A Quantitative Evaluation of Cache Types for High-Performance Computer Systems

C. Eric Wu, Member IEEE, Yarsun Hsu, Senior Member, IEEE, and Yew-Huey Liu

Abstract—Parallel accesses to the table lookaside buffer (TLB) and cache array are crucial for high-performance computer systems, and the choice of cache types is one of the most important factors affecting cache performance. In this paper we classify caches according to both index and tag. Since both index and tag could be either virtual (V) or real (R), our classification results in four combinations or cache types. The real address caches with virtual tags for high-performance computer systems in our study are prediction-based, since index bins are generated from a small array and predictions could be false. As a result, we also discuss and evaluate real address MRU caches with real tags, and propose virtually indexed MRU caches with real tags. Each of the four cache types and MRU caches are discussed and evaluated using trace-driven simulation. Our results show that a virtually indexed MRU cache with real tags is a good choice for high-performance computer systems.

Index Terms—Cache type, cache index, cache tag, MRU scheme, performance analysis, miss ratio, synonym, pseudonym, write back.

I. INTRODUCTION

A cache is a relatively small high-speed memory that is used to hold the contents of the most recently used blocks of main storage. It is not surprising that caches have been extensively studied since IBM introduced the System 360 Model 85 [12] in 1968. In addition to a good survey [17], Smith [19], [21] listed many papers, books, and notes published in the past, and numerous papers appear each year in various forums. Each of the major design decisions has been the focus of many papers. For example, cache size is so important that it is a part of almost all cache studies, such as in [1], [7], [18], [27]. Block (or line) size was examined in detail in [20], and associativity was discussed in [8]. Other factors, such as replacement algorithms [16], [22], [23], prefetch schemes [17], write policies [24], and cache coherence strategies [2], have also been evaluated in many papers. However, with some notable exceptions most work to date has not focused on the choice of cache types.

One common way to classify caches, as is used in [17], is to designate a cache as either a virtual address cache or a real address cache according to the way in which it is indexed. In this paper, the type of a cache is determined not only by its index, but also by its tags. The index is used to select a set of lines in the cache, and each line is associated with a tag to identify the address of the line. Since both the index and the tag could be either virtual or real, there are four possible combinations or types, namely, R/R, V/R, V/V, and R/V. We believe that such a taxonomy is necessary to eliminate possible confusion, because a virtual address cache could have real tags (i.e., V/R-type) and a real address cache could have virtual tags (i.e., R/V-type).

R/R-type caches, indexed by real addresses and having real tags, are used in many conventional computer systems, including the VAX 11/780 [5], IBM 3033 [9], and MIPS R3000 [14]. Since the page offset in a virtual address is the same as that in the real address, many R/R-type caches use bits from the page offset to select a set of cache lines (also called a congruence class) while simultaneously performing virtual-to-real address translation. Conversely, a cache indexed by virtual addresses and having virtual tags is called a V/V-type cache, such as those in the Berkeley SPUR [6], the MU-5 [10], and the first level cache in MIPS R6000 [15].

Caches need not be completely virtual or real. The cache in IBM 3090 system [26], for example, is indexed by virtual addresses while real address tags are used in the cache directory to compare with translated real addresses from the TLB. Such a cache is a V/R-type cache. Since the cache is indexed by virtual addresses, its access is completely overlapped with the TLB access. Other existing commercial systems having virtually indexed caches with real tags include the HP Precision Architecture [11], MIPS R4000, and ELXSI 6400.

The fourth combination is R/V-type caches, indexed by real addresses and having virtual tags. An example of this cache type can be found in [25]. However, there is no commercial computer systems having a R/V-type cache so far. The following table shows examples of various cache types in modern computer systems.

