other elements. IFU can peep out the branch instruction and fetch the target address before the instruction is executed. When instructions in the write-buffer of instruction fetch unit are out of use, IFU will fetch next sequential block instruction. These prefetch miss signals will trigger IC to send the next block instructions. INS represents load or store operation. IFU submodel has two instruction buffers-BUFFER1 for sequential instruction stream ; BUFFER2 for target instruction stream. While BUFFER1 sends an instruction I1 out, if DCB(Delay Compare and Branch), SCB(Squashable Compare and Branch) or JMP is the next instruction, they will split TARG1 or TARG2 at the same time as I1 departs from buffer. TARG1 and TARG2 will trigger IC submodel to send the next block instruction into BUFFER2. Conditional branch instruction SCB and DCB will go into OTHERS submodel after splitting one TARG1. In OTHERS submodel instructions go through five pipeline stages and decide whether branch occurs or not. If branch occurs, the instructions in BUFFER2 are fetched as the next instructions. BUFFER2 becomes the sequential instruction buffer. BUFFER1 changes into target instruction buffer and SCB must squash the instruction following it. On the other hand, if branch fails, the instruction in BUFFER2 is discarded, and new instruction is fetched from BUFFER1. JMP is a special instruction for OTHERS, it will not be sent to OTHERS but "absorbed" by IFU directly. JMP splits one TARG2 to trigger target instruction and then disappears. Target instruction is put into BUFFER2. The instruction following JMP in BUFFER1 is discarded. When instructions in buffer are run out, IFU will split EMPT to IC to get next block instructions. The parent transaction goes into OTHERS.

OTHERS represents the abstraction behavior of IPU, LPU or FPU. We do not care the detail functions of the three models. Each instruction sent from IFU will go through five pipeline stages (instruction fetching, instruction decoding, arithmetic operation, memory access and writing back). Whether conditional branch occurs or not is decided by probability. According to the statistical data [16], the probability of true condition is 70% and 30% for the false case.

IC submodel accepts the miss signals of IFU to trigger next block instructions. We classify the instructions as 7 types: LD, ST, SCB, DCB, JMP, NOP and REM, where REM *represents the other types of instruction unmentioned so far.* For example, all arithmetic instructions are regarded as REM. The probability of the instruction types is shown in table 1. These data are taken from MIPS project [17].

2.3 Data Cache (DC) Submodel

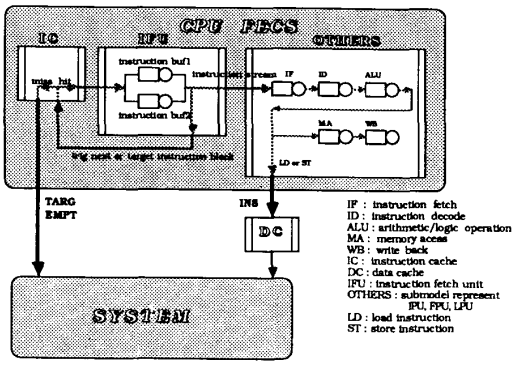


Fig. 3 CPU Functional Diagram

Data cache submodel is similar to the one developed in [18]. The reference stream of each processor is viewed as the merging of two reference streams: the references to shared blocks with probability *shd*, and the references to private blocks. In the DC submodel an explicit representation is chosen for shared blocks, whereas the presentation of private block reference is probabilistic. For private block the reference nature is the same for both the uniprocessor and the multiprocessor, and it is therefore possible to use existing uniprocessor cache measurements to reflect actions resulting from private block references [18~20]. All references to shared blocks in our model include a specific block number, and actions are taken according to the actual state of that block. If the request refers to a shared block, the block number of the reference is determined using a least recently used (LRU) stack. To meet a shared block request, the cache is determined according to a table where the requested block is present. If a cache miss occurs, either for shared block or for a private block, a block must be ejected to make room for the new block. The probability for a shared block to be selected as replacement block is proportional to the percentage of the shared blocks in the cache. The same method is used to select a private blocks as a replacement block. If the selected block is private, it is modified and written back with probability md. Our cache coherence protocol is similar to DRAGON [7][18]. The simulation parameters and range are shown in table 1, which is got from [18].

3. MARS with Shared Bus

3-1. System Architecture and Operation

Shared-bus multiprocessor system is very popular because of its low cost and simplicity. There are four variants of MARS shared-bus system to be compared and analyzed. They are the shared global memory system with or without write-buffer and the distributed global memory systems with or without write-buffer. Architectures of the four variants are shown in Figure 4.

Our cache coherence protocol is called Phoenix [7] ,which is similar to Dragon except two major differences. First, in Dragon protocol, a cache will supply data only when its copy is shared dirty or dirty. In our protocol, data can also be provided by the idle cache when the block state is shared clean or valid exclusive. Hence, we have more chances

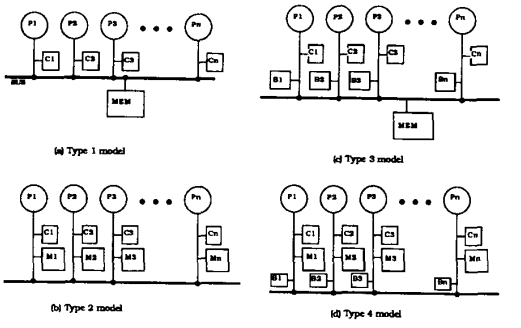


Fig. 4 The variant Shared-Bus System

receiving data from cache instead of the memory to reduce the data access time. Second, when write miss occurs, the block state is marked with shared dirty since we have no idea about the shared value of the block at this moment. The state might be changed after the data has been broadcasted.

