# Considering Local Bus Traffic in Network Performance Simulations

**V. Jonnalagadda* M. Mathure* A. Kornecki** J. Zalewski*****
***School of EE&CS, University of Central Florida, Orlando, FL 32816**
**** Embry-Riddle Aeronautical University, Daytona Beach, FL 32114**
***** Computer Science, Florida Gulf Coast University, Ft. Myers, FL 33965**

**Keywords** Network performance, computer bus, VMEbus, RACEway, real-time networks, SES/workbench.

## Abstract

In a large distributed computing environment based on local area networks, such as Ethernet or FDDI ring, system performance may be significantly degraded by a bottleneck at some segment of a particular network. In this study we examine the effect that local bus traffic has on the overall performance of a network. In particular, we studied server access for networks using two interconnect technologies, VMEbus and RACEway. The results show that local bus traffic on the CSMA/CD or FDDI LAN backbones has a significant impact on overall performance and may decrease access delays by an order of magnitude, or more, depending on traffic load.

## INTRODUCTION

The proliferation of high-speed local area network (LAN) technologies has left users with a bewildering choice of networks and protocols. In a large-scale distributed computing environment forming a grid based on Ethernet or FDDI ring, system performance may be degraded by a network bottleneck.

One of the most demanding applications that current LANs support is server access. With the increase in processor speeds and the number of workstations attached to networks, network utilizations have been increasing with detrimental effects on the overall performance. It is difficult for users to determine in advance whether a given network technology will deliver the expected performance benefits, or to determine how to tune their existing systems to take full advantage of a high-speed network.

Complex networks involve individual nodes built out of multiple processors connected via some type of a bus or other interconnect, and with the nodes connected by some type of local network such as Ethernet. The processors within a node may or may not have local memory configurations and there may also be memory at the global level. Concentrating on performance of such networks brings us to the analysis of the most common LANs with local node traffic.

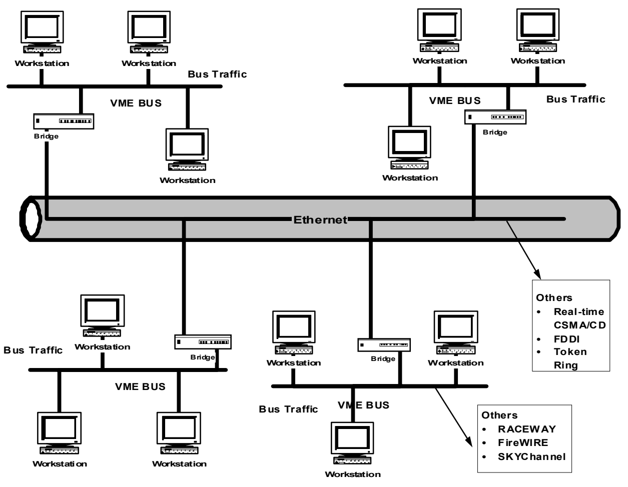

**Figure 1**. Generic Configuration of a LAN.

Figure 1 shows a generic configuration of a LAN with local nodes based on interconnects such as VMEbus or RACEway. Here we have a number of local hosts, workstations, or simply multiple processors housed in a single enclosure, connected via a bus, which use Ethernet to communicate with hosts located on other buses. Hence in addition to regular Ethernet traffic we also have bus traffic that has to be modeled and combined with the LAN traffic. We use the term workstation as equivalent to any local host or processor connected to its peers via a bus or other interconnect.

The purpose of this work is to study the migration of a multiple segment LAN, such as CSMA/CD or FDDI, to an extended LAN by introducing bus traffic on a VMEbus or RACEway as a local interconnect. We evaluate the performance of these network configurations under varying load and different topologies and compare the results to the performance of original CSMA/CD network. The main goal is to see how the local bus traffic affects the overall performance of the local area network (LAN). For this purpose, we estimate the server access time in network and bus models built with the SES/workbench simulator [1].

## MODELING BUS SYSTEMS

Most of the modern computer systems share a common bus, or other type of interconnect, to exchange information. There is a variety of popular bus standards: ranging from VMEbus to Futurebus to PCI, and modern interconnects, from Myrinet, to SKYchannel and RACEway, to InfiniBand, to name a few [2]. For most computer systems, the critical issue is the temporal determinism of system response. The timing of computations in the system depends not only on the instruction execution time and memory access, but also on the timing of data exchange between various system components. The latter, in addition to the hardware characteristics, is a function of the specific bus architecture and the bus access and arbitration protocols. In most cases, the system has a dedicated bus controller/arbiter – a device designated to manage the bus operation and execute specific bus protocol. Modern interconnects often use distributed arbitration, without a central arbiter node.

Queuing models are very adequate for this single-bus tightly coupled multiprocessor architecture [3]. Each processing element is modeled as a finite set of tasks (the task pool), a CPU, a bus interface unit that allows the processing element to access the shared bus, and a set of queues associated with the CPU and the bus interface unit. Each CPU and bus interface unit has a mean service rate and the tasks are assumed to have a mean sleep time in the task pool. Another assumption is that the CPU and the bus interface unit operate independently of each other, and that all the bus interface units in the multiprocessor can be lumped together into a single equivalent Bus Interface Unit.

