Appeared in the Proceedings of the 20th Intl. Symp. on Computer Architecture, May 1993

The Performance of Cache-Coherent Ring-based Multiprocessors

Luiz André Barroso and Michel Dubois
barroso@paris.usc.edu; dubois@paris.usc.edu

Department of Electrical Engineering - Systems
University of Southern California
Los Angeles, CA 90089-2562

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 the speed of shared buses is unlikely to keep up with the bandwidth requirements of new microprocessors. In this paper we evaluate the performance of the unidirectional slotted ring interconnection for small to medium scale shared memory systems, using a hybrid methodology of analytical models and trace-driven simulations. We evaluate both snooping and directory-based coherence protocols for the ring and compare it to high performance split transaction buses.

1.0 Introduction and motivations

In the last decade, parallel processing has become the consensus approach to high-performance computing. Virtually 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 shared memory 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 [8] with a peak performance of 400 MIPS.

This research was supported by the National Science Foundation under Grant No. CCR-9115725. Luiz Barroso is supported by CAPES/MEC, Brazil, Grant No. 1547/89-2.

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 severe problems related to the bus topology itself. First of all, the bus is a mutually exclusive resource and only one processor can transmit at any given time. Second, all processors must participate in an arbitration phase before accessing the bus. Third, the bus clock cycle must be long enough so that signals can propagate throughout the entire extension of the bus. Lastly, a transmitter on the bus drives several receivers at the same time, each receiver adding to the characteristic impedance of the bus and reducing the signal propagation speed. 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. Point-to-point connection links have only one transmitter and one receiver (one at each end) and can be very short depending on the packaging. Their characteristic impedance is very small and, if they are terminated properly, the transmission speed is much higher than on buses. Additionally, signals can be pipelined: 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 are 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 is demonstrated by the IEEE Scalable Coherent Interface (SCI) [12] set of standards, based on 500 MHz 16-bit wide links in its first generation of circuits.

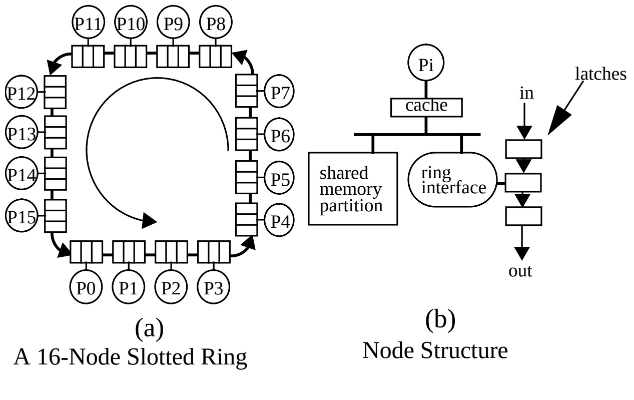
In this paper we evaluate the unidirectional slotted ring as an alternative to buses for cache-based multiprocessor systems with up to 64 processors and in the context of multitasking. The slotted ring architecture is described in the next section. Section 3 briefly explains a snooping cache coherence protocol first proposed in [1] 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 high-end split transaction buses. Related work is discussed in section 5. Final remarks and conclusions are drawn in section 6.

2.0 The slotted ring

The unidirectional ring is the simplest form of point-to-point interconnection. In particular, the unidirectional ring requires the simplest routing mechanism possible: The only routing decision is whether to remove a message from the ring or to forward it to the next node. 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 the board logic will eventually become a performance bottleneck, and therefore simple and fast routing mechanisms are critical.

The general architecture of the unidirectional ring is shown in Figure 1, and consists of a set of processing elements with a CPU, a 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 so that the interconnection behaves as a circular pipeline. The main function of the latches is to hold an incoming message 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.

Figure 1: The unidirectional ring interconnect



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. In a cache-coherent system there are at least two types of messages, which we call *probe messages* (or probes) and *block messages*. Probes are short messages carrying miss or invalidation requests, and consisting typically of a block address field and other control/routing information. Block messages are made up of a header and a cache block, and are needed for misses and write-backs; the header format is similar to that of a probe.