In the four variants shared-bus system, the type 1 system has the worst performance. The type 2 system which has interleaved memory can reduce part of the bus traffic caused by cache misses. How much we can benefit from interleaving memory module depends on the scheme of memory allocation. The more interleaving locality we have, the more improvement we can make. The type 3 system adds a write-buffer to the type 1 system. The type 4 system add a write-buffer to the type 2 system. The write-buffer can reduce the idle time wasted on memory access. When a processor needs to write data, it puts the instruction into the write-buffer and then processes the next instruction. The operation of writing back cache replacement block to memory is the same as the store operation. In such case, writing back to memory and broadcasting are managed by the write-buffer; thus processors are free to process other tasks.

3-2 The Model of Shared Bus System

Basically, the four variants are alike. We only describe type 4 model - shared-bus with buffer and interleaved memory, because the other three can be easily obtained with minor modifications In this model, we make some simplification and assumptions.

1.) Bus arbitration is asynchronous and FCFS.

2.) Synchronizations in parallel programming are not taken into consideration in the model. From the memory reference behavior of several parallel applications running under MACH operation at Stanford University [21], we know the references to write-shared block have low temporal locality and references to shared blocks by different processors are usually interleaved well. Therefore, the assumption is reasonable.

3.) In type 3 and type 4 system models, buffer size is assumed to be infinite.

In the type 1 model, all memory accesses reference to main memory is equivalent to the type 2 model on condition that the parameter of PMEH (Private Memory Element Hit) is zero. The same as above, type 3 model is equivalent to type 4 model with PMEH parameter being zero. The type 1 and 2 model are similar to type 3 and 4 model.

3-3 Simulation Result and Analysis

Type 1 model is the worst case. Since among 4 type models for single-bus system , type 1 all reference misses request the bus. Type 2 model is improved by interleaving memory module to each processor, and this interleaved memory modules can filter out some bus requests. Write-buffer in type 3 and type 4 system can reduce the processor idle time. For each model, PMEH varies from 0.0 to 0.9 to facilitate us to observe the effect of memory allocation.

The processors utility versus processor number are shown in Figure 5. From the above data, we can see the following results:

Type 1 system: When the ratio of data cache hit is 93%, we find the system can support 4 processors at most. If the ratio of data cache hit increases to 97%, the system can support up to 5~6 processors without saturating the system bus.

Type 2 system: From Figure 5.1, we know that the system consisting of 4~5 processors is reasonable if PMEH is less than 20%. When PMEH exceeds 50%, the system can support 8~10 processors reasonably. In Figure 5.2 , HIT is raised from 93% to 97%. While HIT increases to 97%, the system can support 8 processor easily even when PMEH is less than 20%. If PMEH exceeds 50%, the system can support 10~14 processors.

Type 3 system: When the ratio of data cache hit is 93%, we find the system can support up to 6 processors. If data cache hit ratio increases to 97%, the system can support up to 8 processors.

Type 4 system: From Figure 5.3, we know the system consisting of 7~8 processors is reasonable if PMEH is less than 20%. If PMEH exceeds 50%, the system can even support 14 processors reasonably. In Figure 5.4,HIT is raised to 97%. While HIT increases to 97%, the system can support 10~12 processor easily even when PMEH is less than 20%. If PMEH exceeds 50%, we believe that the system can support 12~16 processors approximately.

Figure 6 shows the performance improvement caused by interleaved memory. We use utility increment percentage to observe the effect. We can see the increment percentage is proportional to processor number and PMEH. Obviously, increasing processor number causes serious bus contention, so the effect of interleaving memory module will be more outstanding. There are some decays caused by the saturation of system power. When PMEH exceeds 30%, the increment percentage almost increases linearly. The maximum improvement can reach 154%. In most cases, the improvement can exceed 30%.

Table 2 shows the increment percentage caused by the write-buffer. The increment percentage is proportional to the processor number. When system consists of 10 processors and HIT is 93%, adding write-buffer can increase system power by 24~36%. If HIT increases to 97%, the effect of using write-buffer increased only by 8~29%. From these tables we know that write-buffer is more useful when HIT is low. The queue length of write-buffer is not long. When the number of processors reaches 4, the maximum queue length is 2. When we use 10 processors, the maximum queue length will reach 6. But the throughput is high and average queue length is low, that means the average write buffer queuing time is not long. If we want to support 12~16 processors, the proposed buffer size 4 will be suitable. If the number of processors is less than 4, the buffer size of one or two is good enough.

Table 1 Simulation Parameter

Data cache hit ratio		**93%~97%**	
Instruction cache hit ratio		**98%~99%**	
Pipeline cycle		**50 ns**	
Bus cycle		**100 ns**	
Memory cycle		**200 ns**	
Data cache size		**64k bytes**	
SHD		**0.1%~5%**	
MD	**30%**	**PMEH**	**0.1~0.9**
LDP	**27.8%**	**JMPP**	**3.7%**
STP	**9.3%**	**NOPP**	**2.4%**
SCBP	**2.7%**	**REMP**	**45.9%**
DCBP	**8.2%**		

LDP: probability of load
STP: probability of store
SCBP: probability of squash conditional branch
DCBP: probability of delay conditional branch
JMPP: probability of jump
NOPP: probability of no operation
REMP: probability of the remainder
PMEH: pme memory hit ratio
MD: probability of writing back private data is modified
SHD: probability of reference to shared data

