Ring-Based Multiprocessors

Zvonko Vranešić
Department of Electrical and Computer Engineering, University of Toronto

Ring interconnection networks provide an attractive medium for use in large-scale multiprocessor systems. They require simple interfaces and allow transmission at high clock rates. They can be easily organized into race-free networks, which facilitates implementation of consistency protocols. The paper examines the key features of ring networks and considers some practical examples of their use.

1. Introduction

One of the great challenges in computer architecture is to provide supercomputer performance at a reasonable price. This goal will be attained in the near future with multiprocessor systems that exploit the technology used in affordable workstations. Such systems will be of MIMD (Multiple Instructions and Multiple Data) type, and will use processor modules based on microprocessors developed for use in workstations. Their commercial success will depend not only on their computational capability, but also on the cost/performance ratio. Moreover, successful products will be those that will allow configuration of a viable entry-level machine at a correspondingly low cost, which could then be expanded into a large system merely by acquiring additional hardware modules of essentially the same kind. In order to reach these low cost and easy expandability objectives, the architecture will have to feature simplicity and scalability.

A critical aspect of a multiprocessor system is its backplane, namely, the interconnection network used to connect the processing modules. Efficiency, economy, reliability, and physical realizability of the backplane are the essential features in any large multiprocessor. In this paper we focus on the multiprocessor architectures that use ring-based backplanes, which we believe possess the desired characteristics. We will examine the key aspects of such architectures, guided largely by our experience with the Hector multiprocessor project at the University of Toronto.

Section 2 highlights the key characteristics of ring networks. Sections 2 and 3 discuss some interesting features of two ring-based multiprocessors that have been built recently, the KSR-1 machine which has gained considerable commercial popularity and the Hector multiprocessor that has been developed at the University of Toronto. Section 5 considers the newly defined SCI Standard, which is conducive for use with a ring interconnection medium.

2. Ring Networks

Ring interconnection networks are characterized by point-to-point connections between successive nodes. This allows usage of simple interfaces, since the ring connects to a given node by means of only one input and one output port. This simplicity reflects itself in a relatively low requirement for the number of connecting wires, which often correspond directly to the number of pins on physical connectors. Typically, the number of connections is likely to be considerably smaller than in hypercube interconnection networks or any networks where there exists more than one direction for incoming and outgoing signals.

Point-to-point connections do not suffer from undesirable effects such as loading and signal reflections from multiple connectors, which plague bus-based schemes and effectively reduce their viability to small sizes. Therefore, signals can be transmitted on such links at high clock rates. Since the bandwidth achievable is directly dependent on the clock rate, it is clear

that rings are suitable for implementation of networks with very high bandwidth.

High bandwidth is the design goal for all large multiprocessors, in order to prevent the interconnection network from becoming the bottleneck in the system. One often encounters a mistaken notion that the effective bandwidth can be improved by increasing the size of a ring, namely by including more nodes in the ring. Clearly, this increases the amount of information being transferred on the ring at any given time, but it also increases the latency of the transfers since the information packets stay on the ring longer. The effective bandwidth is essentially determined by the transfer rates attainable at individual nodes, and it can be improved only by increasing the clock frequency or by increasing the width of the transfer path.

Another way of increasing the bandwidth of a multiprocessor interconnection network is by means of a hierarchical structure, whereby a number of localized transfers can take place concurrently on several rings. For example, if several local rings are connected by means of a central ring, then the number of concurrent transfers that can be supported is much higher if the transfers are only between sources and destinations on the same local ring. Transfers that pass through the central ring take more time than local ring transfers, but they are in general shorter than they would be if all nodes were connected to a single long ring.

High transfer rates that can be attained on rings may exceed the speed capability of protocol handling access circuitry in processing modules connected to the nodes. This can be used constructively by running the ring at a higher clock rate and synchronizing with the access circuitry by means of buffers. An attractive possibility is to include a pipelined buffer in the ring path in each node, such that the transmitted data becomes available to the access circuitry when it reaches the first stage of the buffer and remains available (for possible alteration) until it leaves the last stage.