To avoid breaking messages into packets, the protocol must transmit messages in consecutive pipeline stages and consecutive empty stages must be freed to fit a particular message. There are three main solutions to this problem: token passing rings, register insertion rings and slotted rings. In token passing rings, a special bit pattern, called token, is passed on from node to node and the node with the token is allowed to transmit. The main disadvantage of token passing is that only one message may travel on the ring at a time. 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. Finally, in the slotted ring, the ring bandwidth is divided into marked message slots with different sizes and a processor ready to transmit a message waits for an empty slot with the same size as the message. The slotted ring restricts the utilization of the ring bandwidth because the mix of message slots is pre-determined. To limit the impact on performance, the mix of slots must match the expected mix of messages.

Which one of slotted or register insertion rings offers the best performance is not clear. Intuitively, under light loads, the register insertion ring has a faster access time since a message does not wait for a proper slot to pass by. Under medium to heavy loads, the simplicity of enforcing fairness on the slotted ring may yield better performance. 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. A slotted ring interface is also simpler because it does not require a bypass buffer. 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 is not suited for a register insertion ring. An implementation of snooping on a token ring is presented by Delp and others in [7].

3.0 Coherence protocols for the slotted ring

3.1 A snooping protocol

In contrast to other point-to-point interconnections [5,6], the unidirectional slotted ring allows efficient snooping implementations due to its similar cost for broadcasting and for unicasting. Snooping protocols require less state information at the memory modules and arguably are less complex than directory-based protocols.

The snooping cache coherence protocol for the slotted ring, introduced in [1], 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 of that block. The node which has a WE copy of a block is called the *dirty* node. A *dirty bit* per block in memory indicates when a block is cached in WE state. When the dirty bit is set, the dirty node is the owner and is responsible for responding to coherence requests, whereas, when the dirty bit is reset, the home node responds. 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. Only the owner acknowledges a probe message. Coherence actions at the snooper are consistent with typical write-invalidate protocols.

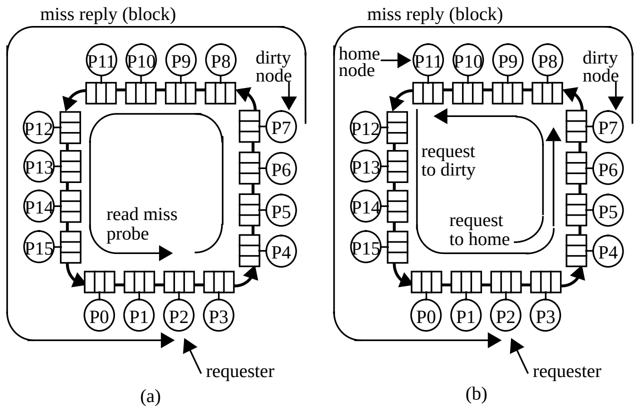
The most important feature of this snooping protocol is that no coherence transaction traverses the ring more than once because probes are snooped on without being removed from the ring; the owner simply acknowledges a probe in an acknowledgment field in the following probe slot of the same type. The latency of misses is independent of the relative positions of the requesting node and the owner. Therefore, the slotted ring with a snooping protocol behaves as a uniform memory access (UMA) interconnect, just like a shared bus.

3.2 A directory-based protocol

The full map directory-based protocol has the same cache states as the snooping protocol. The home node knows whether the block is dirty, and which nodes currently have valid copies by maintaining one set of presence bits (one per node in the system) and one dirty bit per block in a directory [5]. All coherence requests are first sent to the home node, which looks up the directory entry for the block and takes the appropriate coherence actions.

Remote read misses on clean blocks take only one trip around the ring, since they involve the requester and the home node only. Whenever the home node is not the owner, the 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 shown in Figure 2.b. For each write miss and each invalidation 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; (b) directory



We have chosen a full map directory rather than other directory organizations such as limited directory [6] or linked list [12] to compare the snooping scheme with the most efficient directory protocol. Whereas linked-list and limited directory protocols save memory and reduce directory contention especially in large scale systems, they are not likely to outperform full map directories in the context of the slotted ring.

In a linked list protocol such as the one adopted by SCI [12], each blockframe 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 head of the sharing list (the *head* node), which is responsible for maintaining the coherence of the block. Each miss request to a cached block is first transferred to the home node, which then forwards the request to the head node; this transaction requires one or two ring traversals, depending on the relative positions of the requester, the home and the head. Invalidating the sharing list also takes extra ring traversals when the order of 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 1 compares the distribution of remote misses

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.

and invalidations requiring 1, 2 and 3 or more ring traversals in the linked list and full map directory protocols and for three 16 processor benchmarks (see Section 4.1.)