Table 2.1 System Utility Increment Percentage Caused by Write-Buffer (HIT=93%, SHD=5%)

no. of processors \ PMEH %	0.0	0.1	0.15	0.2	0.3	0.5	0.7	0.9
2	2.94	5.14	6.82	7.91	8.84	8.51	9.47	10.31
4	6.41	9.15	9.15	9.26	9.91	9.97	12.92	10.19
6	8.64	12.68	13.66	13.84	18.55	9.98	10.90	10.21
8	22.12	23.80	29.63	28.49	21.90	17.66	11.55	9.05
10	24.36	26.99	36.73	36.71	30.09	28.54	34.86	29.06

Table 2.2 System Utility Increment Percentage Caused by Write-Buffer (HIT=97%, SHD=5%)

no. of processor \ PMEH %	0.0	0.1	0.15	0.2	0.3	0.5	0.7	0.9
2	1.16	2.29	3.41	5.68	6.74	7.26	7.69	7.03
4	10.06	10.48	11.32	11.46	9.55	8.26	5.99	5.88
6	11.93	12.17	11.50	9.23	8.89	6.86	8.92	10.40
8	11.21	16.78	15.23	16.46	8.39	7.38	3.25	10.16
10	10.05	20.51	26.34	29.03	15.02	15.08	10.56	8.18

Because most of the data in a process are private, we conclude that it is not difficult for PMEH to reach 50% or

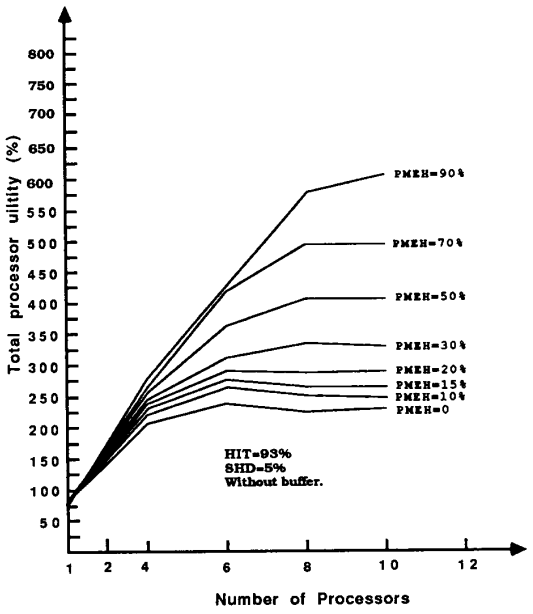


Fig 5.1 Total Processor Utility

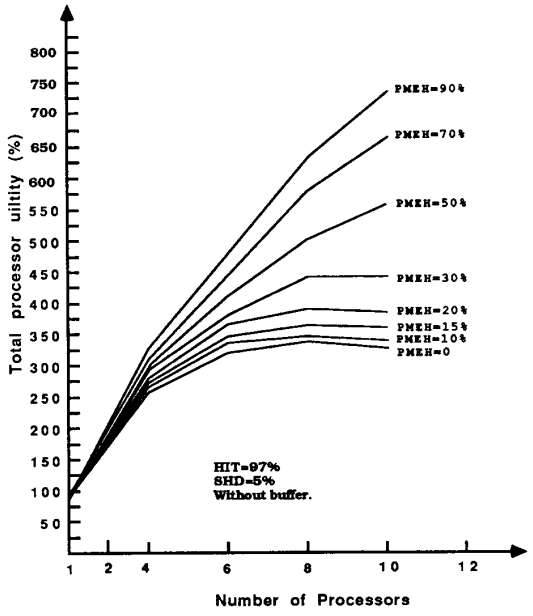


Fig 5.2 Total Processor Utility

even 80%. If this is the case, MARS with shared-bus system can support to 12~16 processors reasonably by using interleaved memory module and write-buffer.

4. MARS with Hierarchical Multiple-Bus

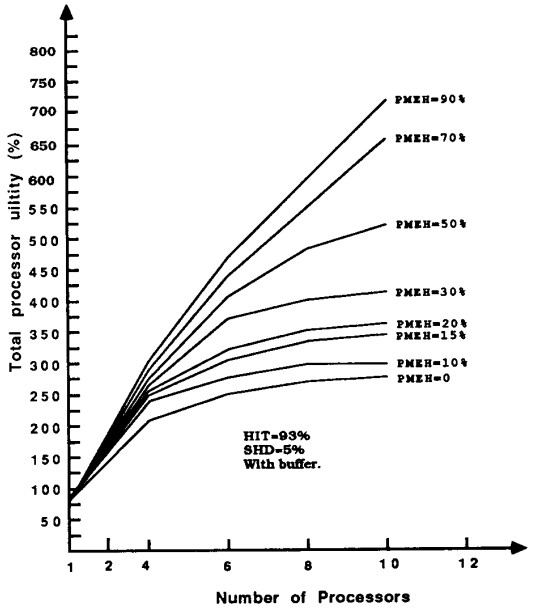


Fig 5.3 Total Processor Utility

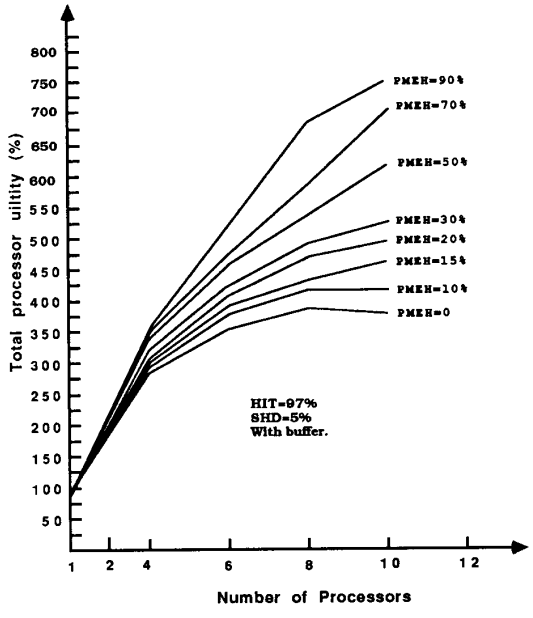


Fig 5.4 Total Processor Utility

4-1. Organization of Hierarchical Multiple-Bus System

The system is shown in Figure 7. Eight MARS processors are grouped as a cluster, CL. Eight processors connected by level 0 multiple-bus are shown in Figure 8.1. Four clusters connected by level 1 multiple-bus, called