3. KSR-1 Multiprocessor

In 1991, the Kendal Square Research company produced a multiprocessor, called KSR-1, based on a two-level hierarchy of rings [1]. The basic structure of this machine is shown in Figure 1. Processing modules are connected by means of low-level rings, each of which has 32 nodes. These rings are interconnected via a high-level ring that has up to 34 nodes.

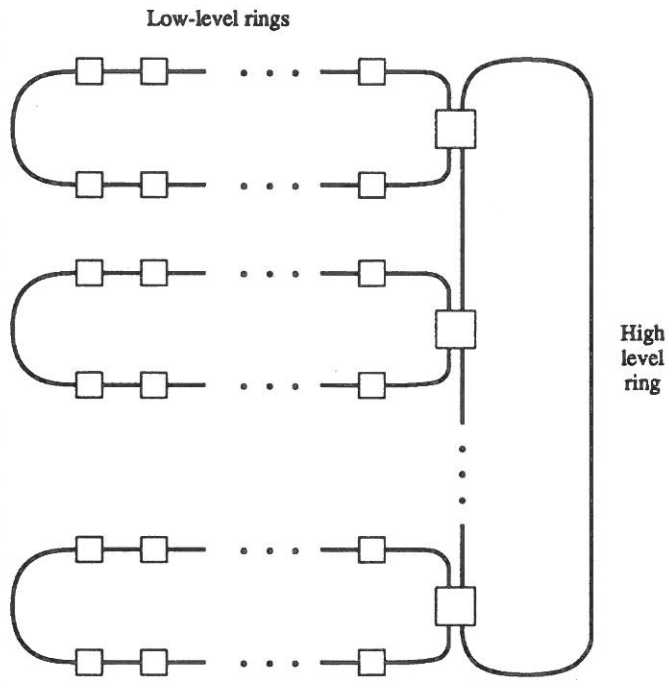

Fig. 1. KSR-1 structure

A distinguishing aspect of the KSR-1 architecture is the organization of its memory. All of the memory is treated as cache, known as COMA (Cache-Only Memory Architecture) model. This means that data corresponding to a specific memory address is not bound to a predetermined physical location. When a processor attempts to access data that is not found within its module, this data has to be found and brought to this module. This is done by sending a request message around the rings. As the message traverses a ring, each node observes the request and the first node that actually has the desired data responds to the message. In response to a read request the data is sent to the requesting node, which means that now there exist at least two copies of this data. In case of a write request, it may be necessary to invalidate all other copies of the data to ensure consistency. This is achieved using a broadcast

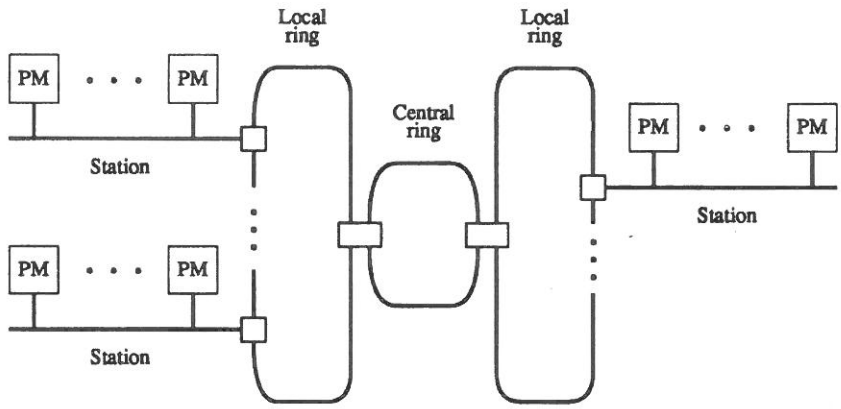

Fig. 2. Hector structure

mechanism, by sending an invalidate message through the rings.