Table 1: Distribution of the number of ring traversals full directory vs. linked list (values in %)

Ring Traversals	**MP3D 16**				**WATER 16**				**CHOLESKY 16**			
	Miss		Invalidate		Miss		Invalidate		Miss		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 protocols are very dependent on the application 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; the full map protocol also generates less traffic than the snooping protocol since the probes are not broadcasted and it might outperform snooping because of the reduced contention. 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, as shown Figure 2.b: if processors *P2* and *P7* share a block in read-write mode, the configuration depicted always occurs for either *P2* or *P7*, no matter where the home node of the block is located (unless it is *P2* or *P7*).

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 cannot respond at the maximum rate of incoming probes (which, by the way, makes register insertion rings unsuitable for snooping protocols.) The slotted ring access control mechanism proposed in [1] overcomes this problem. In this scheme, 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. If the dual directory in the ring interface is 2-way interleaved, two consecutive probes for the same dual-directory bank are always separated by at least one frame, which is no less than 20 nsec. for a 32-bit wide ring using a 16-byte cache block size and clocked at 500 MHz (2 nsec.).

The mix of 2 probe slots per block slot is optimum in the snooping protocol because the numbers of probes and of block messages generated in actual simulations are approximately the same, and probes traverse the whole ring whereas block messages are removed by the destination and travel through half the ring on the average. The optimum mix of probe and block slots is not as predictable for the directory scheme, because transactions do not always commit in a single ring traversal. We also chose a mix of 2 probe slots for each block slot for the directory protocol since this was the mix showing the best performance in our simulations.

4.0 Performance evaluation

Our performance evaluation methodology is a hybrid one relying on both trace-driven simulations and analytical models [6,14]. Very detailed trace-driven simulations of a limited number of configurations are first performed to gain better understanding of the interactions between the programs and the architectures. Then simple analytic models are formulated to capture the essential performance characteristics. The input parameters for the models are derived from simulations, and the outcome of each model is validated by simulations. We used an approximate iterative methodology similar to Menasce and Barroso's [14] for the analytical models. In this methodology, an estimate of the average latencies of memory requests is used to calculate an estimate of the program execution time, which in turn is used to estimate new values for the average latencies, iterating until convergence. A detailed description of the models can be found in [2], along with validation results.

This hybrid approach gives us the accuracy expected from detailed simulations as well as the efficiency of analytical models to explore the design space. 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 whereas the analytical models typically take less than a second of CPU time to produce a complete curve.

All the evaluations reported in this section are performed by first simulating each benchmark for each value of network bandwidth and with 50 MIPS processors; the simulations generate parameter values describing the average behavior of each system, including a count of each type of relevant coherence events. These values 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.

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 [18]: MP3D, WATER and CHOLESKY. These programs were traced using the CacheMire simulator [3] developed by Per Stenstrom's group at Lund University, Sweden, and traces were obtained for systems with 8, 16 and 32 processors. Traces for the second set of benchmarks were obtained from Anant Agarwal's group at MIT [6], and are 64-processor traces of three parallel FORTRAN programs: FFT, WEATHER and SIMPLE. These are all well-known scientific benchmarks and the reader is referred to [18] and [6] for insight into the structure of the programs.

Table 2: Trace characteristics (references in millions)

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

The main characteristics of the traces derived from each benchmark are shown in Table 2. The miss rate values are derived with 128 Kbytes direct-mapped data caches and a block size of 16 bytes. We further assume that instruction references never miss in order to reduce the simulation times. This assumption does 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 and that the time to access a local memory bank is fixed at 140 nsec. for all systems. These assumptions remain fixed throughout the rest of the paper. The simulations of the ring and bus systems were developed using the CSIM package [17] which is a library of C functions for process-oriented simulation.

4.2 Snooping vs. directory for the slotted ring

The comparison between the snooping and the directory-based protocols for the slotted ring is shown in Figures 3 to 5. Processor utilization[2], average ring slot utilization and average miss latency 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.), the links are 32-bit wide, and the processor cycle time varies from 1 to 20 nanoseconds. The processors execute each instruction in one cycle as long as accesses hit in the cache. Hence, a processor cycle of 20 nsec. means 50 MIPS of processing power.