<table>
<thead>
<tr>
<th>Cache type</th>
<th>Computer systems</th>
</tr>
</thead>
<tbody>
<tr>
<td>R/R</td>
<td>IBM 3033, VAX 11/780</td>
</tr>
<tr>
<td>V/V</td>
<td>SPUR, MIPS R6000</td>
</tr>
<tr>
<td>V/R</td>
<td>IBM 3090, HP Precision</td>
</tr>
<tr>
<td>R/V</td>
<td>None</td>
</tr>
</tbody>
</table>

The goal of this paper is to study the choice of cache types for high-performance computer systems and discuss an important modified form, the MRU (most recently used) scheme, for high-performance systems. We simulate various cache organizations using trace-driven simulation and analyze our results.
The IBM 370 address trace used in our experimental study was derived from an instruction trace of IMS (information management system) running on MVS (multiple virtual storage) operating system. It is a mixed operating system and application trace of a representative commercial workload. The trace file is around 51,134 Kbytes in size and contains references, including both virtual and real addresses, for about 1.3 million instructions. In our trace-driven simulation for V/V-type caches each virtual address is tagged with its segment table origin (STO). A segment table in the IBM 370 architecture stores pointers of page tables, which contain the mapping relationship between virtual and real pages. The STO of a segment table is the real address where the segment table starts, thus is used as the identifier of the virtual address space.

This paper is organized as follows. R/R-type caches are analyzed in Section II, and V/V-type caches are discussed in Section III. V/R-type and R/V-type caches are evaluated in Sections IV and V, respectively. In addition, we analyze R/R-type MRU caches in Section VI and propose a new cache structure, the V/R-type MRU cache, in Section VII. Quantitative results from trace-driven simulation are reported in each section, and the advantages and disadvantages of each cache type are also discussed. Summary and concluding remarks are given in Section VIII.

II. R/R-TYPE CACHES

The caches under consideration in our experimental study are caches from 16 to 2048 Kbytes in size and associativity from 1 to 8. We use combined I/D caches with a line sizes of 64 bytes and page size of 4 Kbytes in all configurations. The miss ratio decreases as the cache size increases for a given associativity, and also decreases as the associativity increases for a given cache size. When a write-back instead of a write-through policy is used, one may want to know the write back ratio, defined as the number of write backs over the number of writes. Using trace-driven simulation, Fig. 1 shows the miss ratios and write back ratios of various R/R-type caches.

In a conventional set-associative R/R-type cache the cache array is indexed by bits from the page offset, while the TLB is indexed by bits from the virtual page number. When a hit occurs in the cache directory, a cache line is chosen from the selected set. Since it is not known which cache line has been selected until the end of tag comparison in the cache directory, this approach is called a late-select scheme. Fig. 2 shows the simplified diagram of a 16-Kbyte, four-way (four cache lines in a set) late-select cache with 4-Kbyte page size and 64-byte line size.

Given a fixed page size and line size, the number of bits available for accessing the cache array is fixed and is usually small. As technology improves, so does the size of the cache array. This in turn forces the cache set-associativity (the number of cache lines per set) to increase if address translation is to be overlapped with cache array access. The IBM 3033 [9] is a typical example, having a 16-way 64-Kbyte cache with a line size of 64 bytes. However, increasing cache set-associativity above 16 will make the cache very expensive [17].

If cache access cannot begin until the address translation is completed, as in a 16-way cache larger than 64 Kbytes or a four-way cache larger than 16 Kbytes, it may adversely impact the cycle time or demand additional stages in the CPU's pipeline. One way to overcome the difficulty of overlapping address translation and cache access in a large R/R-type cache is to have part of a virtual page number the same as that of its real page number. Thus, the set selected from the cache array is indexed by part of the virtual address, making parallel accesses to the TLB and cache array feasible.

Another approach for accessing the TLB and cache in parallel is to have a one-to-one correspondence between virtual and real page numbers in certain bit positions. In the IBM RS/6000 [3], for example, the virtual page number in a 1-Mbyte virtual address segment has a one-to-one correspondence to the real page number in a 1-Mbyte real address segment. Since a 1-Mbyte space has 256 4-Kbyte pages, the one-to-
one correspondence exists in the eight least significant bits between virtual and real page numbers. As long as the one-to-one relationship does not change, this implementation allows parallel accesses to the TLB and a four-way cache up to 4 Mbytes.

