JOURNAL OF ELECTRICAL ENGINEERING AND INFORMATION SCIENCE, VOL 2, NO. 3, 1997

Design of a Parallel Computer Network Interface Controller

Sung-Gu Lee

Abstract

This paper describes the design of a network interface controller (NIC) chip which is to be used to support a novel adaptive virtual cut-through routing method for parallel computer systems with direct (i.e., point-to-point) interconnection networks. The NIC chip is designed to provide the interface between a processing node constructed from commercially available microprocessors and another custom-designed router chip, which in turn performs the actual routing of packets to their respective destinations. The NIC, designed using a semi-full-custom VLSI design technique, implements a new adaptive virtual cut-through routing method (presented in an earlier paper) which has been shown to significantly outperform traditional wormhole routing with a minimal amount of hardware overhead. The NIC design has been fully simulated and laid out using a 0.8 μm CMOS process.

I. Introduction

Computer technology has currently evolved to the point where almost all commercial computers above the microcomputer category use parallel computing in order to achieve high performance. The interconnection network is a critical component in this type of parallel computer system because information must be communicated quickly in order for the processing nodes of the parallel computer to cooperate in solving a given problem. An interconnection network is characterized by its topology, routing, and flow control. Information may be sent as messages between the processing nodes of a distributed-memory multicomputer[1] or as data between the processors and memories of a shared-memory multiprocessor[2].

The first-generation multicomputers of the mid-1980's (NCUBE/ten, iPSC, etc.) used "store-and-forward type" packet switching methods borrowed from the computer network field in order to perform the necessary message passing. However, since this communication was typically handled by software in the processor responsible for the computations, inter-processor communication was extremely slow (on the order of a few milliseconds per hop) and was a major limiting factor in the scalability of parallel programs.

Realizing this deficiency, the second-generation multicomputers (NCUBE2, iPSC/860, etc) turned to hardware-supported switching methods for fast inter-processor communication (on the order of microseconds). In this type of method, a dedicated hardware device, *termed a router*, is used to perform flow control without the intervention of the computation processor. All major hardware-supported routing methods can be considered as variants of two popular routing methods termed *wormhole routing* (WR)[3] and *virtual cut-through* (VCT)}[4].

This paper describes the design of a network interface controller (NIC) chip which supports the implementation of a novel adaptive VCT routing method. The NIC resides at the interface between the processing nodes and the interconnection chip of the parallel computer. The rest of this paper is organized as follows. Section II presents background concepts. Section III briefly describes the novel adaptive VCT routing method developed by this researcher. Section IV describes the design of the network interface controller chip. The paper concludes with Section V.

II. Background

The type of parallel computer assumed will be a multicomputer with a *direct interconnection network*, in which a communication link can only be used by the two end-nodes that it is connected to. Examples of direct interconnection networks include 2-dimensional meshes, tori, hypercubes, cube-connected cycles, etc. A *k-ary n-cube* is a generalization of 2-dimensional meshes, tori, and hypercubes in which there are n dimensions and k nodes in a single dimension. K-ary n-cubes can be constructed as torus or mesh networks, where the nodes along one dimension are connected in a cycle in torus networks and in a linear array in mesh networks.

Manuscript received March 24, 1997; accepted May 23, 1997.

Lee is with Department of Electronic and Electrical Engineering Pohang University of Science and Technology(POSTECH), Korea.

Adjacent processing nodes are assumed to be connected by two directed communication links in opposite directions (although undirected links are also possible, directed links are assumed for simplicity). A communication link consists of one or more data lines and several control lines. One or more *physical channels* may occupy a single communication link by using time multiplexing or by partitioning the set of data lines.

One or more *virtual channels* may occupy a single physical channel by using separate flit buffers and control lines for each virtual channel. When two or more *virtual channels* are active over a single physical channel, they must be time-multiplexed; however, when only one of the virtual channels is active, it can utilize the full bandwidth of the physical channel. For simplicity of discussion, unless otherwise stated, it will henceforth be assumed that there is one physical channel per communication link and one virtual channel per physical channel.