Using 16-byte blocks, a frame composed of two probe slots and one block slot occupies 10 pipeline stages. With a minimum of 3 stages per node, the number of stages in the ring is 24 for an 8-node system. Six extra stages are added to accomodate an integer number of frames (3). As a result, the pure round-trip latency for an 8-node 500 MHz ring is 60 nsec. Figure 5 shows the distribution of the latencies of remote misses for the directory protocol. *1-cycle clean misses* are misses to clean blocks mapping to a remote home and taking one ring traversal; *1-cycle dirty misses* are misses to dirty blocks which also require 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. The latency of 1-cycle dirty misses is higher than that of 1-cycle clean misses since they need three hops to commit instead of two. For MP3D, WATER and CHOLESKY, the fraction of 1-cycle clean misses increases steadily as the system size increases. This is a result of the random allocation of shared memory pages among the nodes. When the number of processors increases, a smaller fraction of the shared memory is allocated to each node and the fraction of clean misses which are remote goes up.

The snooping protocol outperforms the directory-based protocol for all system sizes in the case of 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, as expected. Nevertheless, it takes a ring utilization of over 70% — as is the case for the 32 processor systems with a processor cycle time of less than 5 nsec. — for the latency of snooping to approach the latency of the directory protocol.

For WATER, the high hit ratio hides most differences between the snooping and directory-based protocols in terms of processor and ring utilizations. The average miss latency values confirm the impact of the higher latency of 1-cycle dirty and 2-cycle misses. CHOLESKY has a smaller fraction of 1-cycle dirty and 2-cycle misses than WATER and MP3D for each system size, and the gap between the latencies of misses of the two protocols is not

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

Figure 3. Snooping vs. directories; 500 MHz 32-bit rings

8 processors:		16 processors:		32 processors:	
———	snooping	– – –	snooping	++++	snooping
....	directory	–.–.–	directory	xxxx	directory