The success of the COMA model rests on the fact that the ring structure guarantees that a message always traverses the rings in the same order, which is essential for ensuring consistency when several processors are reading and writing the same data concurrently. In order to provide a sequentially consistent view of the physical address space, a Multiple-readers/Single-writer protocol is used.

Since each memory module is treated as a "cache", it is necessary to maintain a directory associated with each module indicating the physical address of the data found in that module. This is achieved by partitioning the memory into 16K-byte pages and further subdividing each page into 128 subpages of 128 bytes each. A memory module can store up to 2048 pages, but it is possible that only some subpages of a given page contain valid data. Therefore, the directory entries are associated with the pages and they comprise the address of the page and status information for each subpage.

A key advantage of the COMA scheme is that a high degree of locality in data accesses can be achieved, because the data is migrated to the processing nodes where it is used. This scheme also helps to reduce congestion at specific memory modules, because it tends to distribute evenly the access requests across all nodes in the system at run time; thus diminishing the importance of initial data allocation. COMA also has some negative aspects. Since it is possible that only a small fraction of subpages of a given set of pages have valid data, there may occur a certain underutilization of the memory capacity. Also, the design of the node interface is complicated by the fact that the directory information must be looked up in a short time while a request packet is passing through the node as it traverses the ring.

4. Hector Multiprocessor

A hierarchically-structured shared-memory multiprocessor architecture, called Hector, has recently been developed at the University of Toronto [2][3]. Its structure is illustrated in Figure 2. A small number of processing modules (PMs), comprising a processor, cache memory and local memory, are connected by means of a bus to form a *station*. The choice of the bus at this lowest level of hierarchy is motivated by the simplicity and efficiency of bus interconnection when few modules are to be connected. The number of PMs that may be included in a station is limited to eight, to avoid the loading problems on the bus. At higher levels of hierarchy, Hector uses bit-parallel rings. Stations are connected to *local* rings, which in turn are connected to higher-level rings. Figure 2 shows only three levels of hierarchy, hence there is only one higher-level ring, which is called the *central* ring.

Hector provides a flat physical address space, where the memory in each PM is assigned a unique contiguous portion of the address space. A processor accesses all memory in the same way. As far as the processor is concerned, the only difference between local-memory and

remote-memory accesses is the time taken to complete the access. Hence, Hector has all the characteristics of a NUMA (Non-Uniform Memory Access) machine. Information transfer in Hector takes place in the form of fixed-sized packets. The transfer is synchronous, such that a packet moves from one ring node to another in one clock cycle.

The communication protocol in Hector is of request-response type, where some form of a response is always expected. Requests and responses are performed by transmitting appropriate packets between the PMs involved in a particular transaction. Both request and response packets are allowed to fail, in which case retransmission takes place. In case of a `read` request, the destination PM normally returns the data as its response. However, if the destination is busy (servicing other requests), it ignores the request and sends a negative acknowledgement packet to the source. In case of a `write` request, the destination writes the data included in the request packet and returns a positive acknowledgement to the source. If the destination is busy, it returns a negative acknowledgement packet. The source will retry its request if it receives a negative acknowledgement or if a time-out period elapses before an acknowledgement is received.

A protocol of this type would have a problem with `read-modify-write` accesses, if the destination location is allowed to respond to another request before the read-modify-write access is successfully completed. In order to prevent this possibility, the read-modify-write access is performed as two connected transfers. First, the source PM issues a `read and lock` request, which reads the desired location and locks this location to prevent it from being read by some other processor. Upon receiving the read data, the source PM sends a `write and unlock` packet which writes the new contents and frees the locked location to respond to further access requests.

The request-response protocol where either request or response packets may fail to cause the desired actions allows simple interface circuits to be used. The obvious drawback is that whenever a packet fails, it is necessary to try again. The number of retries should not be large, because they waste the bandwidth of the communications network and increase the latency of accesses. This number can be kept small be inserting buffer queues in the interfaces of PMs. The Hector prototype machine has only one level of buffers for this purpose. In future machines it is advisable to increase the size of the buffers. An optimal size may not depend only upon the desired reduction in the retry rate, but also on the requirements of block-transfer traffic if such transfers are supported.