Two popular hardware-supported switching methods are *WR* [3] and *VCT*[4]. In WR and VCT, messages arriving at an intermediate node are immediately forwarded to the next node on the path without buffering, provided that a channel to the next node is available. This is referred to as a *cut-through* operation. A cut-through may be performed at an intermediate node as soon as the header of the message arrives with the destination information. Thus, with cut-through, only an on-line *flit*[1] buffer is required to examine the message header at each intermediate node. Thus, when the message has cut-through several nodes, the flits of the message become spread out in the flit buffers along the path used.

If cut-throughs are established through all intermediate nodes, a *circuit* is established to the destination, and the switching method resembles traditional *circuit switching*. If the message header is *blocked* at an intermediate node because the requested outgoing channel is unavailable, the message is kept in the network (i.e., in the on-line flit buffers at each node along the path up to the current node) in WR and completely buffered at the current node in VCT. Since sufficient buffer storage must be available for any message that reaches a node, VCT typically requires packetization of messages and a large partitionable buffer at each node. WR, on the other hand, contributes to network congestion when a message becomes blocked. In the rest of this paper, it will be assumed that messages are sent in fixed-size packets.

WR is the most popular communication method used in current commercial multicomputers due to its low hardware overhead (requiring only a single flit buffer per incoming channel) and high performance. However, VCT can typically achieve higher performance than WR for the following reason: when a packet cannot cut-through an intermediate node, the packet is buffered at that node instead of being kept in the network as in WR. This effect becomes particularly evident with heavy network traffic. The main reason that VCT has not been as popular as WR is that the required buffering capability of the former method has been seen as a large hardware overhead.

Due to its relative unpopularity, VCT has not been as heavily researched as WR. However, this researcher has previously presented a novel adaptive VCT routing method that has significant performance benefits over WR and can be implemented with minimal hardware overhead[5]. Since this paper concerns the design of a network interface controller chip which is designed to support this new adaptive VCT method, the next section is devoted to a brief overview of the new adaptive VCT routing method.

The use of virtual channels was introduced by Dally and Seitz [3] as a means of guaranteeing *deadlock-free* WR. The WR algorithms introduced in [3] are *deterministic* algorithms, in which the routing path used is completely determined by the source and destination node addresses. In[6], Dally showed that virtual channels could also be used with a deterministic WR algorithm to significantly reduce the probability of blocking, and thus increase the throughput of the network.

Realizing the need for adaptability in routing around congested or faulty nodes and links, several researchers[7, 8] used virtual channels to produce *adaptive* deadlock-free WR algorithms, in which a packet is permitted to change its routing path if a blocked channel or faulty node/link is encountered. Adaptive algorithms can be classified into *minimal*[7, 9] and *nonminimal*[8, 10] algorithms, where a minimal algorithm always uses shortest-length paths to each destination, and *fully adaptive*[7, 8] and *partially adaptive* [9, 10] algorithms, where a partially adaptive algorithm differs from a fully adaptive algorithm in that only a subset of the possible paths may be used. Some adaptive algorithms use one *virtual network* (a set of virtual channels) to route adaptively and another *virtual network* to route deterministically in order to escape possible deadlock situations[8, 11]. In all of these WR methods, even though several physical or virtual channels may be unoccupied on minimal paths to the destination, only a subset of the unoccupied channels may be used to route any given packet (the remaining physical or virtual channels are necessary to guarantee freedom from deadlock).

III. Adaptive Virtual Cut-through

This section provides a brief overview of the novel adaptive VCT previously developed by this researcher. For a more comprehensive presentation, the interested reader is referred to[5].

1. Motivation

WR methods have several deficiencies when compared to VCT. First, WR in general has a lower saturation throughput than a comparable VCT method as blocked packets contribute to network

1) A *flit* as defined in [3] is the smallest unit of information that a node may refuse to accept. Thus, in a simple implementation, the number of bits in a flit may correspond to the number of data lines in a single physical channel.

congestion in WR; however, this problem is significantly alleviated by the use of multiple-flit flit buffers and/or multiple virtual channels. Second, even with the use of virtual channels, 100% of the free channels (and flit-buffers) are not available for use by a given packet as some of the channels have to be reserved in order to prevent deadlock. Third, fully adaptive routing in torus networks is difficult and requires a large proportion of the virtual channels to be used for deadlock avoidance. Although virtual channels do not waste channel bandwidth, virtual channels do occupy valuable hardware resources because each virtual channel requires a separate on-line flit buffer and control line and contributes to the complexity of the arbitration logic within the router.