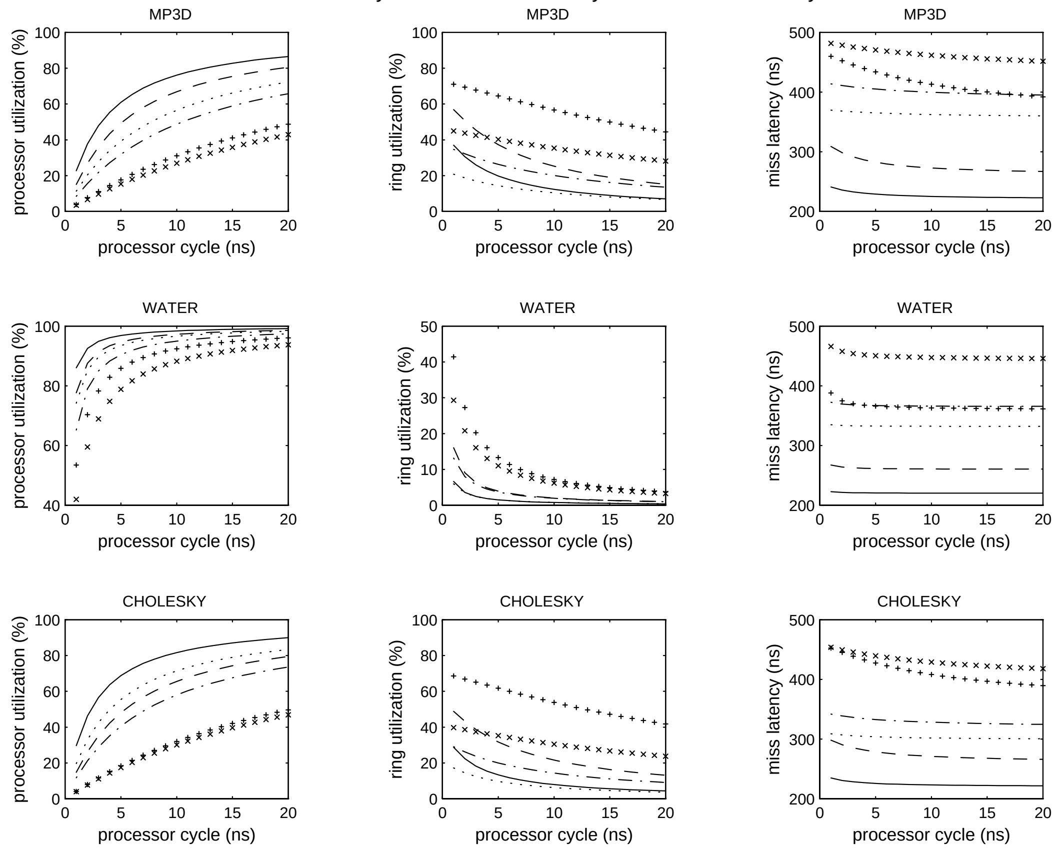


Figure 4. Snooping vs. directories; 500 MHz 32-bit rings

FFT		SIMPLE		WEATHER	
———	snooping	– – –	snooping	++++	snooping
....	directory	–.–.–	directory	xxxx	directory

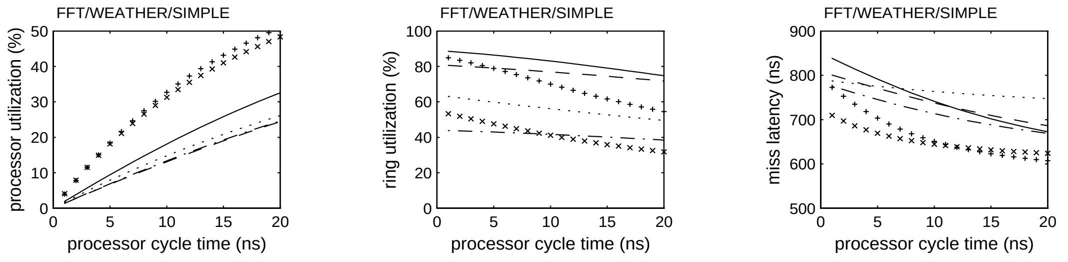
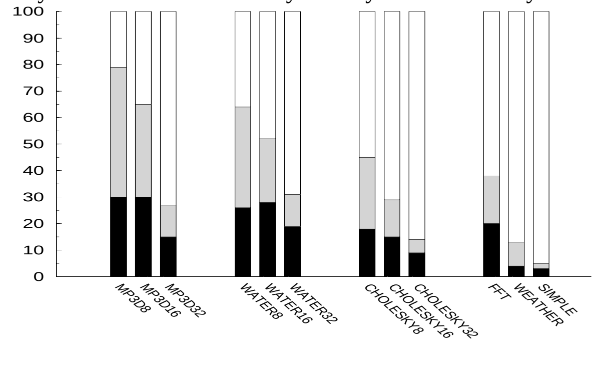


Figure 5. Breakdown of the types of misses in the directory protocol

☐ 1-cycle clean misses ▨ 1-cycle dirty misses ■ 2-cycle misses



as wide. As in MP3D, the only case where the miss latency of the directory protocol approaches the miss latency of snooping is for 32 processors, when the ring utilization for snooping is significantly higher.

For FFT, SIMPLE and WEATHER, which are 64 processor benchmarks, the processor utilization is considerably lower because of higher latencies. Again, the correlation between the mix of remote misses and the differences in performance between the two protocols is obvious. Among the three benchmarks, FFT is the only one that shows a significant number of 1-cycle dirty and 2-cycle misses; consequently, the average miss latency of the snooping protocol is shorter than for the directory-based protocol when ring utilization values are relatively low. This is in contrast with SIMPLE and WEATHER, which exhibit a very small fraction of higher latency misses. In WEATHER, as the processor cycle decreases, the latencies of snooping surpass those of directory because of network contention.

In general, for ring utilizations below 70%, the latencies of the snooping protocol are lower than for the directory protocol. Only when the ring utilization for snooping is high the average miss delays could favor the directory scheme. Our simulation experiments with a 64-bit parallel slotted ring (not shown here) agree with this assessment. With 64-bit parallel rings, utilization levels never surpass 50% and snooping performs significantly better than directory in all cases.

The snooping implementation requires faster ring interface logic, which may be more costly than for the directory protocol. Therefore, in design regions where the performance gap is not very significant, a directory implementation may be preferred. The cost of snooping is mainly affected by the minimum inter-arrival time of probes to a given dual-directory bank, which depends on the ring width, clock cycle, and cache block size. Table 3 displays the probe inter-arrival times for various ring widths and block sizes, considering a 2-way interleaved dual directory and 500 MHz ring links.