A key advantage of Hector architecture is its hierarchical structure and the natural ordering of packet transfer that the ring-based communications system provides. The advantage due to the hierarchy is exploited best if the operating system places processes and pages such that most of the necessary memory accesses are within a cluster at a given layer of the hierarchy, for example within a single station or a single local ring. The Hurricane operating system [4], which has been developed for the Hector multiprocessor, migrates pages with no shared data and replicates read-only pages to the stations where they are being accessed.

The natural ordering in packet transfer provides convenient support for implementation of cache consistency. While the Hector prototype system implements the cache consistency by software means, the architecture allows realization of an efficient hardware-based cache consistency scheme as described in the next section.

5. Cache Consistency

A primary requirement for a hardware-based cache consistency scheme is that it be scalable to large configurations. Simple schemes can be devised if there exists a natural broadcast mechanism that does not affect the backplane communication traffic adversely. Unfortunately, such schemes do not scale well. In small bus-based multiprocessors it is easy to achieve cache consistency using a snooping protocol, but buses are unsuitable for larger machines. Broadcasting capability can be provided with many other types of backplanes, but the traffic caused by the cache consistency protocol is likely to use much of the available bandwidth.

A recently proposed scheme for Hector-like machines [5][6] shows how the hierarchical ring structure can be used effectively. It exploits the

simplicity of broadcasting, but curbs the amount of broadcast traffic by means of a filtering mechanism. Consider first the case without filters, where full broadcast is used. A packet that is to be broadcast is sent to the highest ring in the hierarchy. It circulates once around this ring. As it traverses the ring, at each interface to the next lower-level ring a copy of the packet is made and circulated along the lower-level ring. This is repeated at all interfaces to lower levels until the copies reach the stations which are the bottom level. A copy that is circulating around a ring is removed by the node at which it was created. When a broadcast packet reaches a station, it is switched onto the station bus to allow snooping by the caches on this station.

To illustrate how this broadcast scheme may be used, consider the case where the protocol enforces sequential consistency for write-through invalidating caches. Using this protocol, when a processor changes a cached shared data item the change is written through to the main memory and all other cached copies of this data are invalidated. This can be achieved using the following procedure:

1. The processor sends a write packet to the target memory, and is then blocked.
2. The target memory location is updated, and is then locked.
3. The target memory sends a request for cache invalidation broadcast to the highest-level ring.
4. A "write invalidate" (WI) packet is broadcast through the multiprocessor using the broadcast mechanism described above.
5. Upon receipt of the WI packet, each affected cache invalidates the corresponding cache block; also, the target memory is unlocked and the originating processor is unblocked.

This protocol requires a guarantee that the necessary ordering of shared accesses is preserved, which demands that:

1. there exists a unique path between any two modules,
2. two packets cannot overtake each other, and
3. packets are processed at each node in the order in which they arrive.

These demands are satisfied naturally by the hierarchical ring structure.

In order to reduce the amount of traffic generated by a full broadcast, it is possible to implement a filtering mechanism that suppresses the propagation of invalidation packets to those parts of the system that do not contain cached copies of the data in question. An attractive scheme for doing this makes use of two types of filters [5][6]. Incoming filters suppress the propagation of WI packets to the next lower level of the hierarchy if this part of the system does not contain any copies of the data being invalidated. If all copies of the data that must be invalidated lie in a portion of the system that can be reached without going to the top of the hierarchy, then it is advantageous to initiate the WI broadcast at the lowest-level ring from which all affected copies can be reached. Outgoing filters are used for this purpose, which suppress the propagation of the "request cache invalidation" packet to higher levels. This filtering scheme can be implemented by including a "bit-mask" in invalidate packets, which indicates the portions of the hierarchy that are affected. The scheme proposed in [6] uses a very compact bit-mask, which consists of one field per each level of the hierarchy and a bit in this field is set to 1 for each path to the next lower-level ring (or station) that the packet should reach. A trade-off for using the compact bit-masks is that some unnecessary invalidation traffic passes through the incoming filters. This is a small price to pay for the simplicity and cost effectiveness of the filtering mechanism. The cost of bit-masks, which are kept as an additional part of each memory module, is less than in existing directory-based cache consistency schemes because less specific state information (location of cached copies) is used for filtered broadcasts.