For the above reasons, adaptive VCT is proposed as an alternative to WR. When originally proposed in[4], VCT was described as a deterministic routing algorithm. In deterministic VCT, when the head of a packet is received at an incoming channel, the header is decoded to determine the "one" outgoing channel to connect to. If that outgoing channel is busy, then the packet is buffered at the current node. This method is analogous to deterministic WR as originally described by Dally and Seitz[3]. Thus, it is natural to consider an adaptive version of virtual cut-through in order to be able to navigate around congested or faulty nodes/links.

In order to facilitate discussion of the new adaptive VCT method, a general switching model, shown in Fig. 1, is proposed. When the head of a packet is received at an incoming channel, the destination node information is extracted from the header. A connection is then attempted to the first-choice outgoing channel. If successful, then a cut-through is established through the current node as denoted by (0) in Fig. 1. However, if the requested outgoing channel is blocked (1) because it is being used by another packet, then a connection is attempted to the second-choice outgoing channel (2). This process is repeated until a cut-through is achieved or all outgoing channels permitted by the adaptive VCT algorithm are exhausted, as is the situation in Fig. 1. In this situation, the blocked packet remains in the network (as in WR) until one of the permissible outgoing channels becomes available or the "permitted waiting time" expires. In the latter case, the packet is received at the current node (3) and inserted into the source buffer of the current node just as if the packet originated there (the circuit implementation is discussed in Section IV). The packet at the head of the source buffer waits until one of the outgoing channels permitted to it becomes available, and then exits through that outgoing channel, as shown by (4) in Fig. 1.

Several variations of adaptive VCT are possible depending on the degree of adaptivity permitted, the use of minimal or non-minimal routing, and the permitted waiting time before buffering at intermediate nodes. However, detailed simulation studies[5] showed that a fully adaptive minimal-path adaptive VCT algorithm denoted as the A_{ms} algorithm provides the best performance at the lowest cost. Fig. 2. A_{ms} shows the operation of the A_{ms} algorithm and a similar algorithm referred to as A_{mc}. The

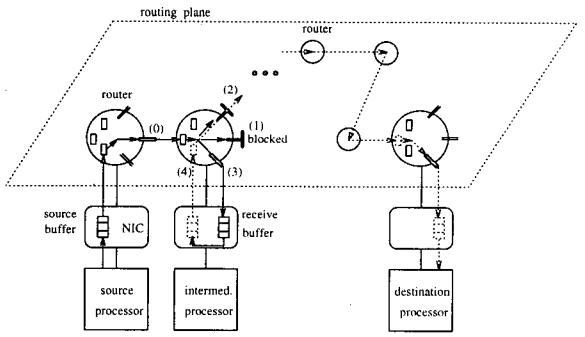

Fig. 1. A general switching model for adaptive VCT.

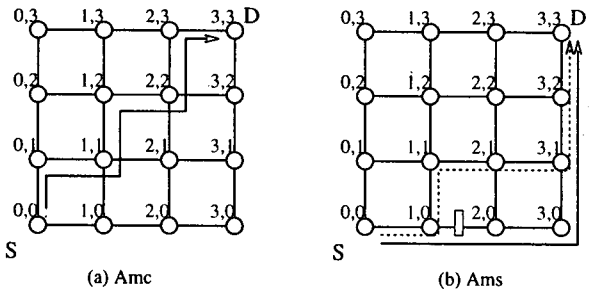

Fig. 2. Fully-adaptice minimal algorithms (a) A_{mc} and (b) A_{ms}.

A_{mc} algorithm tends to follow a "zig-zag" pattern in order to maximize the probability of being able to route around congested or faulty nodes/links. However, since this involves a difficult control algorithm at the routers, a simpler "priority-X-direction" pattern was used in the A_{ms} algorithm. In this algorithm, the X-direction is traversed before the Y-direction, except if a blockage occurs as in the link ((1,0),(2,0)), in which case the blocked link is bypassed while maintaining the minimal-path property.

Simulation studies[5] showed that the A_{ms} algorithm outperformed the alternative adaptive VCT algorithms and the various possible WR algorithms in the literature. Fig. 3 shows the simulation results for the A_{ms} algorithm, deterministic WR[3] (XY_WR), an adaptive WR algorithm based on the turn model[10] (Negative_first), and the $R_{7,0}$ adaptive WR algorithm presented in[9] (Zenith_routing).

