The Performance of Cache-Coherent Ring-based Multiprocessors

Luiz Andre Barroso and Michel Dubois

CENG Technical Report 92-19

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, California 90089-2562
(213)740-4475

The Performance of Cache-Coherent Ring-based Multiprocessors

Luiz André Barroso and Michel Dubois

Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-2562
(213) 740-9130

barroso@paris.usc.edu
dubois@paris.usc.edu

November 1, 1992

Submitted to the 20th Annual International Symposium on Computer Architecture

Submitted to ISCA'93

The Performance of Cache-Coherent Ring-based Multiprocessors

Luiz André Barroso and Michel Dubois
Department of Electrical Engineering-Systems
University of Southern California
Los Angeles, CA 90089-2562
(213) 740-9130

barroso@paris.usc.edu
dubois@paris.usc.edu

November 1, 1992

Abstract

Advances in circuit and integration technology are continuously boosting the speed of microprocessors. One of the main challenges presented by such developments is the effective use of powerful microprocessors in shared memory multiprocessor configurations. We believe that the interconnection problem is not solved even for small scale shared memory multiprocessors, since shared buses are unlikely to keep up with the memory bandwidth requirements of new microprocessors. In this paper we extensively evaluate the performance of the slotted ring interconnection as a replacement for buses in small to medium scale shared memory systems and for processor clusters in hierarchical massively parallel systems, using a hybrid methodology of analytical models and trace-driven simulations. Snooping and directory-based coherence protocols for the ring are compared in the context of multitasking.

Keywords: shared memory multiprocessors, unidirectional ring, cache coherence protocols, performance evaluation.

Submitted to ISCA'93

The Performance of Cache-Coherent Ring-based Multiprocessors

Abstract: *Advances in circuit and integration technology are continuously boosting the speed of microprocessors. One of the main challenges presented by such developments is the effective use of powerful microprocessors in shared memory multiprocessor configurations. We believe that the interconnection problem is not solved even for small scale shared memory multiprocessors, since shared buses are unlikely to keep up with the memory bandwidth requirements of new microprocessors. In this paper we extensively evaluate the performance of the slotted ring interconnection as a replacement for buses in small to medium scale shared memory systems and for processor clusters in hierarchical massively parallel systems, using a hybrid methodology of analytical models and trace-driven simulations. Snooping and directory-based coherence protocols for the ring are compared in the context of multitasking.*

1.0 Introduction and Motivations

In the last decade, parallel processing has emerged as the consensus approach to high-performance computing. Most, if not all, of today's high-performance machines are shared memory or distributed memory multiprocessors, ranging from a few tens to thousands of processors. Even though much of the research in interconnection networks today aims at connecting thousands of processing elements - the massively parallel processing (MPP) trend - the problem of building interconnections for smaller scale multiprocessors is still not solved. As new and faster processors are made available each year, it becomes clear that shared buses, the most popular technology for current commercial systems, cannot cope with state-of-the-art RISC microprocessors, such as Digital's 21064 Alpha [10] with a peak performance of 400 MIPS.

Bus bandwidth is not likely to increase at the same pace as processor and circuit technology improves. The major reasons for the poor technological scalability of bus interconnections are some severe problems related to the bus topology itself. First of all, the bus is a mutually exclusive resource which means that only one processor can transmit at any given time. All processors must participate in an arbitration phase before accessing the bus. Pipelining is limited to overlap between arbitration and transmission. Another limiting factor is the bus clock cycle in which signals must propagate throughout the entire extension of the bus, and which depends on the length of the bus and its capacitive load. The length of a backplane bus is set by the number of processors and by the minimum distance (of about one inch) between two adjacent boards; this distance is dictated by packaging constraints, such as connector sizes and cooling requirements. Lastly, a transmitter on the bus drives several receivers at the same time, each receiver introducing stray capacitances contributing to the characteristic impedance of the bus and reducing the signal propagation speed [8]. A higher characteristic impedance also means higher currents to maintain the specified signal amplitude and thus more power dissipation and higher noise levels. Increasing the bus width to transfer more data per bus cycle is an attractive option. However, bus width is constrained by limited pin count and crosstalk interference, and little performance is gained from bus widths surpassing the size of the average data block.

In the past few years, point-to-point unidirectional connections have emerged as a very promising interconnection technology because they lack most of the problems associated with buses. Point-to-point connection links have only one transmitter and one receiver (one at each end) and can be very short (less than two inches) provided they connect adjacent boards on the backplane. Their characteristic impedance is very small (small capacitive load) and, if they are terminated properly, the transmission speed is much higher than on buses. An additional factor favoring point-to-point transmission is signal pipelining. A new transmission can be started on a point-to-point link before the previous one has reached the receiver. Therefore the clocking speed and the throughput of a point to point link is not limited by wire length. Overall, point-to-point connections are much more technologically scalable than bus connections, and we can expect their delivered bandwidth to benefit continuously from improvements in circuit technology. The potential of point-to-point communications in shared memory multiprocessors is demonstrated by the IEEE Scalable Coherent Interface (SCI) [17] set of standards, based on 500 MHz 16-bit wide point-to-point links in its first generation of circuits.

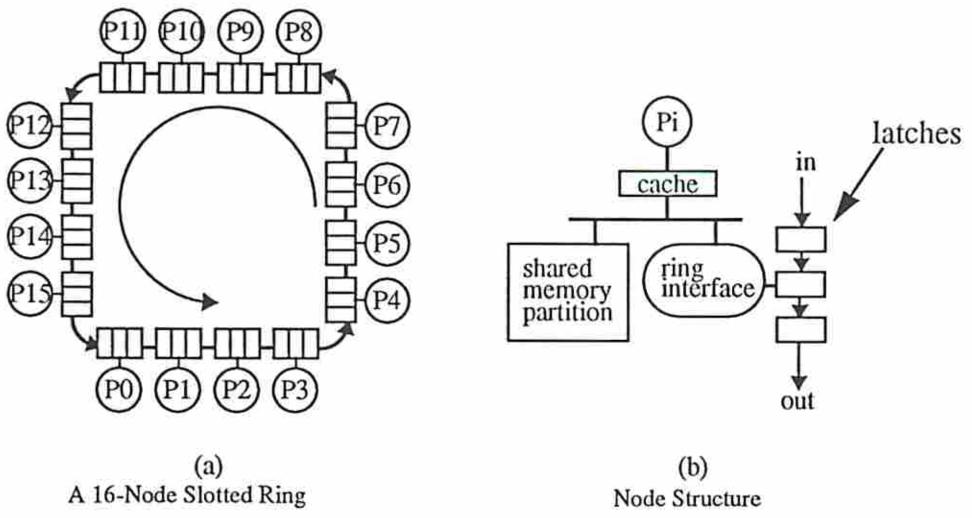
In this paper we evaluate the unidirectional slotted ring [3] as an alternative to buses for cache-based multiprocessor systems with up to 64 processors and in the context of multitasking. The unidirectional ring is the simplest form of point-to-point interconnection. Snooping and directory-based protocols are first described for the slotted ring, and then their performance is compared using a hybrid methodology including trace-driven simulations and analytical models. This study differentiates itself from past works on ring performance by considering the ring in the context of a shared memory multiprocessor, and evaluating its performance at the cache coherence level.

The slotted ring architecture is described in the next Section. Section 3 briefly explains the snooping cache coherence protocol as proposed in [3] as well as a directory-based protocol, both for the slotted ring architecture. Quantitative evaluations of the two protocols are shown in Section 4, after which the performance of the slotted ring is compared to that of a high-end split transaction shared bus. Related work is discussed in Section 5. Final remarks and conclusions are drawn in Section 6.

2.0 The Slotted Ring

The ring is the simplest form of point-to-point interconnection, which means minimum number of links per node and simpler interface hardware. In particular, the unidirectional ring requires the simplest routing mechanism possible: the only decision routing decision is whether to remove a message from the ring or to forward it to the next node. Consequently store-and-forward is avoided; communication delays are shorter and the raw bandwidth provided by point-to-point links is better utilized. Point-to-point connections are becoming so fast that it is possible that the board logic will eventually become a performance bottleneck, and therefore simple and fast routing mechanisms will be critical.

FIGURE 1. The Unidirectional Ring Interconnect



(a)
A 16-Node Slotted Ring

(b)
Node Structure

The general architecture of the unidirectional ring is shown in Figure 1, and consists of a set of processing elements containing a CPU, local cache memory, a fraction of the shared memory space, and a ring interface. The data path on the ring interface consists of one input link, a set of latches, and one output link. At each ring clock cycle the contents of a latch are copied to the following latch, within a ring interface and across the links, so that the interconnection behaves as a circular pipeline. The main function of the latches is to hold an incoming message for a few clock cycles in order to determine whether to forward it or not. The number of latches in each interface should be kept as small as possible so to reduce the latency of messages.

The ring access control mechanism which dictates when a node can send a message is complicated by the fact that messages can be larger than the width of the data path (latches and links), and may span multiple pipeline stages. Furthermore, messages can have different sizes. An interconnection for a cache-coherent system has to deal with at least two types of messages, which we call *probe messages* (or probes) and *block messages*. Probes are short messages carrying miss and invalidation requests, consisting typically of a block address field and other control/routing information. Block messages are made up of a header, which is similar to a probe, plus a cache block, and carry cache blocks for misses and write-backs.

To avoid the complexities associated with breaking messages into packets, the protocol must transmit messages in consecutive pipeline stages. Therefore, consecutive empty stages to fit a particular message must be found. There seem to be three main solutions: token passing rings, register insertion rings and slotted rings. In token passing rings a special bit pattern, called token, is passed from node to node allowing the node with possession of the token to transmit. The main disadvantage of token passing is that only one message may travel on the ring at a time, which is a waste of bandwidth if the ring can accommodate more than one message. In the register insertion approach, chosen for the SCI standard, a bypass FIFO between the input and output stages of the ring interface buffers incoming messages while the local processor is transmitting. When the transmission is completed, the contents of the FIFO are forwarded to the output link and the local processor is not allowed to transmit until the FIFO is emptied.

In the slotted ring approach the ring bandwidth is divided into marked message slots with different sizes circulating continuously through the system. A processor ready to transmit a message waits until an empty slot with the same size as the message passes through. A single bit in the header of the slot identifies an empty slot. The slotted ring restricts the utilization of the ring bandwidth by different message types because the mix of message slots is pre-determined. This restriction has no impact on performance provided the mix of slots matches the expected mix of messages. In a cache-coherent environment there are only two types of messages (probes and blocks). The particular mix depends on the cache coherence protocol, and therefore we postpone this discussion until the next Section.

We chose the slotted ring in our evaluations mainly for its simplicity, but also because it can support both snooping and directory-based protocols. Snooping cannot be implemented on a register insertion ring. An implementation of snooping on a token ring is presented by Delp and others in [9]. In Section 5 we further discuss the trade-offs between different ring access control mechanisms.

3.0 Coherence Protocols for the Slotted Ring

3.1 A Snooping Protocol

It is generally believed that snooping protocols [15] are only suitable for bus-based systems, and therefore protocols based on directories are favored [1,6,7] for point-to-point connected systems, such as the slotted ring. Snooping relies heavily on broadcasting of memory operations, which comes for free in bus systems. In fact, snooping protocols can be implemented on the top of any interconnection topology, provided the cost of the frequent broadcasts is acceptable with respect to the cost of unicasts or multicasts. Snooping protocols require less state information at the memory modules and are less complex than most protocols.