In addition to these approaches, MRU schemes [4, 23] have been used to access TLB and a large R/R-type cache in parallel. We will discuss R/R-type MRU caches in Section VI.

III. V/V-TYPE CACHES

Difficulty may exist in a large R/R-type cache when parallel accesses to the TLB and the cache array are critical to system performance. This suggests that a V/V-type cache be a good choice, in that the virtual-to-real address translation is not needed for accessing the cache.

Smith [17] discussed considerations in building a V/V-type cache. For example, a context switch may invalidate all cache lines in a V/V-type cache, unless all cache lines are tagged with identifiers of their address spaces. This is because a context switch changes the process which the CPU is currently working on. Cache lines could be invalidated each time a context switch occurs, and replaced gracefully [27] one at a time when a new line is needed. Fig. 3 depicts the simplified diagram of a 64-Kbyte four-way V/V-type cache with a line size of 64 bytes.

In addition to context switch considerations, the most serious problem is that of synonyms, when two or more virtual addresses map to the same real address. Synonyms occur whenever two address spaces share code or data. In a V/V-type cache, the only way to detect synonyms is to do the virtual-to-real address translation and then see if any other virtual addresses in the cache map to the same real address. For example, the second level real cache in MIPS R6000 [15] detects and resolves synonym problems that would arise in its first level V/V-type cache.

When a synonym is detected, the synonym line is moved from one set to another, possibly through the next-level memory hierarchy by reloading the cache line. Therefore, synonyms are considered as cache misses and are included when calculating miss ratios of V/V-type caches. Fig. 4 shows the miss ratios and synonym frequencies of various V/V-type caches.

Since the same address trace is used, the differences in miss ratios between Fig. 1 and Fig. 4 are caused by synonyms. In a large V/V-type cache, synonym frequency could be as large as 3%, which could be larger than the miss ratio of the same-size R/R-type cache. For example, a 1-Mbyte four-way V/V-type cache has a synonym frequency of almost 2.1% for the trace, resulting in a miss ratio of about 4.2%. A 1-Mbyte four-way R/R-type cache, on the other hand, has a miss ratio as low as 2.1%. Note that synonym frequency increases as cache size or cache associativity increases. The more hits a V/V-type cache has, the more synonyms it would encounter. Thus, synonyms offset the benefit associated with high associativity. Since synonym frequency is included in miss ratios in Fig. 4, it is shown that the differences in miss ratio among various choices of associativity become smaller as cache size increases, especially between direct-mapped and two-way caches.

If multiple copies of a synonym line are allowed in a V/V-type cache and a mechanism exists to keep them consistent, the impact of synonyms on the miss ratio could be reduced. Fig. 5 shows the miss ratios of various V/V-type caches in which multiple copies of a synonym line are allowed for fast synonym access.
Miss ratios of V/V-type caches are reduced in Fig. 5 because subsequent requests of a synonym line are fulfilled by another copy of the line in the cache. As long as all copies of the synonym line are consistent, a synonym line could have as many copies as needed to satisfy subsequent requests. For direct-mapped V/V-type caches, allowing multiple copies for synonym lines decreases the miss ratio only slightly, because a synonym line duplication may result in the removal of a recently used cache line. For set-associative caches, on the other hand, the difference is significant. Thus, synonym line duplication helps to reduce miss ratios more significantly for set-associative V/V-type caches.

The difference in miss ratios between V/V-type and R/R-type caches is by no means small, even when multiple synonym copies are allowed in V/V-type caches. Furthermore, the increasing synonym frequency may be another important factor which hinders the use of large V/V-type caches. Unless the operating system totally prevents synonyms from happening, we expect that a large V/V-type cache is unlikely to be a good choice for cache types.