Ⅳ. Network Interface Controller Design

1. Communication Architecture

The proposed interconnection network communication architecture for supporting adaptive VCT is best explained on the basis of the communication architecture of the Intel Paragon, shown in Fig. 4. As can be seen from this figure, there is a network interface controller (NIC), source and receive buffers, and a dedicated communication processor (a type of I/O processor) responsible for preparing and sending the packets out onto the network. Packets are initially stored in the source buffer before being sent out to

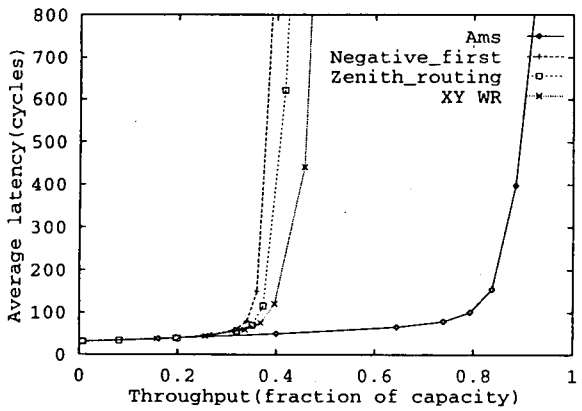

Fig. 3. Average latency comparison with deterministic and adaptive WR algorithms given a 16×16mesh, 20-flit, a uniform destination node distribution, and 32-flit buffers.

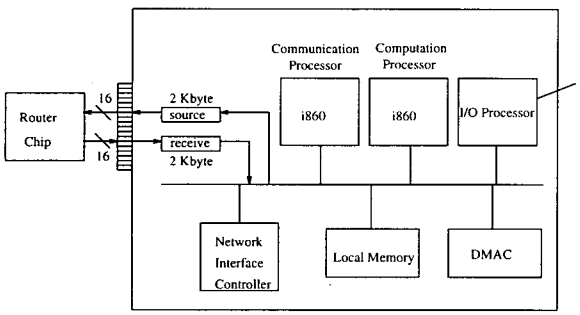

Fig. 4. The network interface for the Intel Paragon.

the router chip. The router chip is connected to other router chips in a connection pattern determined by the interconnection network topology used. A packet which is injected into a router chip from a processing node is routed to the router chip connected to the destination node. Packets to be received by the current node are initially buffered in the receive buffer before being transferred to local memory. In the Intel Paragon, the source and receive buffers are 2Kbyte buffers and the links to the router are 16-bits wide.

To demonstrate the feasibility of adaptive VCT, the A_{ms} algorithm was implemented in a semi-custom-designed VLSI router chip and NIC chip. The complete hardware implementation of the A_{ms} algorithm in a multicomputer would involve using our router and NIC chips in the network interface and adding a small modification to the source buffer. Upon receiving a new packet in the receive buffer, the NIC must examine the header to determine if the packet is destined for the current node or "in transit". In the latter case, the packet must be rerouted to the source buffer to be transmitted when one of its permitted outgoing channels becomes available. (Note that bandwith matching requirements imply that the data transfer rates in the source and receive buffers must match the data transfer rates within the router network.) In order to perform this rerouting efficiently without having to contend for the local bus, a dedicated path must be established to the source buffer - arbitration with packet insertion from the local processing node (on a FCFS basis) is of course necessary.

The router chip to support A_{ms} adaptive VCT routing has been described in[5]. As shown in[5], the router design necessary to support the A_{ms} algorithm has negligible overhead when compared to the design for a deterministic WR chip. This is because the buffering requirement has been moved from the router to the NIC. However, the description of the NIC design in the next subsection will show that the NIC can also be efficiently designed to support the A_{ms} adaptive VCT routing algorithm.