The snooping cache coherence protocol for the slotted ring, introduced in [3], is a write-invalidate write-back protocol, logically similar to an ownership-based snooping protocol for a split transaction bus. Three cache states, *Invalid* (INV), *Read-Shared* (RS), and *Write-Exclusive* (WE), indicate whether the block is not present in the cache, present in read-only mode, or present in read-write mode respectively. The node to which the address of a cache block maps is called the *home* node for that block. The node that has a WE copy of a block is called the *dirty* node. A *dirty bit* is stored with the block frames in memory to indicate when a block is cached in WE state. The dirty bit is very important to the snooping implementation on the ring since it indicates to the home node whether it has the most recent (or valid) copy of a block, and whether it should respond to miss requests for that block. When the dirty bit is set, the node with the WE copy of the block is responsible for responding to coherence requests.

TABLE 1. Snooping Protocol

Transaction	Protocol Behavior
Read Miss	if <block is dirty> dirty node sends copy to requester & changes to RS requester receives block & forwards copy to home node home node resets dirty bit else home node sends copy to requester requester final state is RS

Write Miss	if <block is dirty>
	  dirty node sends copy to requester & changes to INV
	else
	  all caches with RS copy change to INV
	  home node sends copy to requester & sets dirty bit
	requester final state is WE

Invalidation	/* block is already cached in RS state at requester */ all nodes with RS copies change to INV home node sets dirty bit

Read miss, write miss, and invalidation[1] requests are broadcasted through the ring in probe slots. Probes are inserted and removed by the requester and are "snooped" as they pass through each node in the system. The node with the valid block copy (the *valid* node) acknowledges a probe message. Coherence actions at the snooper are consistent with write-invalidate protocols and are shown in Table 1.

The most important feature of this snooping protocol is that no coherence transaction traverses the ring more than once. In this sense, the latency of coherence transactions is minimum. This is possible because probes are snooped on without being removed from the ring. The valid node does not send the acknowledgment in the same probe slot, but rather in an acknowledgment field in the following probe slot of the same type. This approach gives the node enough time to snoop. Furthermore, the latency of misses is independent of the relative positions of the requesting node and of the valid node. Therefore, the slotted ring with a snooping protocol behaves as a uniform memory access (UMA) interconnect, just like a shared bus.

Snooping implementations have harder real-time constraints than non-snooping implementations, since the snooper must react to all remote memory operations issued in the system. With today's point-to-point connection speeds of up to 500 MHz, the snooper hardware could not respond at the maximum rate of incoming probes. That prevents snooping from being implemented on a register insertion ring. The slotted ring access control mechanism proposed in [3] is one way to overcome this problem. In this scheme

1. The difference between a write miss and an invalidation is that an invalidation is issued when the node already has a RS copy of the block, and it is only requesting permission to write.

probe slots on the ring are separated by a minimum number of clock cycles by interleaving them with other types of slots, forming *frames*. A frame is composed of one probe slot for even-address blocks, one probe slot for odd-address blocks, and one block slot. Therefore, if the dual-directory on the ring interface is 2-way interleaved, two consecutive probes for the same part of a dual-directory are always spaced by at least a frame size, which is no less than 25 nsec for a 64-bit wide ring using a 16-byte cache block size and clocked at 200 MHz (5 nsec). The mix of 2 probe slots per block slot is the optimum for the snooping protocol, because the number of probes and block messages generated in actual simulations is approximately the same, and probes traverse the whole ring whereas block messages are removed by the destination and traverse only half the ring on the average.

3.2 A Directory-Based Protocol

The full map directory-based protocol outlined here has the same cache states as the snooping protocol. The home node of a block knows whether the block is dirty, and which nodes currently have valid copies. A directory entry is kept with each memory block with one presence bit for each node in the system. A set presence bit indicates that the node may have a copy of the block.

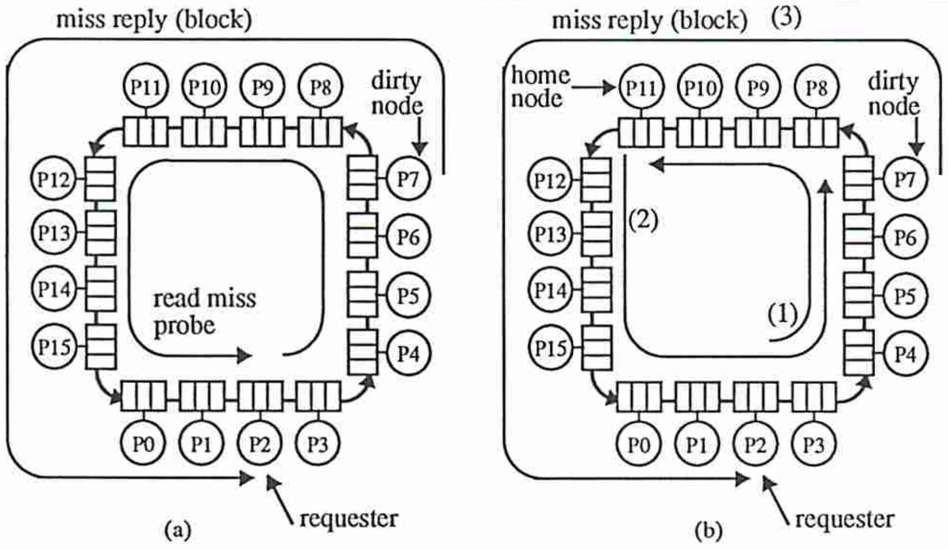
TABLE 2. Directory-Based Protocol

Transaction	Protocol Behavior
Read miss	if <block not dirty> home node records requester as having RS copy home node sends copy to the requester else home node forwards request to the dirty node dirty node changes its state to RS & sends copy to requester requester stores block as RS and forwards it to home node home node records the two nodes as having RS copies
Write miss	if <block uncached> home node records requester as having the WE copy home node sends copy to requester if <block cached not dirty> home node sends invalidation to nodes with RS copies & sends copy to requester; records it as having the WE copy if <block cached dirty> home node forwards request to dirty node dirty node replies to home node and sends copy to requester dirty node changes to INV; requester changes to WE home node records requester as having the WE copy
Invalidation	if <no other cached copies of the block> home node records requester as having the WE copy and acks req. else home node sends invalidation to all nodes with RS copies home node records the requesting node as having the WE copy

All coherence requests are first sent to the home node. The home node looks up the directory entry for the block and takes the appropriate coherence actions. Table 2 summarizes the coherence actions in the directory-based protocol.

From the protocol description in Table 2, read misses on clean blocks take only one trip around the ring, since it is a request/response transaction between the requester and the home node only. The latency in this case is the same as in the snooping protocol. Whenever the home node is not the valid node (i.e., there is a dirty node), a request is forwarded to the dirty node. If the dirty node is on the path between the requester and the home, one extra trip around the ring is needed, as seen in Figure 2.b. Moreover, on all write misses and invalidations for blocks that are cached RS elsewhere, the home node must send a multicast invalidation and wait for the reply before responding; this case also requires one extra ring traversal.

FIGURE 2. Read miss on a dirty block: (a) Snooping vs. (b) Directory-based



The choice of a full map directory protocol rather than other directory protocols such as limited directory protocols [1,2,16] or linked list protocols [14] was made based purely on performance considerations. We wanted to compare the snooping scheme with the most efficient directory protocol, in the context of the slotted ring architecture. Linked-list and limited directory protocols save memory especially in very large scale systems, but they are not likely to outperform full map directories in the context of the slotted ring.

Linked list protocols have been adopted by the SCI standard [17] and thus deserve special attention. In a linked list protocol, each block frame in a cache has one or more pointer fields linking all nodes with cached copies of a block in a *sharing list*. The home node keeps a pointer to the node at the head of the sharing list (the *head* node), which is responsible for maintaining the coherence of the block. For some requests, the unidirectional ring under a linked list protocol must be traversed multiple times. Since the head node is responsible for the coherence of the block, the home node must forward the request to the head node on each miss to a block currently cached. As in the full map protocol, the ring may be traversed once or twice, depending on the relative positions of the requester, the home and the head. Furthermore, invalidating the sharing list takes extra ring traversals when the order the nodes in the sharing list conflicts with the direction of the ring. In the worst case, it may take *n* traversals to invalidate a block shared by *n* nodes. Table 3 compares the distribution of misses and invalidations requiring 1, 2 and 3 or more ring traversals for three 16 processor benchmarks, in the linked list and full map protocols.

TABLE 3. Distribution of number of ring traversals (%): full directory vs. linked list

Ring Traversals	MP3D 16				WATER 16				CHOLESKY 16			
	Misses		Invalidates		Misses		Invalidates		Misses		Invalidate	
	full	l.list	full	l.list	full	l.list	full	l.list	full	l.list	full	l.list
1	70.5	67.0	12.6	7.1	72.4	53.5	12.6	7.2	84.5	66.5	17.1	5.2
2	29.5	32.0	87.4	87.7	27.6	45.9	87.4	88.6	15.5	31.5	82.9	75.5
3 or more	0.0	1.0	0.0	5.2	0.0	0.6	0.0	4.2	0.0	1.8	0.0	19.3

3.3 Discussion

The performance differences between the snooping and the directory protocol are very dependent on sharing patterns. If an application has very little read-write sharing of blocks, the vast majority of misses to shared data are misses on clean blocks, and most invalidations find the block uncached elsewhere. In this case the latencies for the full map and the snooping protocols are similar and the full map protocol needs less bandwidth than the snooping protocol, because the probes are not broadcast in the full map scheme. Therefore, the directory protocol could outperform snooping because of the lesser contention experienced by probe messages.

On the other hand, in applications with a fair amount of read-write block sharing, the coherence transactions in the directory protocol may experience significantly non-uniform and higher latencies. This non-uniformity of latencies cannot be easily avoided by intelligent placement of data or scheduling of processes. In Figure 2.b, if processors *P2* and *P7* share a block read-write, the configuration depicted always occurs for either *P2* or *P7*, no matter where the home node of the block is (except for *P2* or *P7*).

Unfortunately, the optimum mix of probe and block slots is not as predictable for the directory scheme as it is for the snooping scheme, because transactions do not always commit in a single ring traversal. If most misses are to clean blocks, a probe message is sent from the requester to the home node which removes the probe and replies with a block message to the requester. The probe and the block messages traverse half of the ring on the average, and the ratio between probe and block slots should then be set to 1:1 (it is reasonable to consider that invalidations and write-backs partially compensate each other in the total message count). However, if the number of misses and invalidations with more than one ring traversal is significant, the load of probe messages increases. In our experiments, the traffic (in terms of number of slots) generated by probe messages in the directory protocol was between 1 and 2 times the traffic generated by block messages. We chose a mix of 2 probe slots for each block slot for the directory protocol because the performance of the programs are more sensitive to probe slot utilizations than to block slot utilizations (in our simulations, the processor always blocks on a miss or an invalidation, but not on a write-back.)