IV. V/R-TYPE CACHES

Although early cache designs and developments were mostly in the category of R/R-type or V/V-type caches, V/R-type caches have been used in commercial computer systems. Virtual addresses are used in a V/R-type cache to index the cache array, for both cache access and replacement. When two virtual addresses map to the same real address and the line has been brought into a V/R-type cache using one of the virtual addresses, the set selected by the current virtual address is called the primary set, while the other sets in which the synonym line might be found are called the synonym sets. Fig. 6 shows the simplified diagram of a 64-Kbyte four-way V/R-type cache with a line size of 64 bytes.

Unlike a V/V-type cache in which synonyms always need special attention, the mapping of two different virtual addresses to the same real address in a V/R-type cache does not necessarily create a problem. In fact it is a primary hit as long as the line is found in the primary set. In order to distinguish different cases in a V/R-type cache, we define alias as the case in which the two virtual addresses differ only in the bit positions which are not used to index the cache, and pseudonym as the case in which the two virtual addresses differ in the bit positions which are used to index the cache. In other words, it is an alias when the line is found in the primary set, and a pseudonym when the line is found in a synonym set. By these definitions, a V/V-type cache has to deal with both aliases and pseudonyms, while a V/R-type cache needs to handle only pseudonyms.

Clearly, a V/R-type cache does not suffer from context switching as a V/V-type cache does due to its real tags. In addition, pseudonym detection in a V/R-type cache can be done in a single cycle [26]. For example, bits 20 to 25 in Fig. 6 are used to drive out all four sets in which the current reference may reside. To speed up cache access, two real page numbers from the two-way TLB are compared with the 16 real tags simultaneously. If all the 16 cache lines are driven out at the same time, the penalty of synonyms in terms of cycle time could be eliminated. However, as the number of synonym sets increases for larger V/R-type caches, having multiple comparators and driving out all possible cache lines for fast pseudonym detection and resolution may become difficult.

In our experimental study for V/R-type caches we assume that a pseudonym results in moving the cache line from one set to another, thus invalidating the line in its original location. Fig. 7 shows the miss ratios and pseudonym frequencies of various V/R-type caches, where pseudonym frequency is defined as the number of pseudonyms over the number of references. We separate miss ratios and pseudonym frequencies this time for a closer comparison.

It is found that a large direct-mapped V/R-type cache has a lower miss ratio than a direct-mapped R/R-type cache of the same size. For instance, a 512-Kbyte one-way V/R-type cache has a miss ratio of 3.5%, whereas that of an R/R-type cache with the same configuration is 4.5%. Since the trace is generated from a large commercial application, the number of...
virtual pages is much larger than the number of pages available in the cache. As a result, virtual addresses are more uniformly distributed than real addresses across the least significant bits in the page number for cache access. On the other hand, the effect of invalidated cache lines due to pseudonyms in a large direct-mapped V/R-type cache is minimal.

This advantage of a decreased miss ratio in a large V/R-type cache for the trace diminishes as the associativity increases. In addition, an R/R-type or V/R-type cache could be accessed through a hash function, such as that in the HP precision architecture [11]. Rau [13] also proposed a pseudo-randomly interleaved memory approach that would help distribute real page numbers. Thus, the issue of unevenly distributed accesses in large R/R-type caches may be avoided.

Our results show that the pseudonym frequency of a V/R-type cache is much lower than the synonym frequency of a V/R-type cache with the same size. Comparing Fig. 7 and Fig. 4, the difference could be as large as two orders of magnitude. Thus, we conclude that a V/R-type cache is much better than a V/V-type cache for our trace. In the IBM system 370 architecture, the smallest shared unit is a page. In systems with larger shared units, such as a 1-Mbyte segment in SPUR [6], pseudonyms may be practically eliminated.