In order to allow high-speed transfer in the communications network, it is essential that the delay caused by the filtering mechanism at each ring or station interface is minimal. This requirement is met by the above protocol, because the routing decision for an incoming filer

is made simply by observing the state of a single bit, while the outgoing filter prevents propagation to the next higher-level ring only if the bit-mask field that corresponds to this level of hierarchy has all bits equal to zero. This simplicity of routing, coupled with the relatively low cost of bit-mask storage, allows easy scalability of the cache consistency scheme to larger configurations.

6. Block Transfers

In a NUMA multiprocessor, the operating system may often have to transfer large blocks of data from one memory unit to another. Such transfers can be performed much more efficiently if the machine has an inherent mechanism for easy block transfer. In ring-based multiprocessors, the provision of this facility is influenced significantly by the type of communications protocol used.

The three well known protocols that have been used with ring networks in general are: token, slotted, and register insertion protocols. They may all be considered as viable candidates for use in ring-based multiprocessors. The token protocol provides naturally the flexibility needed for block transfers, because a node in possession of the token may perform a contiguous transfer of an entire block. The maximum size of the blocks is restricted only by the size of the buffers in the interfaces of sending and receiving nodes. While this mode of control is very efficient for block transfers, it increases the latency of non-block transfers that usually involve only a small amount of data. The slotted protocol favors the transfer of short messages, where one or two slots suffice for the complete transfer. With this control mechanism, it is necessary to use many slots (not necessarily contiguous) to transfer a large block. The third possibility is to use register insertion. This is the least attractive alternative, due to the added complexity of interfaces and problems in providing a recovery mechanism when a fault occurs.

A recent study [7] has shown that the best overall performance can probably be achieved by a combination of slotted and token protocols. However, the performance of a simple slotted protocol is almost as good. Since this protocol is easier to implement, the slotted ring is the best practical choice.

Assuming that the slotted protocol is used, which favors transfer of short messages, it is useful to include additional hardware to support block transfers. Considerable gains can be achieved if a single read request packet causes a subsequent transfer of an entire block, rather than having to send a separate read request packet for each piece of data that is to be returned in one slot. Such hardware-supported block transfers can improve the performance of a multiprocessor by as much as 50% [8].

7. Scalable Coherent Interface

A notable development in the work on multiprocessor systems is the recently defined SCI standard, known as the IEEE standard project P1596 [9]. The SCI standard defines a multiprocessor backplane that is supposed to provide: fast signalling, scalable architecture, cache coherence, and simplicity of implementation.

It is interesting to observe that the working group that defined the SCI standard began by trying to increase the bandwidth of a backplane bus, but eventually abandoned this approach as being unrealistic. Instead, a scheme that features point-to-point connections has been adopted. Thus, all of the advantages of point-to-point links can be exploited. Yet, in some respects, the SCI scheme simulates a bus-style operation without actually using a bus.

Figure 3 shows a simplified model of a SCI node. External connections come from input and output links. While these links do not have to be arranged in any particular topology, it is obvious that the ring topology is one of the most natural choices. A bypass FIFO is included to buffer packets that arrive while the node is transmitting its own packet. The size of this FIFO can be minimized if the protocol allows the node to start transmission of its own packet only when the FIFO is empty. Then, the FIFO has to be as large as the longest packet that the node can originate. If the node is not sending its own packet, the bypass FIFO need not be included in the transmission path, which is achieved by means of the output multiplexer.