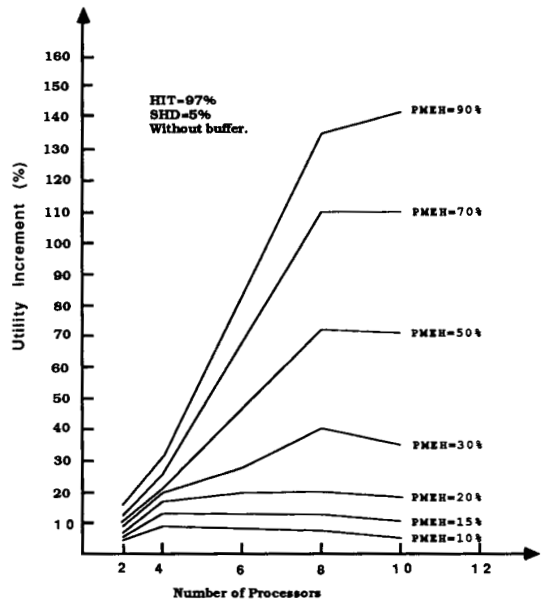


Fig 6.1 System Utility Increment Percentage Caused by Interleaved Memory module.

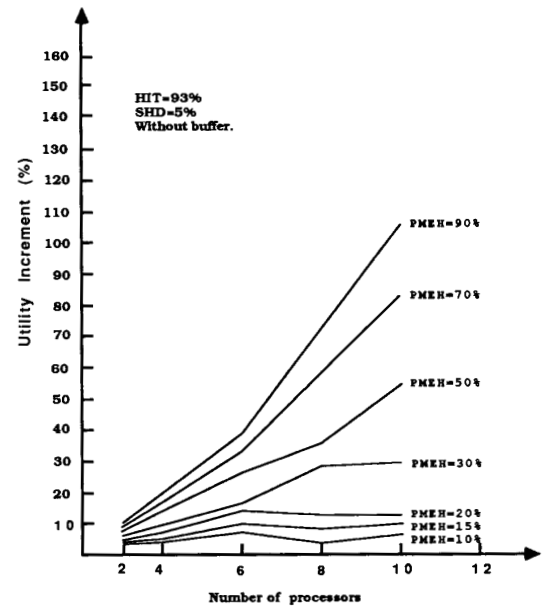


Fig 6.2 System Utility Increment Percentage Caused by Interleaved Memory module.

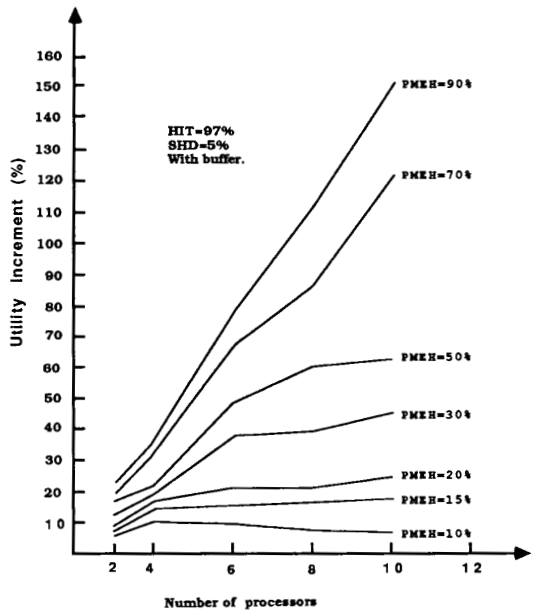


Fig 6.3 System Utility Increment Percentage Caused by Interleaved Memory module.

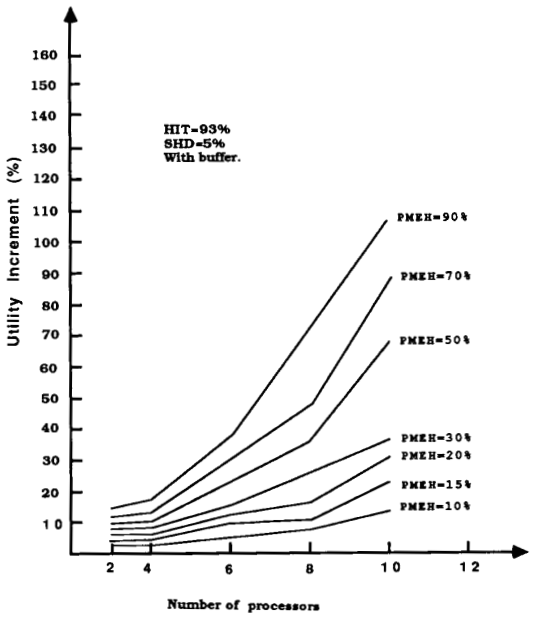


Fig 6.4 System Utility Increment Percentage Caused by Interleaved Memory module.