Given a fixed set-associativity, the pseudonym frequency of a V/R-type cache increases as its cache size increases, as is clearly illustrated in Fig. 7. However, it may either increase or decrease as associativity increases. When the cache size is small, the pseudonym frequency may decrease because a cache with higher associativity has fewer sets. The more sets a cache has, the more likely a synonym line is mapped into a different set. For instance, there is no synonym in a 16-Kbyte four-way cache with 4-Kbyte page size because the cache is completely indexed by page offset. A 16-Kbyte two-way cache has one bit from virtual page number for set selection. For a 16-Kbyte one-way (direct-mapped) cache, the number of bits from virtual page number increases to two, making pseudonyms more likely to happen. On the other hand, the number of elements in a set plays a more important role in larger caches. The more elements a set has, the more likely a cache has a synonym. For large direct-mapped V/R-type caches, pseudonym occurs when the synonym line is the only line in the original set. In an 8-way cache, however, it occurs as long as the line is one of the eight elements in the synonym set. Thus, the pseudonym frequency of a V/R-type cache depends on its set-associativity as well as the number of bits from virtual page number for set selection.

When a write-back instead of a write-through policy is used, one may want to compare write-back ratios between V/R-type and R/R-type caches. The write back ratios of various V/R-type caches are shown in Fig. 8. Our results show that the difference in write back ratios between V/R-type and R/R-type caches is relatively small.

V. R/V-TYPE CACHES

An R/V-type cache was first proposed in [25]. However, we are not aware of any R/V-type caches in commercially available computer systems. There are good reasons why R/V-type caches are not popular. Firstly, virtual tag caches suffer from synonym problems, as we have seen in Section III. Secondly, address translation is required to generate real addresses from virtual ones. Thus, a V/R-type cache is a more intuitive choice than an R/V-type cache.

In order to attain a high hit rate, a conventional TLB has 32 to 128 sets and a degree of associativity, typically two or four, implemented with comparators. The time for comparison plus the time to select the appropriate entry in the chosen set amount to a large fraction of the cycle time. Fig. 9 shows the simplified diagram of a 64-Kbyte four-way prediction-based R/V-type cache. The small array could substantially reduce both the time and the space by eliminating the comparison, thus in a way predicting the needed bits.

The small array in Fig. 9 is a history table. It predicts the needed real index bits based on the most recently used data. The number of entries in the small array does not have to be the same as the number of sets in its TLB. If it is, it is then called a TLBMRU, because its contents are identical to the contents of the TLB's most recently used (MRU) region.

The more entries the small array has, the more accurate its prediction becomes. Therefore, it is possible to increase prediction accuracy at the expense of array size and access time, or to improve access speed at the expense of prediction accuracy. The TLB slice in [25], for example, has a far smaller number of entries than there are sets in its TLB to predict additional real index bits.

Synonyms can also be a problem in set-associative R/V-type caches. When two virtual addresses are mapped to the same real address, they would be in the same set because the cache is indexed by real addresses. Thus, a mechanism for synonym detection and resolution is required.
If synonyms are not considered as cache misses, the miss ratios of various R/V-type caches would be exactly the same as those in Fig. 1. Fig. 10 shows the synonym frequency of various R/V-type caches.

If synonym detection and resolution is as costly in cycle time as a cache miss, allowing multiple copies of a synonym line should help decrease the miss ratio, as we have seen in Section III. Our results support our conjecture. Fig. 11(a) shows the miss ratios (including synonyms) of various R/V-type caches which do not allow multiple synonym copies, and Fig. 11(b) shows those when multiple synonym copies are allowed.

VI. R/R-TYPE MRU CACHES

In Section II, we discussed two approaches which make parallel accesses to the TLB and a large R/R-type cache feasible. Another hardware approach is the use of a history table, similar to the small array in Fig. 9 or the TLBMRU array in an R/R-type MRU cache [4].