4.0 Performance Evaluation

We have quantitatively compared the performance of the slotted ring under different classes of coherence protocols and of a high-performance split transaction bus under a snooping protocol. The evaluation methodology is a hybrid one, composed of trace-driven simulations and analytical models and similar to prior studies [7]. Very detailed trace-driven simulations of a limited number of configurations were first performed to gain better understanding of the interactions between the programs and the architectures. Then a set of simple analytic models were formulated to capture the essential performance characteristics. The parameters of the models are derived in the simulations, and the outcome of the models are validated against those same simulations. The advantage of this approach is that we have the type of confidence in the accuracy of the results that is expected from detailed simulations, but we also have the agility of analytical models to explore the design space more thoroughly and efficiently. Each run of our trace-driven simulations takes an average of 6-8 CPU hours to complete, on a Sun SparcStation2, generating only one point in the design space for each benchmark. The analytical models typically take less than a second of CPU to generate complete curves.

All the evaluations discussed in this section are performed by first simulating each benchmark for each value of network bandwidth and with 50 MIPS processors; the simulations generate parameters describing the average behavior of each system, including a count of each type of relevant coherence event. The parameters are then applied to the analytical models to generate all the curves. All model predictions fall within 15% of the simulated values for latencies, and within 5% for processor and network utilizations. The formulation of the model for the slotted ring with snooping is presented in the Appendix, as well as selected validation results. The models for the directory-based slotted ring and the split transaction bus follow a very similar derivation.

4.1 Benchmarks

The models and simulations are driven by two sets of benchmarks. The first set is a group of three programs from the Stanford SPLASH benchmark suite [29]: MP3D, WATER and CHOLESKY. These programs were traced using the CacheMire simulator [4] developed by Per Stenstrom's group at Lund University, Sweden, and traces were obtained for systems with 8, 16 and 32 processors. MP3D is a rarefied fluid flow simulation program used to study the forces applied to objects flying in the upper atmosphere at hypersonic speeds, and it is based on Monte Carlo methods. WATER evaluates the interactions in a systems of water molecules in liquid state and consists of solving a set of motion equations for molecules confined in a cubic box for a number of time steps. CHOLESKY performs a parallel Cholesky factorization of a sparse matrix, and it uses supernodal elimination techniques.

Traces for the second set of benchmarks were obtained from Anant Agarwal's group at MIT [7], and are 64-processor traces of three parallel FORTRAN programs: FFT, WEATHER and SIMPLE. FFT is a radix-2 fast Fourier transform program. SIMPLE solves equations for hydrodynamics behavior using finite difference methods. WEATHER also uses finite difference methods to model the atmosphere around the globe. The main characteristics of the traces derived from each benchmark are shown in Table 4. The miss rate values were derived with 128 Kbytes direct-mapped caches with a block size of 16 bytes. We assumed throughout that instruction references never miss, in order to reduce the simulation times. This assumption did not impact the results since the actual hit rate in the instruction cache is extremely high. The analysis also assumes that a processor blocks on all misses and invalidations. The time to access a local memory bank is 140 ns for all systems. The detailed simulations of the ring and bus systems were developed on top of the CSIM package [28] which is a library of C functions for process oriented simulation. A lot of effort was put in optimizing and verifying the correctness of the simulators, which model the architectures and cache coherence protocols in very fine detail.

TABLE 4. Basic Trace Characteristics (references in millions)

benchmark	proc.	data references	instruction references	private data references	shared data references	total data miss rate	shared data miss rate
MP3D	8	3.76	7.51	2.48 (22% write)	1.27 (33% write)	3.29%	9.44%
	16	3.94	8.23	2.50 (22% write)	1.43 (30% write)	4.54%	12.17%
	32	4.64	11.16	2.51 (22% write)	2.08 (21% write)	16.55%	35.74%
WATER	8	11.05	25.89	9.54 (18% write)	1.50 (7% write)	0.21%	1.38%
	16	11.36	27.15	9.55 (18% write)	1.81 (6% write)	0.32%	1.82%
	32	11.60	28.12	9.56 (18% write)	2.03 (6% write)	0.73%	3.82%
CHOLESKY	8	6.97	15.00	5.29 (21% write)	1.62 (14% write)	2.88%	10.61%
	16	8.91	21.26	6.27 (20% write)	2.55 (9% write)	6.12%	18.96%
	32	13.75	37.84	8.21 (18% write)	5.33 (5% write)	19.47%	46.71%
FFT	64	4.31	3.12	3.28 (27% write)	1.03 (50% write)	6.85%	26.12%
WEATHER	64	15.63	13.64	13.11 (16% write)	2.52 (19% write)	5.25%	30.78%
SIMPLE	64	14.02	11.59	9.94 (35% write)	4.07 (11% write)	15.97%	54.16%

4.2 Snooping vs. Directory Protocols for the Slotted Ring

The comparison between the snooping and the directory-based protocols for the slotted ring is shown in Figures 3 to 6. Processor utilization[2], average ring slot utilization and average miss latencies are displayed for systems with 8, 16 and 32 processors for the SPLASH benchmarks, and for systems with 64 processors for the remaining benchmarks. The clock rate of the ring is fixed at 500 MHz (2 nsec) and the processor cycle time varies from 1 to 40 nanoseconds. The simulations assume a scalar RISC-like single cycle instruction execution while accesses hit in the cache. Therefore, a processor cycle of 20 nsec. means 50 MIPS of processing power.

Using a 2 nsec. ring clock cycle the pure round-trip latency for an 8 processor ring is 50 nsec (excluding contention) and it increases linearly with the number of nodes. In the lower right corner of each Figure we show the distribution of the latencies of remote misses for the directory protocol[3]. Remote misses are misses to shared data requiring a

2. fraction of time that the processor is busy, instead of waiting for misses or invalidations.

ring access. In both protocols all read misses to unmodified blocks and write misses to uncached blocks are satisfied locally when the requester is also the home node. *Clean misses* take one ring traversal only, involving one probe message and one block message; *1-cycle misses* are misses to dirty blocks taking one ring traversal because of the fortunate relative position of the dirty node with respect to the requester and the home node; *2-cycle misses* are the remaining shared misses taking two ring traversals. It should be noted that the latency of 1-cycle misses is larger than that of clean misses since they require 3 hops to commit instead of two (requester to home, home to dirty, dirty to requester).

For MP3D, WATER and CHOLESKY, the fraction of clean misses increases steadily as the system size increases. This is a result of the random allocation of shared memory pages among the nodes, and of the fact that the traces for all three system sizes were derived using the same input data set. Consequently, for systems with 8 processors, a larger fraction of the shared memory space is allocated to each node than for systems with 16 processors and the probability that a miss to a shared data block will not be a remote miss is higher.

The snooping protocol outperforms the directory-based protocol for all system sizes for MP3D, because the fraction of 2-cycle misses is significant in all cases. The performance gap between the two schemes is not as wide for the 32 processor system, in which the fraction of 2-cycle misses is smaller. The ring utilization levels are always higher for snooping, because probes are always broadcast in the snooping protocol but not in the directory-based protocol. Nevertheless the difference in ring utilization levels is not enough to affect the latencies significantly because the ring is far from saturation. Notice how the effect of 1-cycle and 2-cycle misses causes the miss latencies to be higher in the system with 16 processors and the directory-based protocol than in the system with 32 processors and the snooping protocol.

For WATER, the extremely high hit ratio hides most of the differences between the snooping and directory-based protocols in terms of processor and ring utilization levels. The miss latency values however indicate the effect of the longer latency of 1-cycle and 2-cycle misses. CHOLESKY has a smaller fraction of 1 and 2-cycle misses for each system size than WATER and MP3D, and the gap between the latencies of misses for the two protocols is not as wide. For all benchmarks, and for systems with 32 processors or less the latency values do not respond significantly to the increase in ring utilization levels. In these cases, the processor utilization drops only because of the ring latency and not because of increases in contention for message slots.

For FFT, SIMPLE and WEATHER, which are 64 processor benchmarks, a slight increase can be seen in the latency values when the processing power increases. Despite that, the utilization levels for systems with 100 MIPS processors (10 ns cycle time) is still no larger than 50%. For such systems, the pure latency of requests, excluding contention for the network, is responsible for lowering the processor utilization figures, since the ring is still relatively underutilized. Again, the correlation between the mix of remote misses and the differences in performance between the two protocols is obvious. Among the three