2. *Design Description*

Fig. 5 shows a block diagram of our NIC (network interface controller) chip, which is designed to interface to our router chip and support the A_{ms} adaptive VCT algorithm. This chip has been completely designed, laid out, and simulated using Mentor Graphics CAD software. The send queue is a queue is of "send packet requeusts". If an entry is detected at the head of the send queue, the sending control logic forms a packet header from the destination information and sends it out followed by the packet body. The packet body is sent out from the source buffer starting at the address pointed to by "data address". The receiving control logic, upon receiving the header of a packet, determines if the packet is destined for the current node or is simply being stored temporarily. In the former case, the packet is stored in the receive buffer. In the latter case, the received packet is stored in the source buffer and an entry is appended to the tail of the send queue. The only addition required to the basic NIC design to implement the A_{ms} adaptive VCT algorithm is a flit buffer and some extra control logic to send temporarily buffered messages to the source buffer.

In our NIC implementation, the source buffer and receive buffer are implemented as SRAMs with a possible maximum size

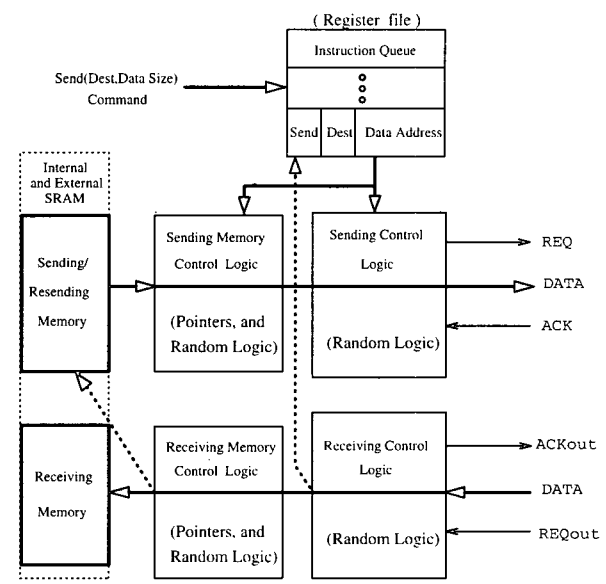

Fig. 5. Block diagram of the network interface controller chip.

of 32 Kwords (256 words in an internal SRAM and the rest in an external SRAM) and a word size of 6 bits (to interface to our 6-bit router chip). Logically, the source buffer is partitioned into two sections. In the first section, which we arbitrarily chose to be 2 Kwords, packets are stored in any "packet slot" as they are received. The second section is partitioned into K buffer classes to implement the "structured buffer pool technique" for deadlock avoidance[12]. The second section of source buffer memory is used only when the first section is completely filled. Our simulation results in[5] showed, however, that we almost never have to use this second section.

The NIC chip was designed using a 0.8 μm CMOS process and is currently awaiting fabrication. Fig. 6 shows the layout for the NIC chip. The NIC has been fully simulated at the gate level and at the transistor level using Mentor Graphics simulation tools. Fig. 7 shows part of the test waveform output for a typical receive operation.

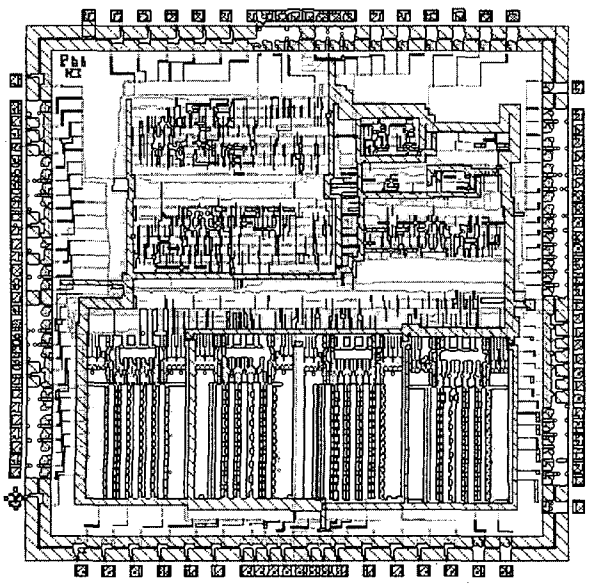

Fig. 6. Layout of the network interface controller chip.