In an MRU cache the cache array is logically partitioned into $S$ regions [23], where $S$ is the associativity of the cache. The main advantage of an MRU cache is the potential for a faster cache access by predicting that the reference is in the most recently used region of the cache. By doing so, the tag comparison can be removed from the critical path of cache access. When the requested line is in the cache but the MRU prediction fails (called a non-MRU hit), backup hardware is responsible for recovering from the mistake in one cycle. Fig. 12 depicts the simplified diagram of a 64-Kbyte four-way R/R-type MRU cache. The TLBMRU array in Fig. 12 is used to predict the additional bits needed to index the cache array, and its number of entries is the same as the number of sets in the TLB. Note that the hit/miss ratios of both the cache array and the TLB do not change with the addition of the MRU and TLBMRU arrays. We define MRU hit ratio as the number of references in the MRU region over the total number of references, and MRU ratio as the MRU hit ratio over the cache hit ratio. Fig. 13 shows the MRU ratios of various R/R-type MRU caches.

An MRU cache achieves fast cache access at the expense of non-MRU hit penalty. For example, a 16-Byte four-way MRU cache has a cache hit ratio of 89.7% and a MRU hit ratio 80.7% for the trace. About 9% of total references are found in the cache, but not in the MRU region. If the size of an R/R-type MRU cache is larger than its set-associativity times the page size, it has an extra penalty for non-TLBMRU hits in addition to penalty for non-MRU hits. The TLB miss ratio in a computer system is generally lower than the cache miss ratio, because page addresses rather than line addresses are stored in the TLB. Fig. 14 shows the miss ratios and the MRU ratios of TLB’s with various configurations.
There is a conflict in optimizing both the MRU hit ratio and the cache hit ratio. Given a fixed cache size, we need to decrease the set-associativity in order to increase the MRU hit ratio. On the other hand, decreasing the set-associativity would decrease the hit ratio of the cache. The best choice of associativity can be determined using results from trace-drive simulation. Let $C_a$ and $M_a$ be the cache hit ratio and the MRU hit ratio of the cache with the given size and associativity $a$.

Clearly, we have $M_1 = C_1$ for a cache with the given size, and $M_a$ is equal to $C_1$ of a cache whose size is $a^{-1}$ times of the given size.

Let $b$, $b > a$, be another choice of associativity. Apparently, $C_b > C_a$ and $M_b > M_a$. If we assume a one cycle cache, two cycles for non-MRU hits, and the next-level memory is $n$ cycles away, the choice of $b$ for associativity is better than $a$, if and only if

$$(C_b - C_a) \times (n - 2) > (M_a - M_b) \times (2 - 1).$$

The gain in cycle time by a cache with higher associativity $b$ is $(C_b - C_a) \times (n - 2)$, while the loss in cycle time due to a smaller MRU region is $(M_a - M_b) \times (2 - 1)$. When the gain is larger than the loss, a cache with the higher associativity is a better choice. For an $n$ between 10 and 15, we conclude that a four-way 128-Kbyte R/R-type MRU cache is better than a one-way, two-way, or eight-way R/R-type MRU cache. This verifies the choice of associativity in [4], although the miss ratios are different because we have used different traces. Note that a similar approach can be also applied to TLB, since a TLB is actually a small virtual address cache required to implement a paged virtual memory efficiently.

For a system with multilevel caches, the distance of the next-level memory drops to the range of five to ten cycles. Therefore, a first-level MRU cache with high associativity is unlikely to be a good choice for multilevel caches.

There are two major factors contributing to the difference in MRU ratio between V/R-type and R/R-type caches. One is the number of bits from virtual or real page number for indexing the cache, and the other is the pseudonym frequencies of direct-mapped V/R-type caches. For a direct-mapped V/R-type
cache, the pseudonym frequency is relatively low, indicating that invalidated lines due to pseudonyms are not frequently encountered. It is shown in Fig. 7 by the relatively flat solid-pluss line. On the other hand, virtual page numbers are more uniformly distributed than real page numbers in our trace, and there is no mechanism such as a hash function to distribute page or cache line allocation. The differences in miss ratio for direct-mapped V/R-type and R/R-type caches illustrate this point. The number of bits from virtual or real page number is limited in small- to mid-size caches, thus balances MRU ratio in V/R-type and R/R-type caches. For large MRU caches, the effect of cache indexing bits plays a more important role, thus favors V/R-type MRU caches in terms of MRU ratio.