3. As said before, the latency of all remote misses is constant for snooping, if not for contention delays.

FIGURE 3. MP3D: Snooping vs. Directory; 500 MHz ring

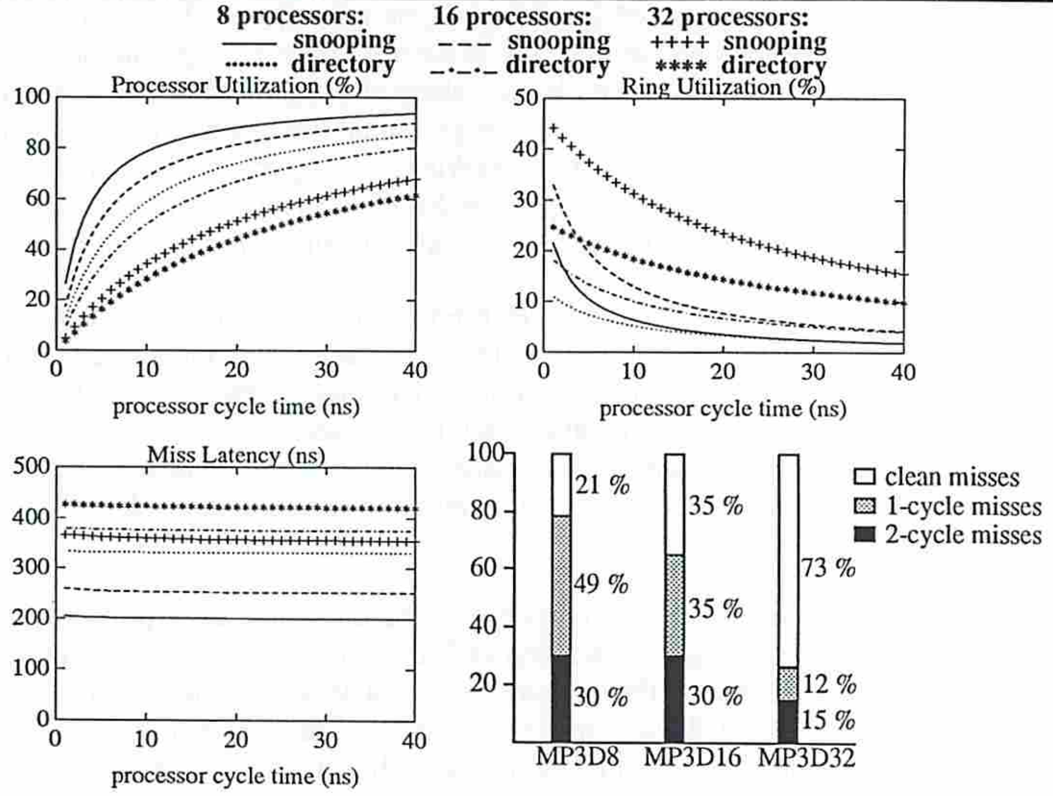


FIGURE 4. WATER: Snooping vs. Directory; 500 MHz ring

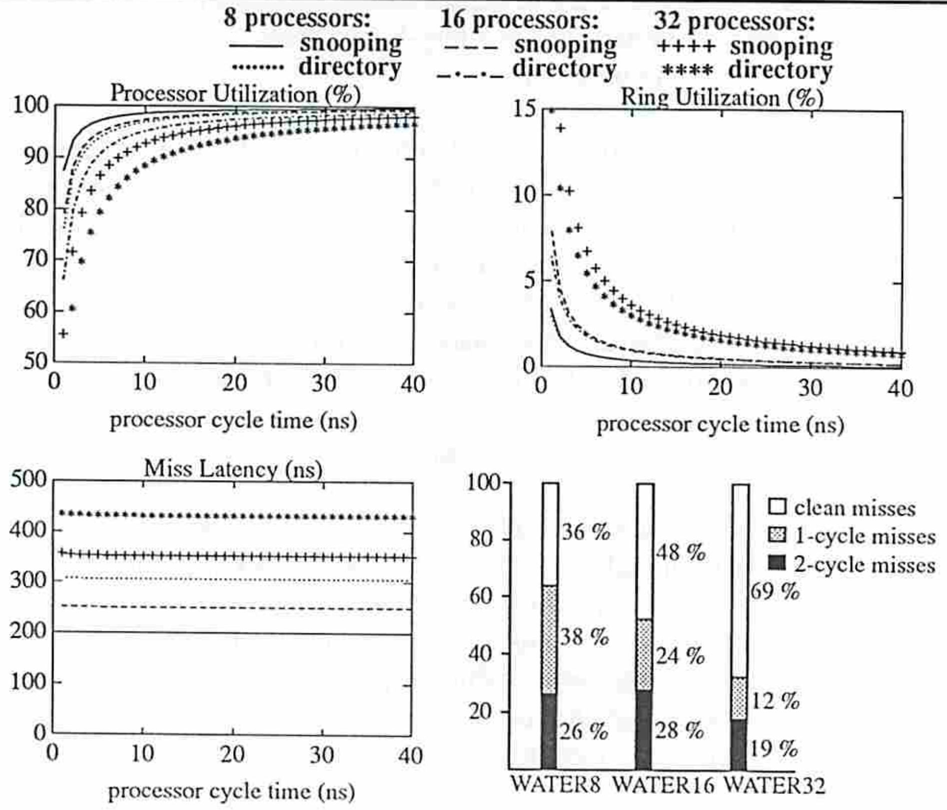


FIGURE 5. CHOLESKY: Snooping vs. Directory; 500 MHz ring

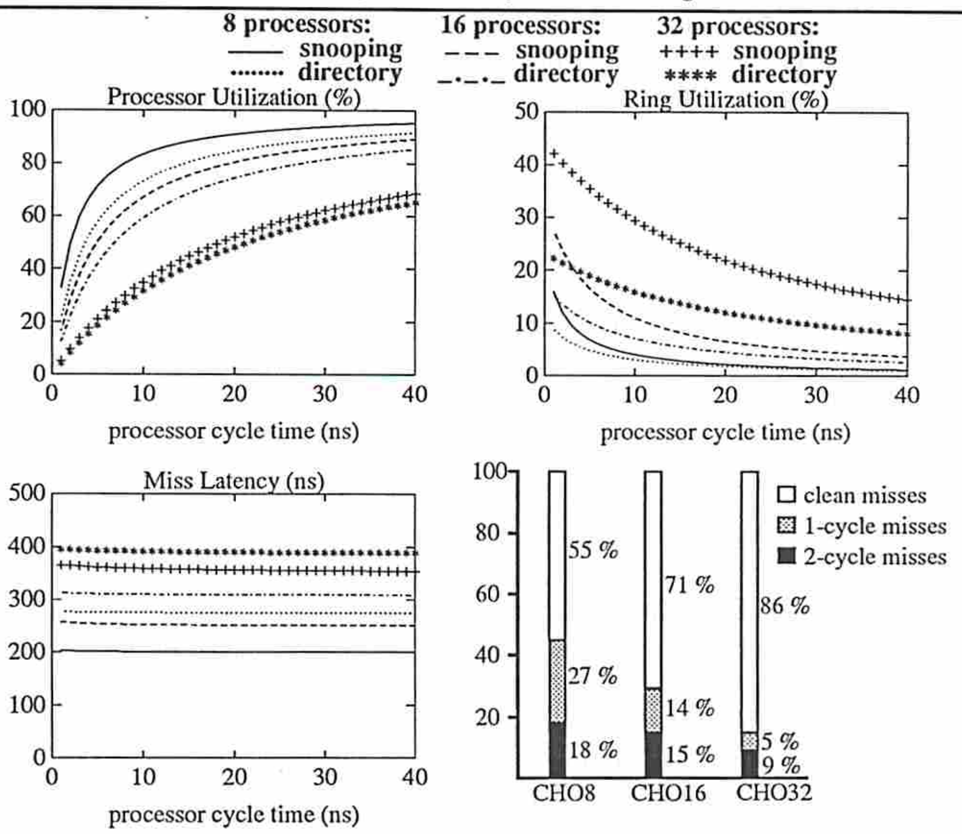
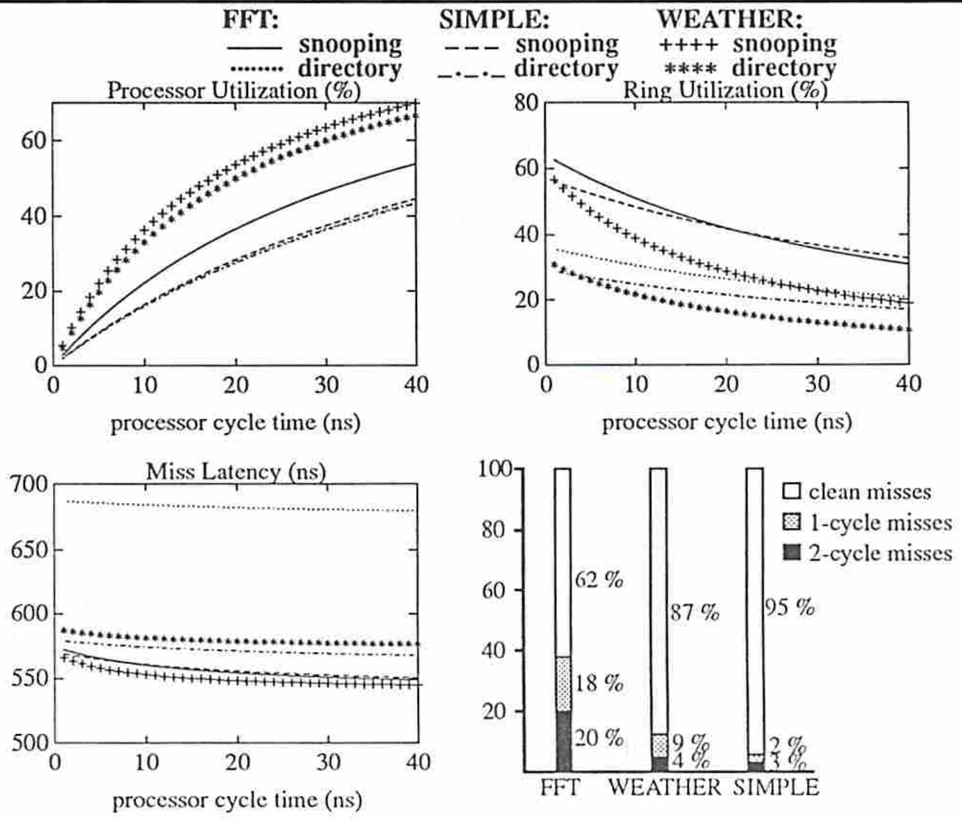


FIGURE 6. FFT, WEATHER and SIMPLE: Snooping vs. Directory; 500 MHz ring



benchmarks, FFT is the only one that shows a significant number of 2-cycle misses and 1-cycle misses. Consequently the snooping protocol shows a much better average miss latency than the directory-based protocol for this benchmark[4]. However, for the SIMPLE benchmark, in which there is a very small fraction of higher latency misses, the difference in average latency figures is negligible.

Although it seems possible that the directory-based protocol could outperform the snooping protocol in some special cases, that scenario never happened in our evaluations. The utilization levels of the ring were never high enough to start affecting the latencies significantly because the configurations with larger number of processors, which might saturate the ring, also have the larger round-trip delays. Since the processor always blocks on misses and invalidations, the round-trip delay itself becomes a lower bound on the minimum inter-arrival interval of messages for each node.

4.3 Slotted Ring vs. Split Transaction Bus

Having determined that the snooping protocol performs better than a full-map directory protocol on the slotted ring, we now compare the performance of the slotted ring with that of a split transaction shared bus. As before, we vary the processor cycle time from 1 to 40 nanoseconds. Both the ring and bus are 64-bit wide and use the same processing element configuration outlined in Section 2. The bus architecture is similar to a split transaction version of the Futurebus+ (IEEE 896.x standard), with a 3-state write-invalidate snoopy protocol. Figures 7-12 compares the performance of a 200 MHz and of a 500 MHz ring to a 50 MHz and a 25 MHz bus. The bus clock rates were chosen to represent aggressive values considering current technology and the range of system sizes.

The bus clock cycle remains constant for all system sizes, which is somewhat optimistic because of the electrical limitations of buses mentioned previously. As a result, the pure latency to satisfy a remote miss remains constant for the bus case (assuming no contention), while it increases linearly with the number of nodes for the ring case. Using a 16-byte cache block, the minimum number of bus cycles to satisfy a remote miss is 8, excluding arbitration delays and the time to fetch the block in the remote node's memory or cache. The pure latency of misses is about 550 nsec. for the 25 MHz bus and 350 nsec. for the 50 MHz bus. The limited bandwidth of the bus makes the actual miss latency values quite sensitive to variations in the processor speed, whereas the latency values for the ring remain nearly constant.

Note that the processor speed is only one of the factors affecting the load on the interconnect. The average miss ratio for shared data and the fraction of shared data references are also good indicators of the load on the interconnect, for given processor cycle and system size. Figures 7-12 show performance results for MP3D which (as can be seen from Table 4) has a relatively high miss ratio for shared data, and also has a significant fraction of shared data accesses. In the 8 processor MP3D the performance of the 50 MHz bus is comparable to the 200 MHz ring for slower processors (≤ 25 MIPS),

4. the processor utilization curves for SIMPLE (snooping and directory) and FFT-directory superimpose.