Table 3: Snooping rate (nsec.)

block size	ring data width (bits)		
	16	32	64
16 bytes	40	20	10
32 bytes	56	28	14
64 bytes	88	44	22
128 bytes	152	76	38

4.3 Slotted ring vs. split transaction bus

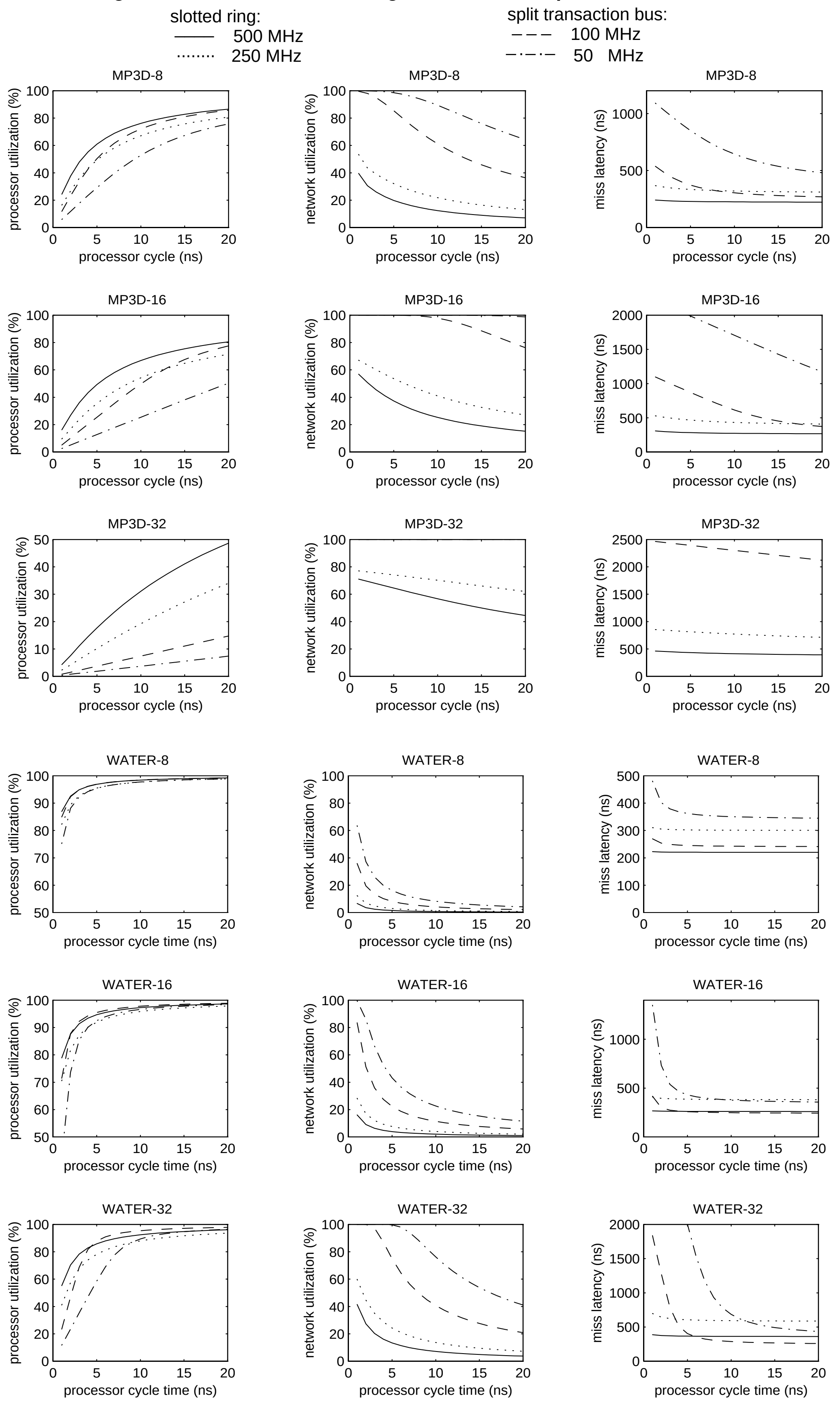
We now use the snooping protocol to compare the performance of the slotted ring with that of a shared split transaction bus. The bus architecture is similar to a split transaction version of the FutureBus+ (IEEE 896.x standard), with a 3-state write-invalidate snooping protocol and physical shared memory partitioned among the processing nodes. Figure 6 compares the performance of 32-bit wide rings, clocked at 250 MHz and 500 MHz, to 64-bit wide buses, clocked at 50 MHz and 100 MHz. The bus parameters were chosen to represent aggressive values considering current technology and the range of system sizes.