group, are shown in Figure 8.2. Four groups connected by level 2 multiple-bus as a set are shown in Figure 8.3. Input buffer and output buffer are put between different levels of the multiple-bus. Communications between processors of the same cluster use normal circuit-switch multiple-bus, but communications between clusters use packet-switch method. This is somewhat like CM* bus transform method. In multiprocessor system cache coherence is important, one solution is to cache only those data structures that cannot cause inconsistency. This can be done under software control;

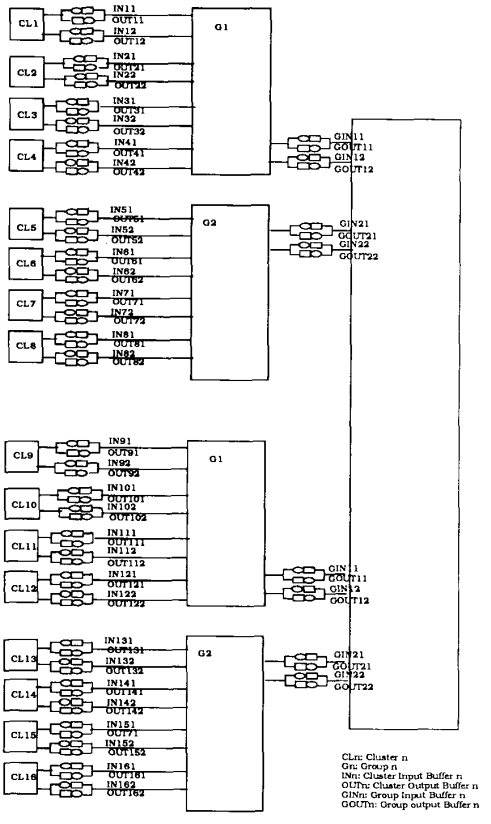


Fig. 7 Hierachical Multiple-bus System

the compiler tags data as cacheable or noncacheable. Because of its simplicity, we use the method to maintain cache coherence.

4-2. The Model of Hierarchical Multiple-Bus System

In this model , the operation of bus arbitration is assumed asynchronous. Packet-switch size is one word. Synchronization in parallel programming is not taken into consideration. The memory cycle is 200ns and bus cycle is 100ns. The time needed to write block into buffer is 100ns.

4-3. Simulation Result and Analysis

There are many factors that affect our system performance. There are PMEH, the ratio of data cache hit(HIT) and the ratio of shared data(SHD).

First we range PMEH from 0.2 to 0.8, assuming that HIT is 93%, 97% and SHD is 5% in order to observe the effect of memory allocation in system performance. When HIT is 93%, the effective uniprocessor utilization curve is plotted in Figure 9.1 and the mean utilization versus number of processors is shown in Figure 9.2. If PMEH is less than 40%, the system will saturate at 40 processors approximately. But the mean processor utilization is only

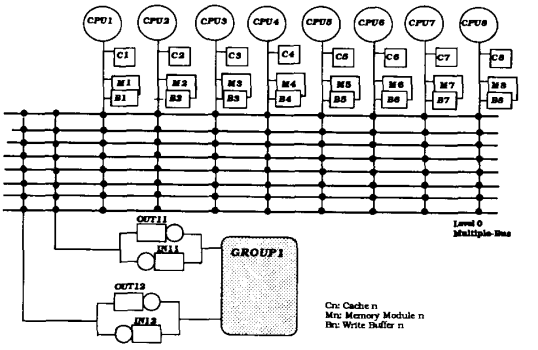


Fig. 8.1 CLUSTER1 (CL1) Submodel

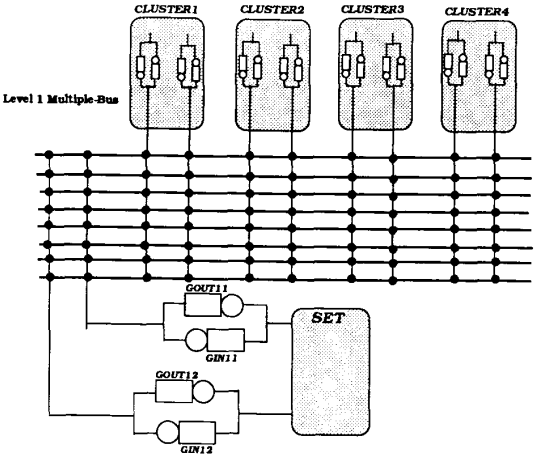


Fig. 8.2 GROUP1 (G1) Submodel

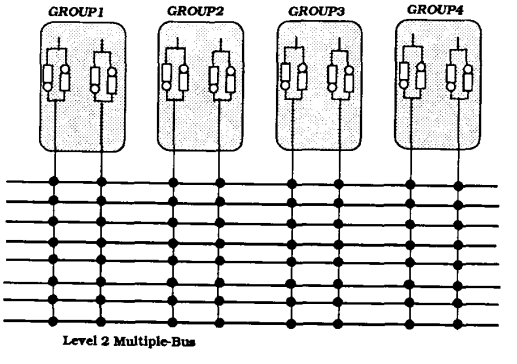


Fig. 8.3 SET Submodel

about 13%~19% which is not acceptable. In such situation, system consisting of 10~12 processors is more reasonable. When PMEH increases to 0.6, system can support 20~24 processors. If PMEH increases to 80%, system performance will be good enough even if it supports 64 processors. From the analysis, we can predict that if data hit ratio is 93% or lower, SHD is 5% or higher, and PMEH is less than 60%. the