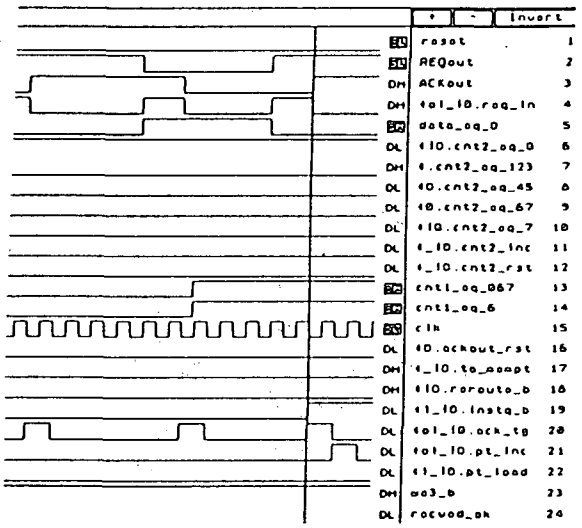

Fig. 7. Simulation test waveform output for a receive operation.

V. Conclusion

This paper has described the design of a network interface controller (NIC) chip which supports a novel adaptive VCT algorithm previously developed by this researcher in[5]. Although WR has been more popular in parallel computer interconnection networks due to its lower perceived hardware cost, an adaptive VCT algorithm with minimal hardware overhead and excellent performance characteristics was presend in[5].

In order to demonstrate the feasibility of the proposed adaptive VCT algorithm, labeled A_{ms} in[5], the prototype of the hardware necessary to support the A_{ms} algorithm should be developed. Thus, the router chip to support the A_{ms} algorithm was fully designed and tested[5]. In this paper, we showed that the network interface controller (NIC) chip required to support the A_{ms} algorithm can also be efficiently designed. The NIC chip has been completely laid out and simulated using Mentor Graphics CAD software and is currently awaiting fabrication.

References

[1] W. Athas and C. L. Seitz, "Multicomputers : Message-passing concurrent computers," *IEEE Computer*, pp. 9-23, Aug. 1988.

[2] L. N. Bhuyan and D. P. Agrawal, "Design and performance of generalized interconnection network," *IEEE Trans. Comput.*, Vol. C-32, pp. 1081-1090, Dec. 1983.

[3] W. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, Vol. C-36, pp. 547-553, May 1987.

[4] P. Kermani and L. Kleinrock, "Virtual cut-through : A new computer communication switching technique," *Computer Networks*, Vol. 3, pp. 267-286, 1979.

[5] H. S. Lee, H. W. Kim, J. Kim, and S. Lee, "Adaptive virtual cut-through as an alternative to wormhole routing," in *Proc. 24th Int'l Conf. on Parallel Processing*, (Oconowoc, WI), pp. 68-75, Aug. 1995.

[6] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel and Distributed Systems*, Vol. 3, pp. 194-205, Mar. 1992.

[7] D. H. Linder and J. C. Harden, "An adaptive fault-tolerant wormhole routing strategy for *k*-ary *n*-cube," *IEEE Trans. Comput.*, Vol. C-40, pp. 2-12, Jan. 1991.

[8] W. J. Dally and H. Aoki, "Deadlock-free adaptive routing in multicomputer networks using virtual channels," *IEEE Trans. Parallel and Distributed Systems*, Vol. 4, pp. 466-475, Apr. 1993.

[9] Y. M. Boura and C. R. Das, "A class of partially adaptive

routing algorithms for n-dimensional meshes," in *Proc. of Int'l Conf. on Parallel Processing*, Vol. III, pp. 175-182, Aug. 1993.

[10] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *Proc. 19th Int'l Symp. Computer Architecture*, pp. 278-287, 1992.

[11] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE Trans. Parallel and Distributed Systems*, Vol. 4, pp. 1320-1331, Dec. 1993.

[12] M. Gerla and L. Kleinrock, "Flow control : a comparative survey," *IEEE Trans. Comm.*, Vol. COM-28, pp. 553-574, Apr. 1980.

Sung-Gu Lee was received the B.S.E.E. with the highest distinction from the University of Kansas, Lawrence, in 1985 and the M.S.E. and Ph.D. from the University of Michigan, Ann Arbor, in 1987 and 1990, respectively. He is currently of an associate professor in the Department of Electrical Engineering at the Pohang University of Science and Technology(POSTECH), Pohang, Korea. His research interests are in parallel and fault-tolerant computing. Currently, his main research focus is on the high-level and low-level aspects of interprocessor communications for parallel computers.