FIGURE 7. MP3D: 8 processors; slotted ring vs. split transaction bus

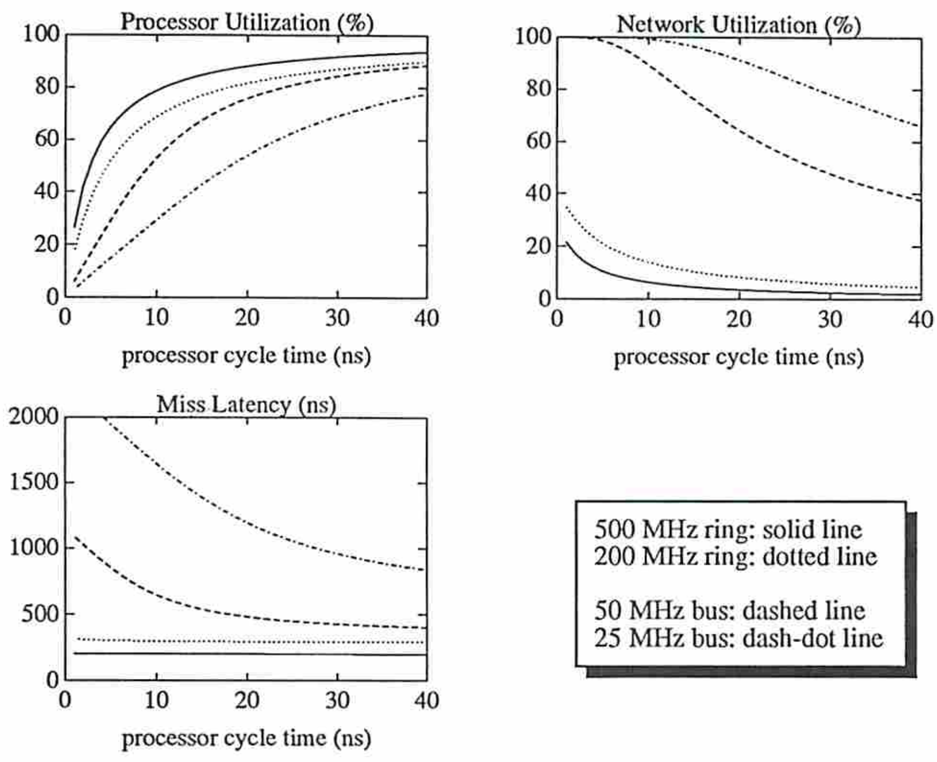


FIGURE 8. MP3D: 16 processors; slotted ring vs. split transaction bus

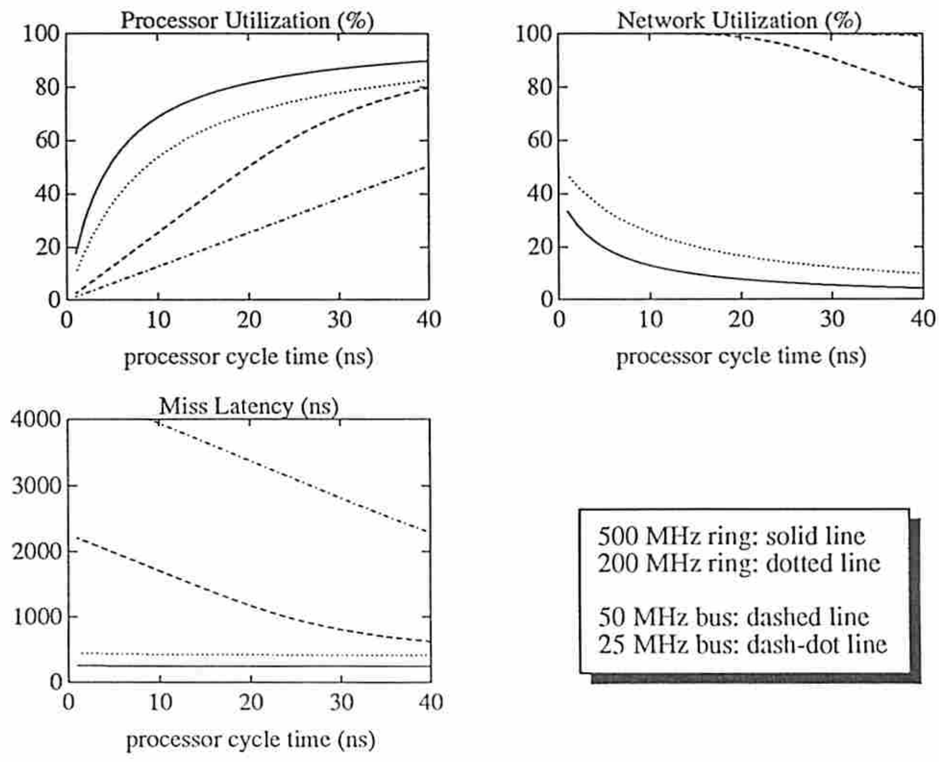


FIGURE 9. MP3D: 32 processors; slotted ring vs. split transaction bus

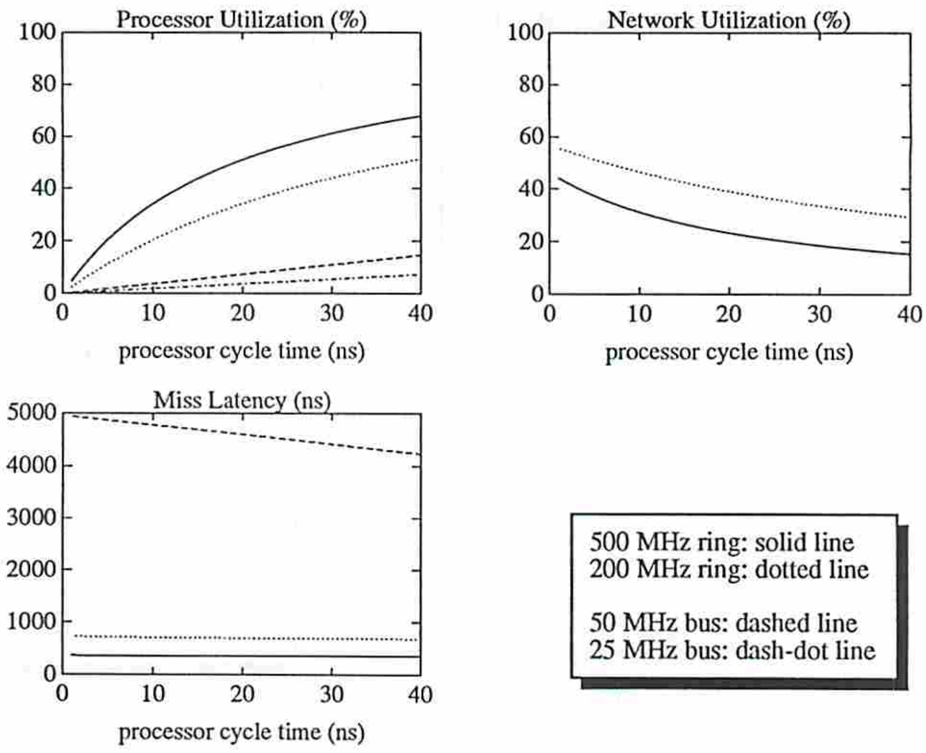


FIGURE 10. WATER: 8 processors; slotted ring vs. split transaction bus

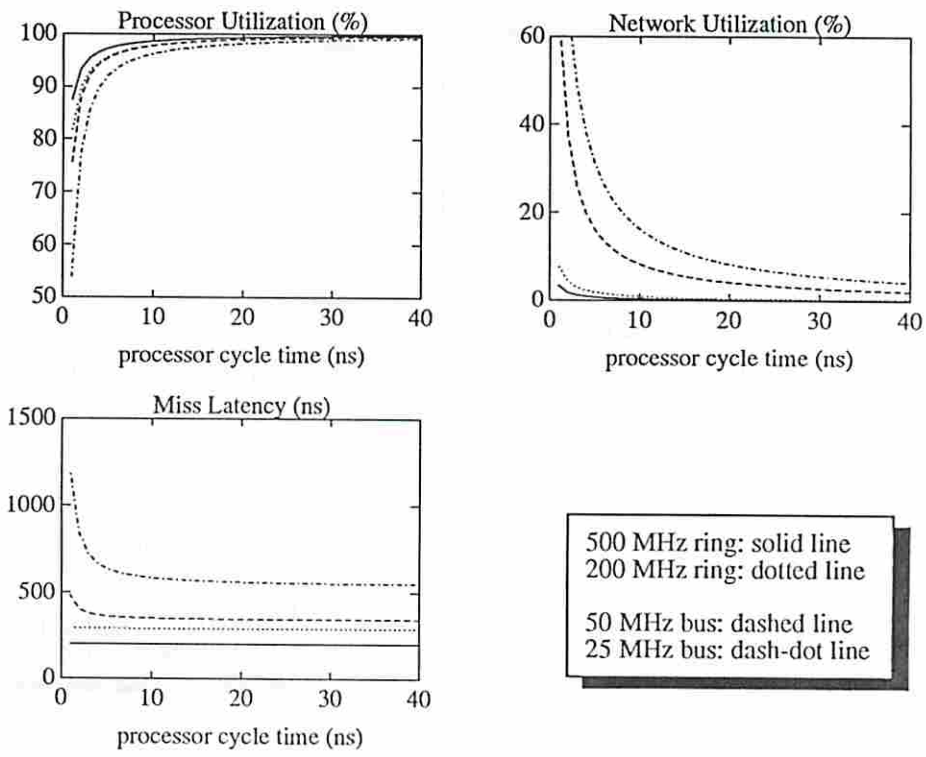


FIGURE 11. WATER: 16 processors; slotted ring vs. split transaction bus

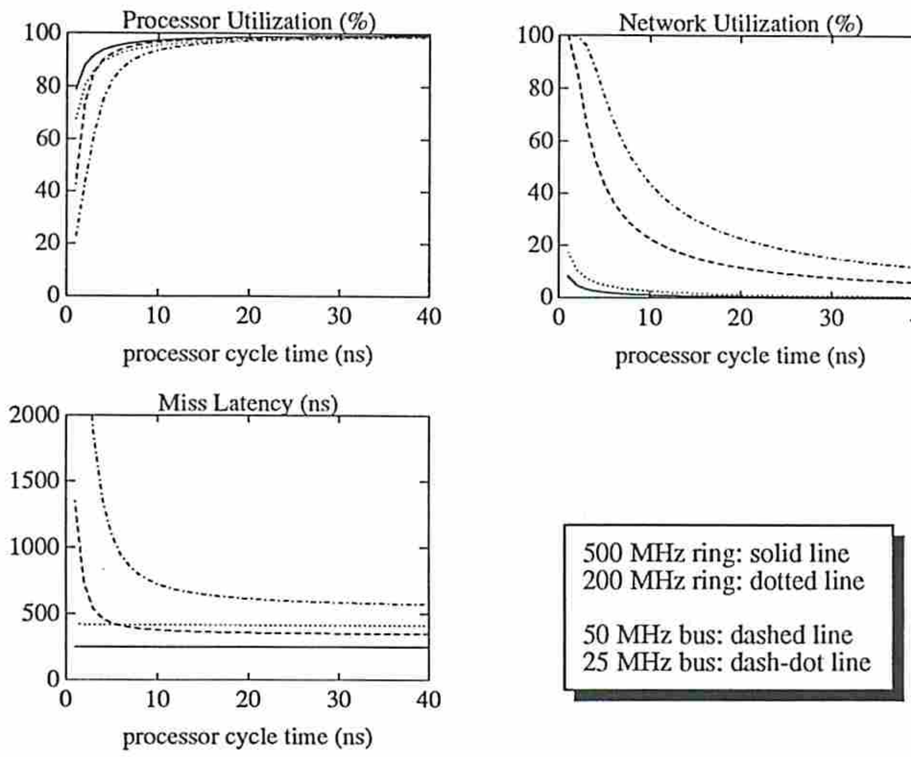
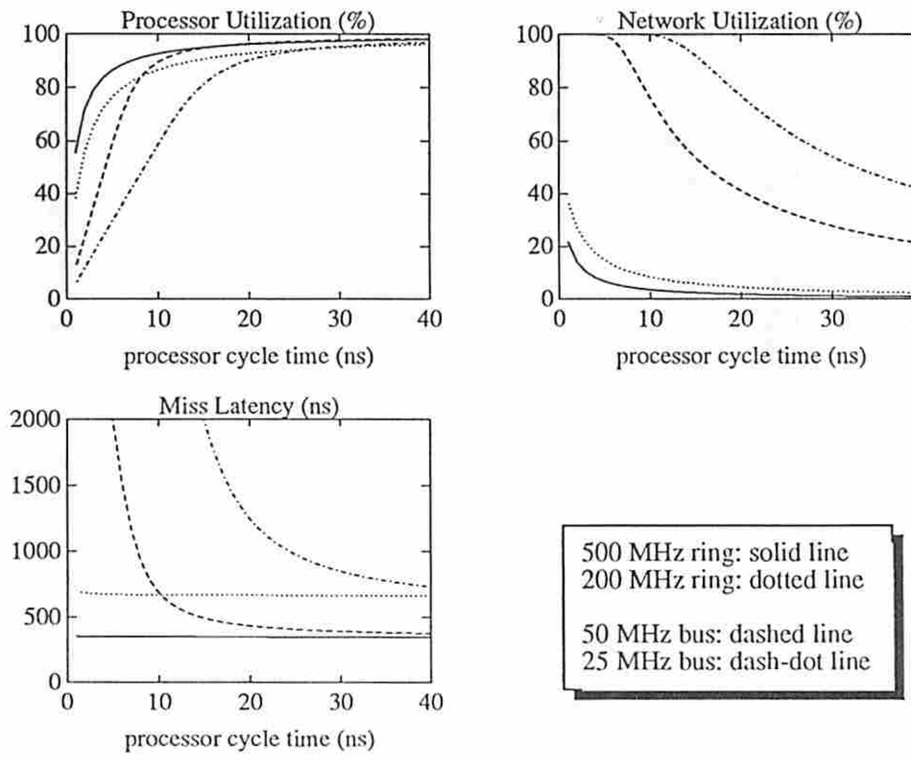