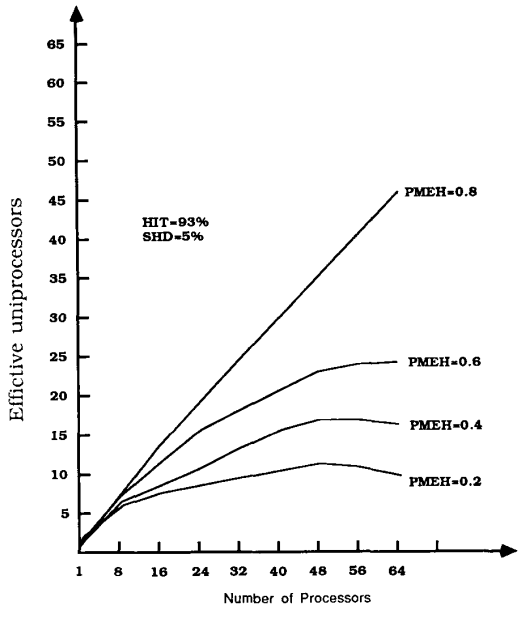


Fig. 9.1 Effective uniprocessors

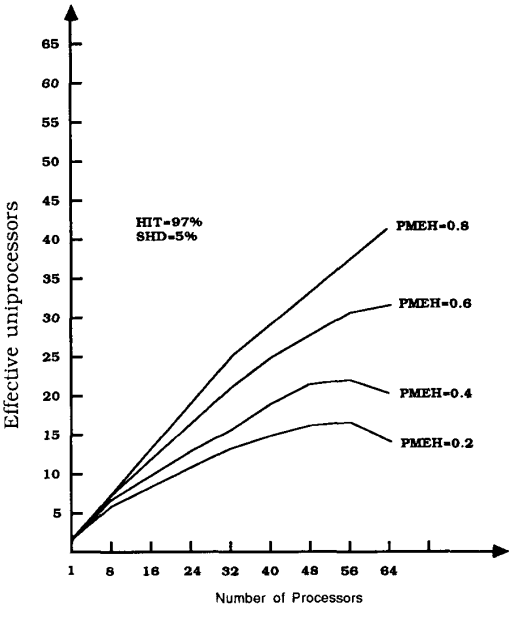


Fig. 10.1 Effective uniprocessors

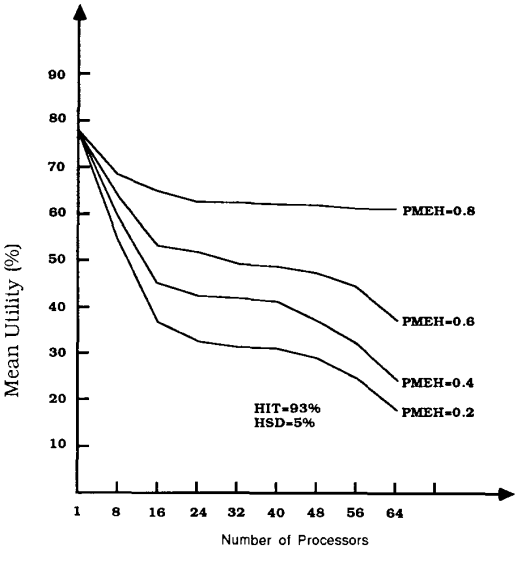


Fig. 9.2 Mean Utility

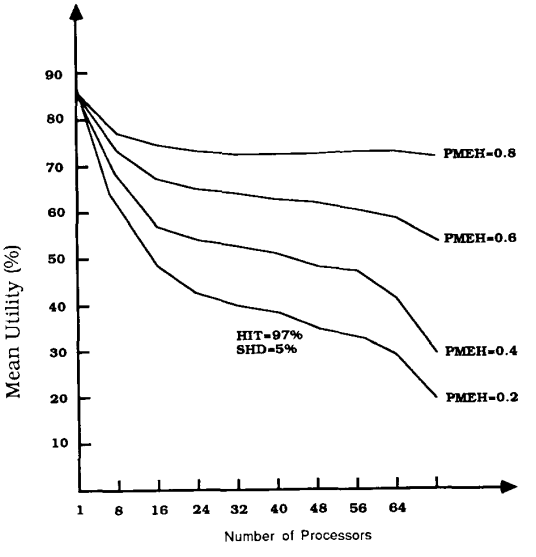


Fig. 10.2 Mean Utility

system would not be as good as we expected. Fortunately, it is rare that HIT is lower than 93%. If data cache hit ratio increases to 97%, as shown in Figure 10.1, the system is saturated at 40 processors when PMEH is less than 40%. If PMEH is less than 40%, system composed of 24 processors is reasonable. If PMEH is higher than 60%, the system can support 64 processors at most and the mean processor utilization keeps in 48.5%~67%. When PMEH reaches 80%, effective uniprocessor utilization is linearly proportional to the number of processors. From the curves in Figure 10.1 and 10.2, we estimate that the system can support 128 processors and keep mean processor utilization in 60%~65%. As we mentioned before, to make PMEH reach 80% is not impossible.

Because the shared data is noncacheable, increasing shared data reference frequency will make bus traffic heavy. Figure 11 shows the data collected when PMEH is 0.4 and SHD varies from 0 to 20%. In Figure 12, the PMEH is changed to