The bus clock cycle is constant across 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, whereas it increases roughly linearly with the number of nodes for the ring case. With a 16-byte cache block, the minimum number of bus cycles for a remote miss is six, excluding arbitration delays and the time to fetch the block in the remote memory or cache. 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 are more stable.

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 a given system size. MP3D has a relatively high miss ratio for shared data (see Table 2), 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 250 MHz ring for slower processors ($\leq$ 50 MIPS), but it falls behind for increasingly faster processors due to bus conflicts. For the 16 processor MP3D the performance gap between ring and bus configurations increases as the buses enter saturation while the ring utilization remains under 50% even for 100 MIPS processors. In the 32 processor MP3D both buses are completely saturated, whereas the ring utilization stays under 80%. The behavior of CHOLESKY is very similar to MP3D and is not reported here for lack of space.

The evaluations using WATER show a different behavior. In this case the miss rate values are extremely low, as is the fraction of references to shared data. The load on the interconnect is much lighter than in MP3D. For 8 and 16 processors, the buses only start to saturate for processor speeds higher than 200 MIPS. Even for 32 processors, the two bus systems still show good performance levels for 100 MIPS processors. For the 16 and 32 processor configurations, the pure latencies of the 50 MHz bus and of the 100 MHz bus are shorter than those of the 250 MHz and of the 500 MHz ring (respectively); hence, the bus configurations could outperform the slotted

Figure 6. 32-bit wide slotted ring vs. 64-bit wide split transaction bus



rings for slower processors even if only by a narrow margin. However, in all cases, the slotted ring is less affected by contention delays. Eventually, as the buses reach saturation, the ring configurations show far better performance figures.

For 64 processors the bus systems are completely saturated in all cases. It is also unlikely that buses for 64 processors can be clocked even at 50 MHz, using current bus technology. Therefore, we will not compare buses and rings for 64 processors.

We have applied the hybrid performance model to determine the bus clock cycle required of a 64-bit bus system to reach the same processor utilization (i.e., the same program execution time) as 32-bit slotted ring systems clocked at 250 and 500 MHz, for processor speeds of 100, 200 and 400 MIPS. These results are shown in Table 4. For systems with 8 processors, bus speeds ranging from 80 to 100 MHz are required to match the performance of a 250 MHz ring (with the exception of WATER). It takes bus clock frequencies of 100 to 170 MHz to compete with the 500 MHz ring in systems with 8 processors. For the 16 processor configurations, the range of bus cycles matching the performance of a 500 MHz ring becomes even more challenging, falling between 150 to 300 MHz. For 32 processors, the bus systems to match the ring systems are probably impractical, considering that high-performance buses are more difficult to build for larger number of processors. For all the cases shown in Table 4, the utilization levels of the buses matching ring performance are significantly higher than those of the ring slots. With the exception of WATER8 and WATER16 the bus shows utilization levels above 50%, and is frequently saturated. By contrast, the slotted ring utilizations are seldom above 50%, and never over 75%.

Table 4: Bus clock cycle (nsec.) to match the performance of slotted ring configurations

Benchmarks	250 MHz Ring			500 MHz Ring		
	(MIPS) 100	200	400	(MIPS) 100	200	400
MP3D 8	12.5	10.3	8.9	7.8	6.6	5.6
WATER 8	19.6	19.1	17.7	10.0	10.0	9.9
CHOLESKY 8	12.8	10.6	9.0	7.6	6.6	5.7
MP3D 16	9.0	7.1	6.2	6.5	4.9	4.0
WATER 16	25.4	21.4	16.5	14.1	12.9	10.9
CHOLESKY 16	6.8	5.4	4.7	4.9	3.7	3.1
MP3D 32	3.8	3.7	3.6	2.4	2.1	2.0
WATER 32	21.4	13.9	9.2	16.2	11.0	7.3
CHOLESKY 32	3.7	3.5	3.4	2.3	2.0	1.9

Considering that today's high speed buses are clocked at 10 to 30 nsec. and that, barring any breakthrough in bus technology, these values are expected to improve rather slowly, it is likely that new bus systems with state-of-the-art microprocessors will be limited to up to 8 processors.