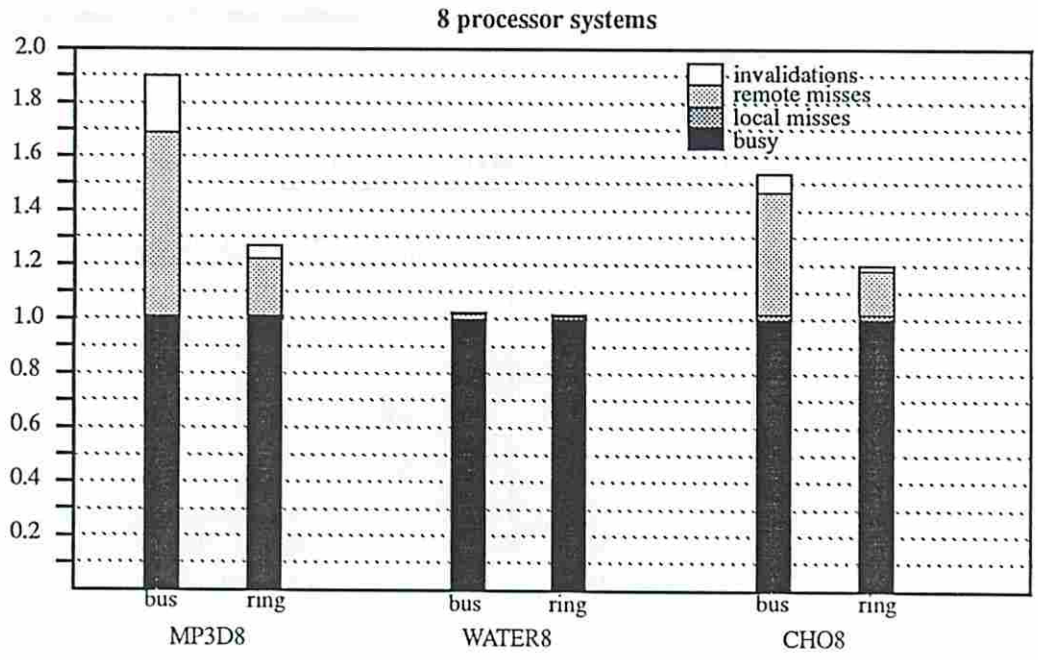
FIGURE 12. WATER: 32 processors; slotted ring vs. split transaction bus



but it falls behind for a processor cycle under 20 nsec (50 MIPS) because of bus conflicts. In the 16 processor MP3D the 50 MHz bus is already entering saturation for a 40 nsec processor cycle. In the ring configurations the processor utilization is still over 50% even for 100 MIPS processors. In the 32 processor MP3D both buses are completely saturated, whereas the ring latency is still low. The behavior of CHOLESKY is very similar to MP3D.

The evaluations using WATER show a very different behavior. In this case the miss rate values are extremely low, as is the fraction of references to shared data (see Table 4). The load on the interconnects is much lower than in MP3D. For 8 and 16 processors, the buses saturate for processor speeds higher than 100 MIPS. Even for 32 processors, the 50 MHz bus still shows a very good performance level for 100 MIPS processors. For the 16 and 32 processor configurations, the pure latency of the 50 MHz bus is lower than that of the 200 MHz ring and there is a region where the 50 MHz bus could outperform the 200 MHz slotted ring. However, in all cases, the slotted ring is all but unaffected by contention delays which is a result of a higher bandwidth. Eventually, as the buses reach saturation, the ring configurations have far better performance.

FIGURE 13. Contribution of misses and invalidations to the total program execution time



For 64 processors the bus systems are completely saturated in all cases. It is also very unlikely that buses for 64 processors can be clocked at 50 MHz, using current bus technology. Therefore, we will not compare buses and rings for 64 processors. In Figure 13 the contribution of latencies of misses and invalidations to the total program execution time, is shown for 100MIPS processors in an 8-processor configuration. The 500 MHz slotted ring is compared to a 50 MHz bus. The unit of the vertical axis is the time to execute the program if all accesses hit in the cache, i.e., the fraction of the execution time where the processors are busy. Remote accesses (misses and invalidations) can rapidly expand the total execution time of the program if the number of misses to shared data and invalidations is significant, as is the case in MP3D and CHOLESKY. In these cases the slotted ring shows significantly better performance than a split transaction bus even for small scale systems. A very low miss rate and a relatively small fraction of accesses to shared data are the reason why both bus and ring systems show very little miss/invalidation overhead for WATER. Such behavior is not likely to be representative of typical parallel programs due to the decreased spatial locality usually observed in shared data, and to the presence of false sharing.

TABLE 5. Bus clock cycle (ns.) to match the processor utilization values of a slotted ring

Benchmarks	100 MHz Ring		200 MHz Ring		500 MHz Ring	
	Processor speed (MIPS)		Processor speed (MIPS)		Processor speed (MIPS)	
	100	200	100	200	100	200
MP3D 8	18.2	14.7	11.8	9.5	6.3	5.3
WATER 8	35.8	33.8	19.9	18.6	8.7	8.4
CHOLESKY 8	19.0	15.2	12.1	9.9	6.1	5.3
MP3D 16	13.1	10.8	9.2	7.0	6.1	4.5
WATER 16	46.2	34.7	28.6	23.4	14.0	12.5
CHOLESKY 16	9.8	8.1	6.9	5.3	4.6	3.5
MP3D 32	6.2	5.9	3.6	3.3	2.2	1.8
WATER 32	28.4	18.4	22.6	14.6	15.9	10.7
CHOLESKY 32	5.9	5.6	3.5	3.2	2.1	1.8

In Table 5, we applied the same hybrid performance evaluation methodology to determine the bus clock cycle required of a bus system to attain the same processor utilization (i.e., the same program execution time) as a slotted ring system, for ring clock rates of 100, 200 and 500 MHz. and for processor speeds of 100 and 200 MIPS. Ring and bus widths are 64 bits. A 50 MHz bus (20 nsec) is competitive with a 100 MHz slotted ring for systems with up to eight 100 MIPS processors, whereas it takes a 200 MHz bus to be competitive with a 500 MHz slotted ring. For 32 processors, the bus systems to match the ring systems are probably impossible to implement, considering that busses with high clock rates are more difficult to build for larger number of processors. Moreover, for all the cases shown in Table 5, the utilization level of the buses matching ring performance are at least twice as high as that of the ring slots. With the exception of WATER8 and WATER16 the bus shows utilization levels above 50%, and is frequently saturated. In contrast, the slotted ring utilizations are always below 50%, and seldom above 30%.

Considering that today's high speed buses are clocked at 20 to 100 nsec [27] and that, baring any breakthrough in bus technology, these values are expected to improve rather slowly, it is safe to say that new bus systems with high performance microprocessors will probably be limited to up to 8 processors.

The evaluation results showed here also indicate that the slotted ring would benefit dramatically from latency tolerance techniques, such as lockup-free caches [26], weak ordering schemes [11,12,30,32] and multithreaded processors [25] because the large latencies observed for the slotted ring are not caused by heavy contention but by pure delays. In other words, there is latency to be tolerated despite the fact that the network is often underutilized. Since most latency tolerance techniques have the collateral effect of increasing the load on the interconnect because of the overlap of communication and computation, they can be self-defeating in an interconnect working close to saturation. This is would probably happen in a split transaction bus using today's high performance microprocessors. The latencies for the slotted ring however are still relatively stable and the network still far from saturated, in all the configurations that we have simulated, which indicate that the ring would be able to accommodate the increase in the load without significantly altering the expected latencies.

5.0 Related Work

Most studies of cache coherence performance for non-bus systems have assumed variations of directory schemes and have considered complex interconnections, more suited for large scale systems. Architectures such as the Stanford DASH [22] and the MIT Alewife [2] are examples of that. They differ in philosophy with the approach taken here since our focus is on interconnections for small scale systems, or for processor clusters in large-scale hierarchical multiprocessors.

The performance of unidirectional ring interconnections has been the subject of extensive analysis in the context of Local Area Networks (LANs) since the mid 70's. Hafner et al [18] proposed a register insertion ring LAN in 1974. Hooper and Needham describe the architecture of the Cambridge Fast Ring [20] that uses the slotted ring approach in a LAN environment. Bhuyan et al. [5] among several others have modeled the performance of ring LANs.

Still in the context of distributed systems, Delp et al proposed a token ring distributed shared memory system (Memnet [9]) with cache coherence maintained in hardware by means of a snooping-like coherence protocol. More recently Scott et al [27] have analyzed the performance of the SCI ring, which is an implementation of the register insertion access control strategy. They model the ring as a M/G/1 queue and derive the expected latency of messages with respect to network throughput, assuming an exponentially distributed arrival of messages. The authors point out that the register insertion approach suffers from fairness of access problems and may lead to starvation of a node. The mechanism proposed by SCI to avoid starvation is shown to impact the effective throughput of the ring. In the slotted ring it is very simple to avoid starvation of clusters by not allowing a node to reuse a message slot immediately after removing a message from that slot. Our simulations show that this has no significant impact on the system performance. Scott also makes a comparison between the register insertion ring and a shared bus, but without taking the cache coherence level into consideration.