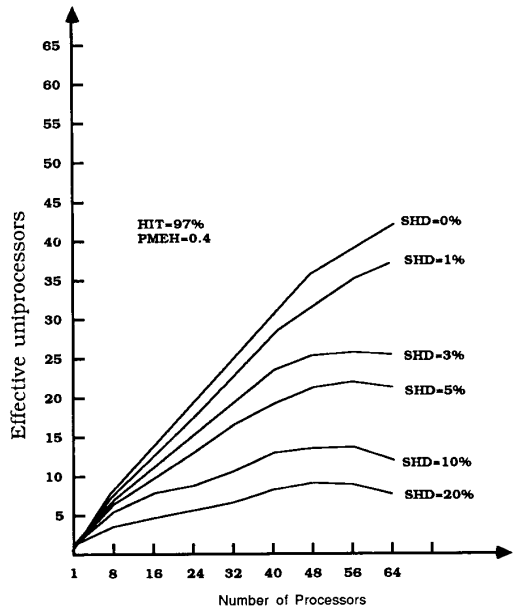


Fig. 11.1 Effective uniprocessors

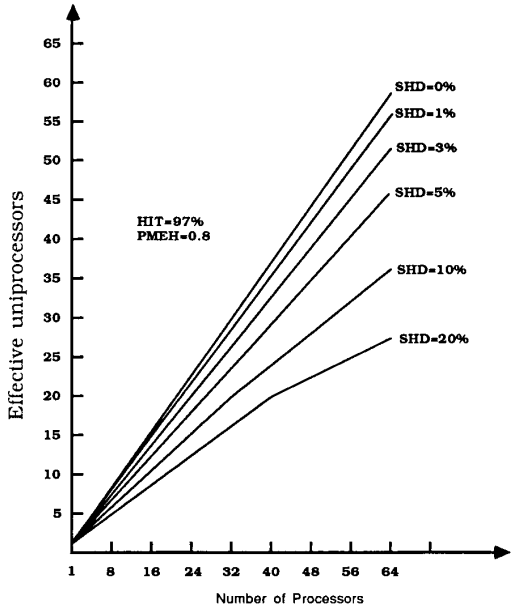


Fig. 12.1 Effective uniprocessors

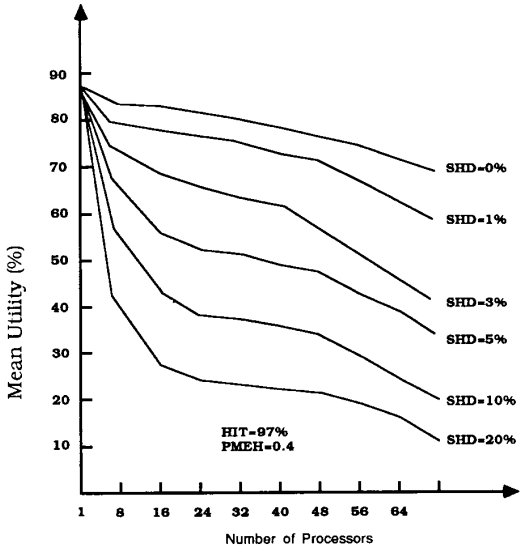


Fig. 11.2 Mean Utility

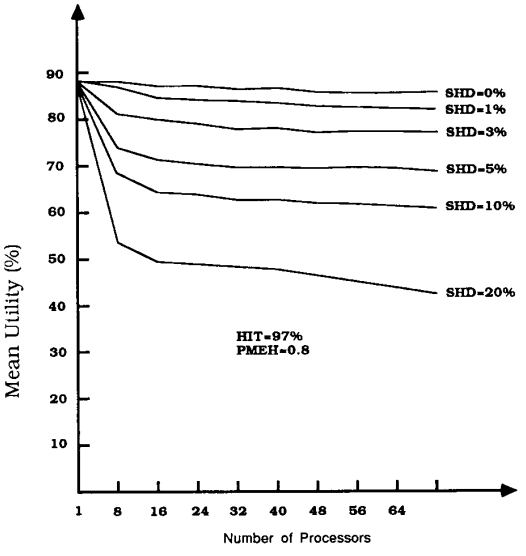


Fig. 12.2 Mean Utility

0.8. Comparing Figure 11.1 and 11.2, we see that the system can support 32 processors reasonably when SHD is 5%. In fact, the system saturates at 48~52 processors in this case. If SHD is larger than 5%, the system can support 10~20 processors. In this case, we can say that the system is not fully utilized. If SHD is less than 3%, the system can support 64 processors easily. From Figure 12.1 and 12.2 we know that the system can support 64 processors and keep processor utilization between 60% and 80%. Thus we know that, if PMEH reaches 80%, the system can get its best performance. For the system to support more than 64 processors, the system parameters must be chosen carefully. Under the condition that HIT is 93% and SHD is 5%, PMEH must exceed 70% to satisfy the requirement. If HIT increases to 97%, PMEH must be larger than 60% in order to reach the goal. The main reason is that the private data is the dominant

part of a process. In such case, the most part of memory references can be satisfied by its own memory module. If SHD exceeds 5%, PMEH needs to be more than 80% to support 64 processors.

The above simulation results tell us that if SHD exceeds 10%,the system must be improved to satisfy our request, i.e. 64 processors. Fortunately, the SHD falls between 0.1% to 5% in most case [18]. IF not, there are two ways to solve the bottleneck problem. One is to decrease memory access time, and the other is to reduce the frequency of bus request. Improving memory speed can be by hardware technique. What we emphasize is only how to decrease the frequency of bus requests. In our model, the key factor that affects the frequency of bus requests is the noncacheability of shared data. To solve the problem, the cache coherence protocol can be improved to cache shared, writable data during periods when they are modified by only one processor [10]. In such case, the ratio of shared data cache can be considerably reduced. This improvement in cache coherence protocol can make our system more powerful. So there is no problem for our system to support 64 processors.

5. Conclusion

In this paper, we estimate the performance of MARS with shared-bus and with hierarchical multiple-bus. We use hierarchical technique to simplify the system model. In these models, the effects of memory accesses of shared data are taken into consideration. The interaction between processors is managed in these models, too.