Since there is a provision for transmission at very high clock rates, it is unlikely that the

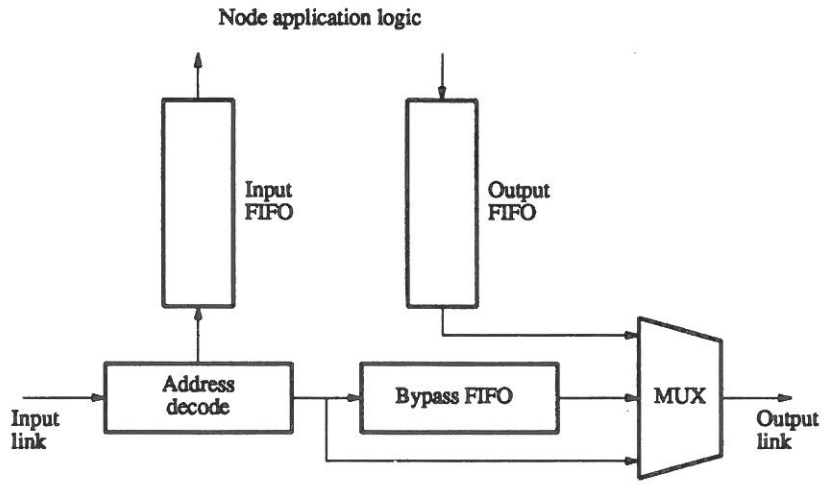

Fig. 3. SCI node model

node application logic can handle the data at such rates. This necessitates inclusion of additional buffers. Packets addressed to the node are buffered in an input FIFO, while packets that have been prepared for transmission are placed in an output FIFO.

Two types of signalling methods can be used. One possibility is to employ ECL differential signalling, involving parallel transmission of 16 data bits and 2 control bits. Using both edges of a 250 MHz clock, this allows a transmission rate of 1 GByte/s. The second possibility is to use a fiber-optic transmission medium. This scheme employs bit-serial links and is intended for longer links (from tens to thousands of meters).

SCI is described in terms of a distributed shared-memory model with cache coherence. This is the most complex service that it can provide. However, it can also be used in a message passing system or in a system where hardware supported cache coherence is not needed. It is even possible to dynamically mix the transactions of all of the above types in one system.

Most of the architectural definitions conform to the IEEE 1212 standard that defines Common Control and Status Registers architecture. A 64-bit architecture is assumed, where an address consists of 16 bits that specify one of 64K nodes and 48 bits define the addressable space within a node.

Transmission protocol is based on a single-requestor single-responder principle, where a packet originates at a source and is addressed to a single target. If a "send" packet sent by the source is accepted by the target, the latter returns a "done-echo" packet. If the "send" packet is not accepted, the target returns a "retry-echo" packet. Packets that are not accepted by the target are retransmitted by the source. An acceptance protocol based on aging ensures that a node will eventually have its retransmitted send packet accepted by the target. It is also necessary to guarantee that all nodes can transmit their send packets. This is achieved using an allocation protocol based on increasing priority of delayed packets in output queues. The transmission protocol has a critical requirement that echo packets cannot be dropped. Thus, the source of a send packet must preallocate space for the expected echo packet, before it is allowed to transmit its send packet.

A large SCI system may consist of a number of rings. In such a system, remote transactions between different rings pass through "intermediate agents", namely bridges or switches. A remote transaction is broken down into a series of requestor-responder transactions between the requestor and the agent on one side, and the agent and the responder on the other side.

Cache coherence in SCI is achieved by means of a distributed directory-based protocol. A doubly-linked list is established for each cache line containing shared data. Each processor node that caches a given line of shared data includes pointers (as part of the cache-line tag)

to the previous and the next nodes that share the line. The head of this list has a pointer to the memory node that holds the line. When a new node accesses the memory node to read this line, the node becomes the new head of the list and the memory directory is updated by replacing the pointer to the previous head with the address of the new head. A write access to the memory can be done only by the head of the list. If any other node wishes to obtain a write access, it may do so by inserting itself at the head of the list and purging the rest of the entries in the list.