It is not clear for us which scheme, slotted or register insertion, offers the best performance. Intuitively, under very light loads the register insertion ring may have faster access since a message does not wait for a proper slot to pass by. On the other hand, under medium to heavy loads, the simplicity of enforcing fairness on the slotted ring may yield a better performance than in the register insertion approach. The delay of transmitting a message in the register insertion ring can vary significantly depending on the activity of other nodes in the message path. We also believe that it is simpler to implement a slotted ring interface because it does not require a bypass buffer.

The Hector multiprocessor [31], under development at the University of Toronto, uses unidirectional token rings similar to our slotted ring. It is a shared memory multiprocessor with a three level interconnection hierarchy. Processing elements with a private cache and a part of the physical memory space are connected to a shared bus forming what is called *stations*. Stations are grouped in together in a *local ring*, which in turn are connected by a second level *global ring*. All rings are unidirectional. Even though the initial Hector architecture did not include hardware support for cache coherence (shared data were marked as uncachable), more recent work by Farkas at al. [13] recognizes the need for it and specifies a hierarchical cache protocol based upon the broadcasting of requests, as in the snooping protocol used here.

The only commercial implementation of a ring connected multiprocessor that we are aware of at this time is the Kendall Square Research KSR1 [21]. It is a shared memory cache-coherent multiprocessor based on a two-level hierarchy of unidirectional rings that uses a memory strategy called *Allcache*. In this strategy, the usual main memory is replaced by a very large cache as in the Data Diffusion Machine [19]. There is not much information currently available on the architecture or cache coherence protocol of the KSR1. They use proprietary microprocessors capable of delivering 20 MIPS and a 2-level coherence protocol. The latency of satisfying misses in a single unidirectional ring cluster is quite high in this implementation - typical miss latencies for requests satisfied within a 32-node ring:0 cluster is 6 to 7 microseconds in a 32 processor ring:0. The slotted ring architecture under snooping is capable of delivering latencies one order of magnitude shorter for a system with the same number of nodes.

6.0 Conclusion

In this paper we have evaluated a particular architecture for small to medium scale shared memory multiprocessor systems: the unidirectional slotted ring. A comparative analysis of two cache coherence strategies, snooping and directories, for the slotted ring was performed with a hybrid methodology incorporating detailed trace-driven simulations and analytical models. Contrary to the common wisdom, the snooping strategy shows a better performance than the directory-based strategy on the slotted ring. We justified the differences in performance between the two approaches by looking into the particular sharing patterns of the various benchmarks used in the evaluations. We also compare the performance of the slotted ring under snooping with that of a split transaction bus connected multiprocessor. The results show that even for systems with as few as 8 processors, the slotted ring shows a far better performance than a classic shared bus, for benchmarks with a significant fraction of remote misses and invalidations. Whereas the speed of bus interconnections are expected to lag further and further behind with respect to microprocessors, the ring interconnection is likely to keep up with future generations of microprocessors. A point is also made that slotted ring multiprocessors may benefit significantly from the use of latency tolerance techniques, which will be the subject of our future investigations.

7.0 References

[1] A. Agarwal, R. Simoni, J. Hennessy and M. Horowitz, "An Evaluation of Directory Schemes for Cache Coherence", Proceedings of the 15th International Symposium on Computer Architecture, pp-280-289, June 1988.

[2] A. Agarwal, D. Chaiken, K. Johnson, D. Kranz, J. Kubiatowicz, K. Kurihara, B-H Lim, G. Maa and D. Nussbaum, "The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor", in M. Dubois and S. Thakkar, editors, Scalable Shared Memory Multiprocessors", Kluwer Academic Publishers, 1991.

[3] L. Barroso and M. Dubois, "Cache Coherence on a Slotted Ring", Proceedings of the 1991 International Conference on Parallel Processing, Vol. I, pp. - , August 1991.

[4] M. Brorsson, F. Dahlgren, H. Nilsson and P. Stenström, "The CacheMire Test Bench - A Flexible and Efficient Approach for Simulation of Multiprocessors", Department of Computer Engineering, Lund University, Sweden, September 1992.

[5] L. Bhuyan, D. Ghosal, and Q. Yang, "Approximate Analysis of Single and Multiple Ring Networks", IEEE Transactions on Computers, Vo. 38, No. 7, pp. 1027-1040, July 1989.

[6] L. Censier, and P. Feautrier, "A New Solution to Coherence Problems in Multicache Systems", IEEE Transactions on Computers, C-27(12), pp. 1112-1118, December 1978.

[7] D. Chaiken, C. Fields, K. Kurihara and A. Agarwal, "Directory-Based Cache Coherence in Large Scale Multiprocessors", IEEE Computer, Vol. 23, No. 6, pp. 49-59, June 1990.

[8] D. Del Corso, H. Kirrman and J. Nicoud, "Microcomputer Buses and Links,", Academic Press, 1986.

[9] G. Delp, D. Farber, R. Minnich, J. Smith and M-C. Tam, "Memory as a Network Abstraction", IEEE Network Magazine, pp. 34-41, July 1991.

[10] Digital Equipment Corp., "Alpha Architecture Handbook", DEC, Massachussets, February 1992.

[11] M. Dubois, J-C. Wang, L. Barroso, Y-S. Chen, and K. Lee, "Delayed Consistency and its Effects on the Miss Rate of Parallel Programs", Proceedings of the 1991 Supercomputing Conference, November 1991.

[12] M. Dubois, L. Barroso, K. Öner, and Y-S. Chen, "Scalability Problems in Shared Memory Multiprocessors", Proceedings of the PARLE'92, June 1992.

[13] K. Farkas, Z. Vranesic and M. Stumm, "Cache Consistency in Hierarchical-Ring-Based Multiprocessors", University of Toronto, Department of Electrical Engineering, EECG TR-92-09-01. To appear in Supercomputing'92, November 1992.

[14] S. Gjessing, D. Gustavson, J. Goodman, D. James and E. Kristiansen, "The SCI Cache Coherence Protocol", In M. Dubois and S. Thakkar, editors, Scalable Shared Memory Multiprocessors, pp. 219-237, Kluwer Academic Publishers, 1991.

[15] J. Goodman, "Using Cache Memory to Reduce Processor-Memory Traffic", Proceedings of the 10th International Symposium on Computer Architecture, pp. 124-131, June 1983.

[16] A. Gupta, W-D. Weber and T. Mowry, "Reducing Memory Traffic Requirements for Scalable Directory-Based Cache Coherence Schemes", Proceedings of the 1990 International Conference on Parallel Processing, Vol I, pp. 312-321, August 1990.

[17] D. Gustavson, "The Scalable Coherent Interface and Related Standards Projects", IEEE Micro, Vol. 12, No. 1, February 1992.

[18] E. Hafner, Z. Nenadal and M. Tschanz, "A Digital Loop Communication Systems", IEEE Transactions on Communications, COM-22, 6, pp. 877-881, June 1974.

[19] E. Hagersten, A. Landin and Haridi, S., "DDM - A Cache-Only Memory Architecture", IEEE Computer, Vol. 25, No. 9, pp. 44-56, September 1992.

[20] A. Hooper, and R. Needham, "The Cambridge Fast Ring Networking System, IEEE Transactions on Computers, Vol. 37, No. 10, pp. 1214-1233, October 1988.

[21] Kendall Square Research, "Announcing the KSR1 Supercomputer", Internet comp.arch newsgroup posting, January 1992.

[22] D. Lenoski, J. Laudon, T. Joe, D. Nakahira, L. Stevens, A. Gupta and J. Hennessy, "The DASH Prototype: Implementation and Performance", Proceedings of the 19th International Symposium on Computer Architecture, pp. 92-103, June 1992.

[23] D. Menasce, and L. Barroso, "A Methodology for Performance Evaluation of Parallel Applications in Multiprocessors", Journal of Parallel and Distributed Computing, January 1992.

[24] A. Norton and G. Pfister, "A Methodology for Predicting Multiprocessor Performance", Proc. of the 1985 International Conference on Parallel Processing, IEEE Computer Society, pp. 772-781, August 1985.

[25] R. Saavedra-Berrera, D. Culler and T. von Eicken, "Analysis of Multithreaded Architecture for Parallel Computing", 2nd Annual ACM Symposium on Parallel Algorithms and Architectures, Greece, July 1990.

[26] C. Scheurich and M. Dubois, "Lockup-free Caches in High-Performance Multiprocessors", Journal of Parallel and Distributed Computing, Vol. 10, 1990.

[27] S. Scott, J. Goodman and M. Vernon, "Performance of the SCI Ring", Proceedings of the 19th International Symposium on Computer Architecture, June 1985.

[28] H. Schwetman, "CSIM: A C-Based, Process-Oriented Simulation Language", Proceedings of the 1986 Winter Simulation Conference, pp. 387-396, 1986.

[29] J. Singh, W-D. Weber and A. Gupta, "SPLASH: Stanford Parallel Applications for Shared Memory", SIGArch Computer Architecture News, Vol. 20, No. 1, March 1992.

[30] J. Torrellas and J. Hennessy, "Estimating the Performance Advantages of Relaxing Consistency in a Shared Memory Multiprocessor", Proceedings of the 1990 International Conference on Parallel Processing, Vol I, pp. 26-33, August 1990.

[31] Z. Vranesic, M. Stumm, D. Lewis and R. White, "Hector: A Hierarchically Structured Shared Memory Multiprocessor", IEEE Computer, Vol. 24, No. 1, pp. 72-78, January 1991.

[32] R. Zucker and J-L Baer, "A Performance Study of Memory Consistency Models", Proceedings of the 19th International Symposium on Computer Architecture, pp. 2-12, June 1992.

Appendix: A Model for the Performance of Programs on the Slotted Ring

Here we briefly describe our modeling approach to the performance of applications executing on a slotted ring multiprocessor. Our model is similar to what is used by Norton and Pfister in [24] and Menasce and Barroso in [23]. It is an iterative approximate method in which a performance model of the program generates input parameters to solve a performance model of the interconnection, which in turn provides new parameters for the model of the program. This is a hybrid methodology, since the performance model for the program receives as inputs some statistics from trace driven simulations of the program under study. The parameters and formulas used here consider only the snooping protocol for the slotted ring. A model for the directory-based protocol follows a very similar structure, but with different parameters, since metrics that reflect the intensity of read-write sharing have to be taken into account.