In MARS with share-bus, our Phoenix protocol is designed to manage the cache coherence[20]. We analyzed four variant types share-bus system. Comparing the performance of type 1 and type 2 systems, we observe that interleaving memory to each processor without write-buffer can improve system performance from 2.75% for two processors system to 145.1% for 10 processors system. Interleaving memory to each processor with write-buffer can improve system performance from 5.14% for 2 processors system to 154.2% for 10 processors system. The more processors there are, the more improvement in performance can be achieved by interleaved memory.

The write-buffer can improve system performance by 5%~36%. In most cases the improvement caused by write-buffer exceeds 10%. The improvement is roughly proportional to number of processors. If the number of processors is less than 4, the write-buffer size 2 is good enough. If the number of processors reaches 10, the write-buffer maximum queue length will reach 6, but the mean queue length is low. For this reason, write-buffer size 4 is suitable. Because PMEH is high, our MARS system with shared-bus can support 12~16 processors reasonably.

About hierarchical multiple-bus system, we use packet-switch to transfer data. The shared data is noncacheable. This protocol makes SHD and PMEH decisively influential on system performance. If PMEH is 80%, the system can almost get its best performance, it can support 64 processors and keep each processor utilization between 60% and 80%. In such case, it is not difficult for the system to support 128 processors. To make PMEH reaches 80% is not impossible, because most memory accesses are the private data references. However, if SHD exceeds 10%, it is difficult to support 64 processors. This bottleneck can be released by putting shared data into cache to reduce data cache miss and using more efficient cache coherence protocol to get data consistence.

Reference

[1]G. S. Jang, F. Lai, H. C. Lee, Y. C. Maa, T. M. Parng and J. Y. Tsai, "MARS-Multiprocessor Architecture Reconciling Symbolic with Numerical Processing, A CPU Ensemble with Zero-Delay Branch/Jump," Int'l Symposium on VLSI Tech., Systems, and Applications, May 1989, pp. 365-370.

[2]G. S. Jang, T. J. Horng, F. Lai and T. M. Parng, " Integer Processing Unit for MARS, Architecture and Implementation," Int'l Symposium on IC Design and Manufacture, Singapore, Sep. 1989, pp. 43~52.

[3]M. G. H. Katevenis, "Reduced Instruction Set Computer Architectures for VLSI," Ph.D. dissertation, Computer Science Division , University of California,Berkeley,Oct. 1983.

[4]G. Radin, "The 801 Minicomputer," Proc. SIGARCH/SIGPLAN Symposium on Architectural Support for Programming Languages and O.S., ACM, Palo Alto, 1982. pp. 39-47.

[5]C.E. Gimarc and V. M. Milutinovic, "A Survey of RISC Processors and Computers of the mid-1980s,"Comput.,Sep. 87.

[6]J. Y. Tsai, K. C. Chen and F. Lai, "RISC Architecture for Lisp with Powerful Environment Control and Fast List Access," Int'l Symposium on Computer Architecture and Digital Signal Processing, Hong Kong, Oct. 1989, pp. 165~171.

[7]H. C. Lee and F. Lai, "MARS-Multiprocessor Architecture Reconciling Symbolic with Numerical Processing," submitted to Journal of Information Science and Engineering, May. 1989.

[8]G.F. Pfister et al., "The IBM Research Parallel Processor Prototype (RP3): Introduction and Architecture," Proc. 1985 Int'l Conf. Parallel Processing, pp. 764-771.

[9]W.C. BRANTLEY,K.P. MCAULIFFE and J.Weiss, "RP3 Processor-Memory Element," Proc. 1985 Int'l Conf. Parallel Processing, pp. 782-788.

[10]P. Stenstrom, "Reducing Contention in Shared-Memory Multiprocessors," Computer, Nov. 1988, pp. 26-37.

[11]K. Hwang, F. A. Briggs, "Computer Architecture and Parallel Processing," McGraw-Hill, Highstown, New Jersey, 1984.

[12]"PAWS 3.0 - Performance Analysis Workbench System User's Manual,"Information Research Associates,Austin,Texas, 1987.

[13]C. H. Sauer, K. M. Chandy, "Computer Systems Performance Modeling," Prentice-Hall, Inc., Englewood Cliffs, 1981.

[14]E. D. Lazowska, J. Zahorjan, G. S. Graham and K. C. Sevcik, "Quantative System Performance-computer system analysis using queuing network models,"Prentice-Hall,Inc. 1984.

[15]A. Thomasian and K. Gargeya, "Speeding up Computer System Simulations using Hierachrical Modeling,"ACM SIGMETRIC,'86.

[16]J. K. F. Lee and A. J. Smith, "Branch Prediction Strategies and Branch Target Buffer Design," Comput.,Jan.1984,pp. 6~22.

[17] T. R. Gross, J. L. Hennesy, S. A. Przybylski and C. Rowen, "Measurement and Evaluation of MIPS Architecture and Processor"ACM Trans.Comput.,V.6,No.3,Aug.'88,pp. 229~257.

[18]J. Archibald, J. L. Bear, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model," ACM Trans. Computer, Vol. 4, No. 4, Nov. 1986, pp. 273~298.

[19]C. L. Mitchell, "Processor Architecture and Cache Performance," Technical Report, No. 86-296, Computer System Laboratory, Departments of Electrical Engineering and Computer Science, Standford University.

[20]W. H. Liao, "Cache-Based Memory System," Master Thesis, Computer Science Div., Dept. of Electrical Engineering, National Taiwan University, Jun. 1989.

[21]A. Agarwal and A. Gupta, "Memory-Reference Characteristics of Multiprocessor Applications under MACH," Proc. 1987 ACM SIGMETRICs Conf., 1988, pp. 215-225.