The attractive feature of the SCI cache coherence scheme is that it scales well, because the memory directory and the processor cache-tag storage requirements do not increase as the size of the linked list increases. But, this additional storage presents a large fixed overhead that must be paid for in all cases.

The SCI may provide a unified approach for building future multiprocessors. However, its practical viability cannot be assessed until integrated circuits become available that implement the full protocol and some prototype systems are built.

8. Concluding Remarks

Perhaps the most attractive advantages of ring-based interconnection networks are the simplicity of interface circuitry and the speed of transfers that can be achieved. However, it is important not to overlook the broadcast capability where a message visits all nodes in a well defined race-free order. This capability provides nicely the basis of the cache consistency mechanism.

Ring interconnection lends itself well to implementation of hierarchically structured interconnection networks. Such networks decrease the latency of transfers and use the bandwidth efficiently, particularly when a large number of transfers span only a small portion of the hierarchy.

Finally, hierarchically-structured ring-based multiprocessors tend to be highly scalable, if there is a cache consistency mechanism provided that does not rely on full broadcasting.

References

[1] KENDAL SQUARE RESEARCH, *KSR-1 Principles of Operations*, Waltham, Ma., 1991.

[2] Z. VRANESIC, M. STUMM, D. LEWIS AND R. WHITE, Hector: A hierarchically Structured Shared-Memory Multiprocessor, *IEEE Computer*, Vol. 24, No. 1, pp. 72–79, Jan. 1991.

[3] M. STUMM, Z. G. VRANESIC, K. FARKAS AND R. UNRAU, Experiences with the Hector Multiprocessor, *Proc. Int. Parallel Processing Symp.*, Newport Beach, Ca., April 1993.

[4] M. STUMM, R. UNRAU AND O. KRIEGER, Designing a scalable operating system for shared memory multiprocessors, *Proc. Usenix Workshop on Micro-kernels,* April 1992.

[5] K. FARKAS, Z. VRANESIC AND M. STUMM, Cache Consistency in Hierarchical Ring-based Multiprocessors, *Proc. Supercomputing 92*, Minneapolis, Mn., pp. 348–357, Nov. 1992.

[6] K. FARKAS, Z. VRANESIC AND M. STUMM, Scalable Cache Consistency for Hierarchically-Structured Multiprocessors, in press.

[7] S. J. WILTON, Block transfers in a shared memory multiprocessor, Master's thesis, University of Toronto, August 1992.

[8] S. J. WILTON AND Z. G. VRANESIC, Architectural Support for Block Transfers in a Shared Memory Multiprocessor, *Proc. Fifth IEEE Symp. on Parallel and Distributed Processing*, Dallas, Tx., Dec. 1993.

[9] D. GUSTAVSON, The Scalable Coherent Interface and Related Standards Projects, *IEEE Micro*, pp. 10–22, Jan. 1992.

Received: September, 1993
Accepted: December, 1993

Contact address:

Prof. dr. Zvonko Vranesic
Dept. of Electrical and Computer Engineering
University of Toronto, Canada
e-mail: zvonko@eecg.toronto.edu

ZVONKO VRANEŠIĆ received the B.A.Sc., M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Canada, in 1963, 1966 and 1968, respectively. From 1963 to 1965 he worked as a design engineer for Nothern Electric Co. Ltd., Bramalea, Ontario, Canada. In 1968 he joined the faculty of the Departments of Electrical Engineering and Computer Science at the University of Toronto, where he is now a Professor. During the academic years 1977/78 and 1984/85 he was a Senior Visitor in the Computer Laboratory at the University of Cambridge, England, and in the Institut de Programmation at the University of Paris 6, France.

His research interests include computer architecture, VLSI systems, fault tolerant computing, local area networks and many-valued switching systems.

He is a member of the Association of Professional Engineers of Ontario and a Senior Member of IEEE. He was the Chairman of the 3rd International Symposium on Multiple-Valued Logic in 1973 and of the 18th International Symposium on Computer Archictecture in 1991.