In general, the decision on MRU cache type is not necessarily an easy one, since their relative performance depends on many factors, including pseudonym frequency, non-TLBMRU hit ratio, and the penalty to resolve them in each case. In addition to non-MRU hit penalty, a V/R-type MRU cache might have high pseudonym frequency for some traces, whereas an R/R-type MRU cache has an extra penalty for non-TLBMRU hits.

VIII. SUMMARY

Caches play an important role in computer performance. When the cache size increases due to technology improvement, cache type may be one of the most important factors on cache performance. In this paper, we evaluated various cache types and MRU schemes by trace-driven simulation. Generally speaking, a virtual tag cache does not have to wait for the completion of the address translation but may suffer from synonyms. The advantage of a virtual address cache is the inherent parallel accesses to the TLB cache array. Although it might also have the advantage of more evenly distributed accesses over a real address cache, the issue could be avoided by simple hash functions.

Parallel accesses to the TLB and the cache array of a large R/R-type cache are crucial for high-performance computer systems. MRU schemes help reduce the length of cache access path and make the implementation of a large, high-performance R/R-type cache feasible. A V/R-type MRU cache eliminates the TLBMRU array in R/R-type MRU caches, thus reducing hardware overhead, and eliminates the extra penalty for non-TLBMRU hits.

REFERENCES

Yarsun Hsu (M'79–SM’90) received the B.S. and M.S. degrees in electronics engineering from National Chiao Tung University, Taiwan, in 1971 and 1973, respectively, and the Ph.D. degree from Rensselaer Polytechnic Institute, Troy, NY, in 1979. From 1973 to 1975, he was an ROTC officer teaching electronics in Taiwan Navy, and from 1975 to 1976, he was an instructor in the Department of Electronics Engineering, National Chiao Tung University, Taiwan. From 1976 to 1979, he was a research assistant in the Microelectronics Laboratory, Rensselaer Polytechnic Institute. He was employed by General Electric Company to work on the design and development of semiconductor power electronics from 1979 to 1982. He joined IBM T. J. Watson Research Center, Yorktown Heights, NY, in 1982 as a research staff member. From 1982 to 1984, he worked on the VLSI technology for high-density dynamic RAM’s. Since 1984, he has been involved in the design and analysis of parallel systems, initially working on a multistage interconnection network (MIN)-based shared-memory scalable parallel system. He has successfully architected and designed a low-latency high-bandwidth CMOS switch that can be used to interconnect hundreds to thousands of 32-b microprocessors. He received an Outstanding Technical Achievement Award for his contribution to this project. Currently, he is responsible for the modeling and performance evaluation of a massively parallel message passing system as well as a parallel file system to enable parallel processors to access I/O data efficiently. His current interests include multiprocessor computer architecture, parallel and distributed systems, interconnection network, I/O subsystem, parallel file system, and VLSI design.

Yew-Huey Liu received the B.S. degree in agriculture chemistry from National Taiwan University in 1981 and the M.S. degree in computer science from Michigan State University in 1986. She is currently a Ph.D. candidate with the Department of Computer Science, Michigan State University. Previously, she was an assistant system manager at the ANR Computer Center, College of Agriculture, Michigan State University, from 1985 to 1988. She is a staff programmer at IBM T. J. Watson Research Center. Since joining IBM in 1988, she has been involved in the implementation of UNIX device drivers, the simulation of a guest operating system on Parallel AIX (PAIX) operating system, the design and implementation of checkpointing and rollback recovery techniques on PAIX and AIX/ESA, and cache performance analysis. Her current interests include computer architecture and operating systems, especially in the field of checkpointing and restart for computer programs.