The evaluation results showed here also indicate that the slotted ring could benefit from latency tolerance techniques, such as lockup-free caches, weak ordering schemes [9,11] and multithreaded processors [15] because the large latencies observed for the slotted ring are, in most cases, 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 would probably happen in a split transaction bus using very fast processors. The latencies for the slotted ring however are still relatively stable and the network never saturates in the configurations that we have simulated. This indicates that the ring would be able to accommodate the increase in the load without significantly altering the expected latencies.

5.0 Related work

The performance of unidirectional ring interconnections has been the subject of extensive analysis in the context of Local Area Networks [4]. In the context of distributed systems, Delp et al proposed a token ring distributed shared memory system (Memnet [7]) with cache coherence maintained in hardware by means of a snooping-like coherence protocol. More recently Scott et al [16] 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 the starvation of a node. The mechanism proposed by SCI to avoid starvation is shown to impact the effective throughput of the ring. By contrast, starvation of clusters in the slotted ring architecture is easily avoided by preventing a node from reusing a message slot immediately after removing a message from that slot. Our simulations show that this has no significant impact on system performance. Scott also compares a shared bus to the register insertion ring without considering the cache coherence level.

The Hector multiprocessor [19], under development at the University of Toronto, uses a hierarchy of unidirectional slotted rings. 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 [10] recognizes the need for it and specifies a hierarchical cache protocol based on the broadcasting of requests, as in the snooping protocol used here. Finally, the Kendall Square Research KSR1 [13] is a shared memory multiprocessor commercially available and based on a two-level hierarchy of unidirectional slotted rings with a snooping cache protocol. The implementation of snooping in the KSR1 is not as efficient as the one studied here.

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 common wisdom, the snooping strategy outperforms the directory-based strategy for nearly all system configurations analyzed. The differences in performance between the two protocols can be explained from the particular data sharing patterns of each benchmark.

We have also compared the performance of the slotted ring with that of a split transaction bus under snooping. The results show that, even for systems with as little as 8 processors, the slotted ring can outperform a pipelined split-transaction bus for benchmarks with a significant fraction of remote misses and invalidations. Whereas the speed of bus interconnections is expected to lag further and further behind with respect to microprocessors, we expect the speed of ring interconnections to keep up with future generations of microprocessors.

7.0 References

[1] L. Barroso and M. Dubois, “Cache Coherence on a Slotted Ring”, Proceedings of the 1991 International Conf. on Parallel Processing, Vol. I, pp. 230-237, August 1991.

[2] L. Barroso and M. Dubois, “The Performance of Cache-Coherent Ring-based Multiprocessors”, University of Southern California, Technical Report No. CENG 92-19, November 1992.

[3] M. Brorsson, F. Dahlgren, H. Nilsson and P. Stenström, “The CacheMire Test Bench - A Flexible and Efficient Approach for Simulation of Multiprocessors”, Proceedings of the 26th Annual Simulation Symposium, March 1993.

[4] L. Bhuyan, D. Ghosal, and Q. Yang, “Approximate Analysis of Single and Multiple Ring Networks”, IEEE Trans. on Computers, Vol. 38, No. 7, pp. 1027-1040, July 1989.

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

[6] 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.

[7] 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.

[8] Digital Equipment Corp., “Alpha Architecture Handbook”, DEC, Massachussets, February 1992.

[9] M. Dubois and C. Scheurich, “Memory Access Dependencies in Shared Memory Multiprocessors”, IEEE Trans. on Software Eng., 16(6), pp. 660-674, June 1990.

[10] K. Farkas, Z. Vranesic and M. Stumm, “Cache Consistency in Hierarchical Ring-Based Multiprocessors”, Proceedings of Supercomputing’92, November 1992.

[11] A. Gupta, J. Hennessy, K. Gharachorloo, T. Mowry and W.D. Weber, “Comparative Evaluation of Latency Reducing and Tolerating Techniques”, Proceedings of the 18th Intl. Symp. on Computer Architecture, May 1991.

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

[13] Kendall Square Research, “Technical Summary”, Walthan, Massachusetts, 1992.

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

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

[16] S. Scott, J. Goodman and M. Vernon, “Performance of the SCI Ring”, Proceedings of the 19th Intl. Symp. on Computer Architecture, June 1985.

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

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

[19] 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.