The model for the program derives the execution time based on the statistics collected from trace-driven simulation, from basic timings of the architecture, and from estimates of the network contention. The statistics derived from trace-driven simulations are the following:

TABLE A1. Parameters from trace-driven simulations of the program

Parameter	Definition
N_{acc}	Total number of accesses by a processor
N_{lmiss}	Total number of misses to local memory by a processor
N_{smiss}	Total number of misses to shared memory by a processor
N_{inv}	Total number of invalidations sent by a processor
N_{wback}	Total number of write-backs by a processor

The timing parameters from the architecture are the time to satisfy a hit, the time to satisfy a local miss (L_{lmiss}) and the clock cycle of the interconnection. Here we assume that the private cache can respond for hits in one processor cycle (*Pcyc*). Therefore, execution time of the program, considering homogeneous execution in all processor nodes is given by

$$PET = N_{acc}*Pcyc + N_{lmiss}*L_{lmiss} + N_{shmiss}*L_{smiss} + N_{inv}*L_{inv} \quad \text{(EQ 1)}$$

In Equation 1 above, the only unknowns are the latency of misses to shared memory (L_{smiss}) and the latency of invalidations (L_{inv}). These parameters depend not only on the pure latency of remote accesses but also on the contention level in the interconnect, which have to be determined from a performance model of the interconnection. A performance model of an interconnection generally receives as input the average or distribution of the access rate by each processor. Since we are considering a shared memory environment in which messages are either probes (miss requests/invalidation requests) or block messages (messages that carry a cache block), it is important to determine the rates of probes and block messages. These rates can be obtained from Equations 2 and 3 as follows

$$\lambda_{probes} = \frac{N_{smiss} + N_{inv}}{PET} \times N_{proc} \qquad \text{(EQ 2)}$$

$$\lambda_{block} = \frac{N_{smiss} + N_{wback}}{PET} \times N_{proc} \qquad \text{(EQ 3)}$$

where N_{proc} is the number of processors in the system. We now derive a simple performance model of the slotted ring similar to the one used by Bhuyan and others in [5]. The throughput of the slotted ring for probes and block messages is given as below,

$$\mu_{probes} = \frac{P_{slots}}{pipesize \times R_{clock}} \qquad \text{(EQ 4)}$$

$$\mu_{probes} = \frac{B_{slots} \times 2}{pipesize \times R_{clock}} \qquad \text{(EQ 5)}$$

where P_{slots} and B_{slots} denote the number of probe and block slots respectively, *pipesize* is the number of pipeline cycles in the slotted ring, and R_{clock} is the ring clock cycle time. The factor of two in the nominator of Equation 5 comes from the fact that a block message travels only half the ring, on the average. From Equations 2 to 5 we can calculate the average slot utilizations for probes and block messages. From this point on we proceed only with the derivation for probe messages, since the one for block messages is identical. The utilization of probe slots is given by Equation 6. and it can be seen as the probability that a given slot will be busy, i.e., not available.

$$U_{probes} = \frac{\lambda_{probes}}{\mu_{probes}} = 1 - p_0 \qquad \text{(EQ 6)}$$

When a processor has a message ready to send it first has to wait until the first appropriate slot passes through it. If we assume that the arrival process is Poisson distributed, the residual time to find the beginning of a slot can be considered to be uniformly distributed between *0* and T_{frame}, where T_{frame} is the time interval between two slots of the same type. Therefore, we can write the average waiting time to find an empty probe slot as

$$W_{probes} = T_{frame} \times \left(1/2 + \sum_{i=1}^{\infty} i \times U_{probes}{}^{i} \times (1 - U_{probes})\right) \qquad \text{(EQ 7)}$$

which reduces to

$$W_{probes} = T_{frame} \times \left(1/2 + \frac{U_{probes}}{1 - U_{probes}}\right) \qquad \text{(EQ 8)}$$

Notice that there is no queueing effect in each node since we consider that a processor can have at most one outstanding request pending at any time. Now, the unknown terms in Equation 1 can be calculated as follows

$$L_{smiss} = W_{probes} + R_{clock} \times pipesize + L_{lmiss} + W_{blocks} \quad \text{(EQ 9)}$$

$$L_{inv} = W_{probes} + R_{clock} \times pipesize \quad \text{(EQ 10)}$$

We start by setting W_{probes} and W_{blocks} to zero, and iterate until the percentile error between successive iterations is less than a given threshold. Menasce and Barroso [23] proved that as long as the latency of interconnection accesses is a monotonically non-decreasing function of the request rate, this iteration converges. The convergence of our particular implementation of this method is very fast, seldom requiring more than 10 steps. A similar model was derived for the performance of a packed-switched bus. The model predictions for processor and ring slot utilizations shows very good accuracy for the 8, 16 and 32 processor configurations. Predictions of message latency are not as accurate, nevertheless they are still useful in determining an approximate tendency of the curves. The split transaction bus model was also very accurate in predicting utilization and latency parameters, with respect to the results of the simulations. Figures A1-A3 show some sample results of the validation experiments for the slotted ring model.

FIGURE A1. MP3D: Slotted ring with 20ns processor cycle (model vs. simulation)

(+) 8 processors; (*) 16 processors; (o) 32 processors; (...) model

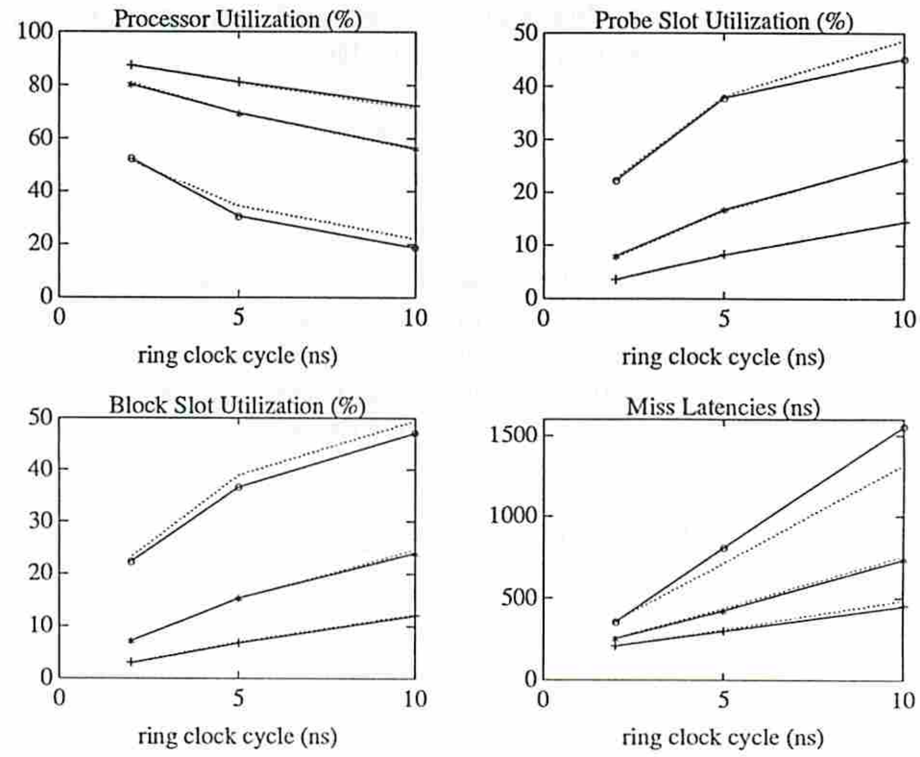


FIGURE A2. WATER: Slotted ring with 20ns processor cycle (model vs. simulation)

(+) 8 processors; (*) 16 processors; (o) 32 processors; (...) model

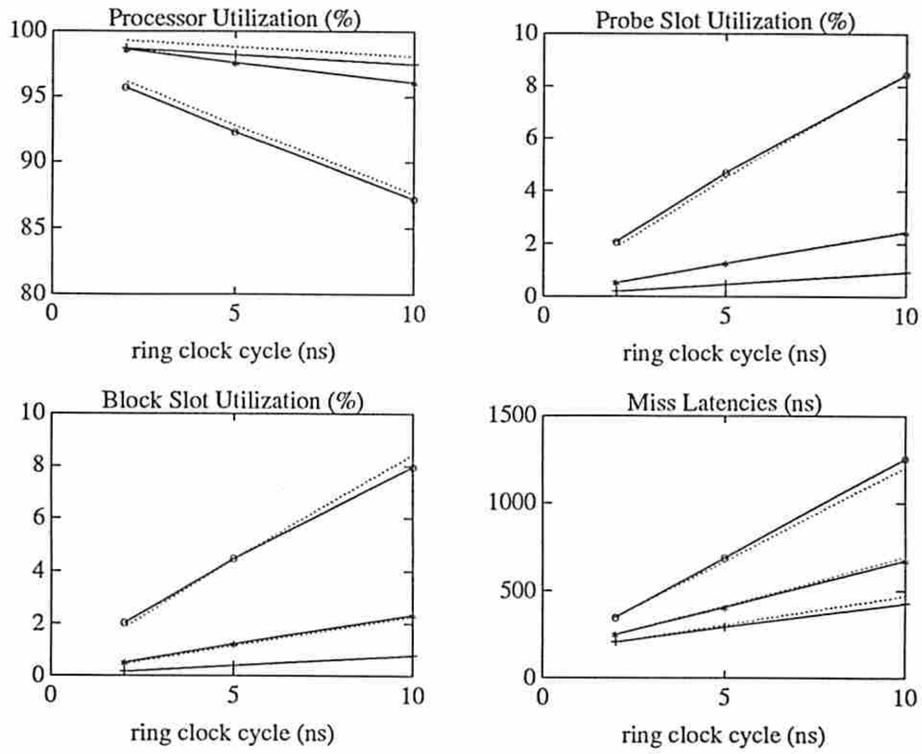
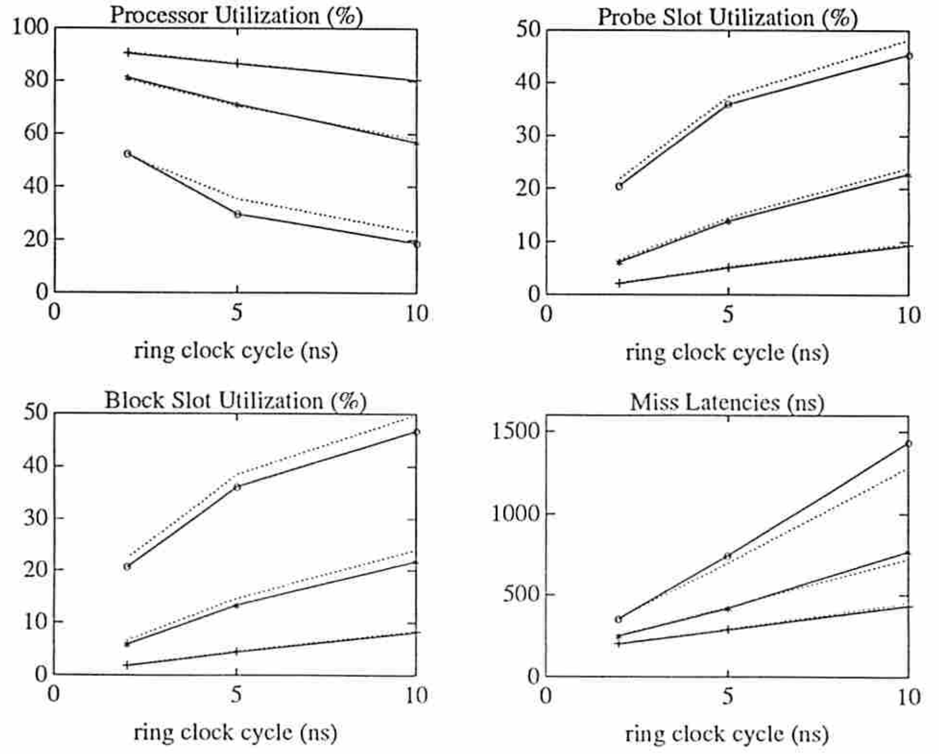


FIGURE A3. CHOLESKY: Slotted ring with 20ns processor cycle (model vs. simulation)

(+) 8 processors; (*) 16 processors; (o) 32 processors; (...) model


Processor Utilization (%)
Probe Slot Utilization (%)
Block Slot Utilization (%)
Miss Latencies (ns)
ring clock cycle (ns)