A task is assumed initially asleep in the task pool. When it awakens, it will queue for CPU usage in the ready list, which is a list of tasks that are eligible to use the CPU as a result of either being awakened or as a result of having finished using some resource. Interrupt driven jobs are not placed on the ready list; instead they have their own higher priority queue. After using the CPU, the task may request more CPU service (in which case it will re-queue in the ready list), it may go back to sleep in the task pool, or it may request bus usage.

More advanced bus architectures can be modeled as a version of a queuing system with customers, servers, and resources. There are several studies in literature modeling the bus systems in such a way [4]. The models, mostly, focus either on distributed systems and describe networking applications or discuss multiple bus architectures and some real-time issues. When building the bus architecture model, we focus on the bus performance. The three basic metrics, identifying the performance of any bus system, are:

- system throughput – how much data can be transmitted per time unit,
- bus utilization – what percentage of time the bus is busy,
- bus access latency – the time needed by the bus to grant ownership to the requesting agent (a part of bus response time).

A high average system throughput is critical for most systems, for example, in various data acquisition/fusion applications. Related high utilization of the bus indicates that the margin for the system upgrade is limited. For real-time systems and distributed computing applications, more critical is the determination of system response, that is, bus access time, used as a performance measure in our study [5].

The above-mentioned criteria of bus performance are affected by the following factors:

- number of agents (processors) attached to the bus,
- latency of the memory,
- bus access protocol and the arbitration method used,
- physical characteristics of the bus (multiplexed lines, transmission speed, etc.),
- workload characteristics of the processor operation.

Selection of the workload, including the frequency of requests and the size of packets sent over the bus (often related to the local processor cache hit/miss ratio) is critical to the performance analysis study. The workload depends on the application and must be clearly identified before simulation model is developed.

### VMEbus Model

If one wants to enhance functionality and performance of the computer networks, by changing the configuration and introducing bus traffic to make them suitable for distributed computing applications, the VMEbus system [6] is a prime candidate because it is both rigorously defined and widely supported. The VMEbus provides support for multiprocessing using shared memory. Modules can be designed to act as masters, slaves or both. Before a master can transfer data it must first acquire the bus using a central arbiter (Fig. 2).

A master module can initiate data transfer cycles. A slave module detects bus cycles generated by a master module and participates in the cycle, if it is selected to do so. If certain addresses are selected by the master module, an on-board signal is sent by the local monitor and a message broadcast to all modules. To prevent a system crash, or other problem with data transfer, from locking up the system, a bus timer times all transfers and kills transfers which take longer than some specified time interval. Four of the most important cycles on the VMEbus are: read/write, block transfer, read-modify-write, and address-only.

To avoid inconsistency while updating shared memory, read-modify-write bus cycles are used. The read-modify-write cycle allows updating shared memory as an atomic transaction and prevents race conditions.

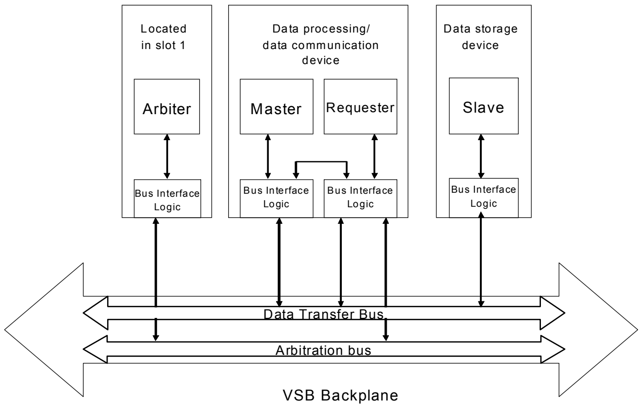

**Figure 2**. Typical VMEbus Architecture.

The system parameters that are considered in the VMEbus simulation model are as follows:

- number of processors in the system (optimally - 3 [5]),
- bus transmission speed, the time to send unit data through the bus (typically - 40 MB/s),
- arbitration time, the time required to select the next bus owner,
- transmission unit, the maximum amount of data transferred in a single transfer operation (256 bytes for VMEbus).

**RACEway Model**

Due to constantly increasing speeds of processors and peripheral devices, traditional buses are too slow to handle huge traffic in modern computer systems. One solution to this is a new interconnect called the RACEway [7], which is a hierarchical crossbar-based network. Unlike most multistage interconnection networks, the RACEway network provides more than one possible path between two system nodes (except that they are on the same crossbar switch element) and an adaptive routing capability.

Figure 3 shows the system level architecture of the RACEway implementation in a Mercury multicomputer system [8]. There are four types of system nodes: Processing Nodes, Memory-only Nodes, External I/O Interface Nodes, and Standard Bus Interface Nodes (i.e. VME). Each processing node contains a computing element (CE), an ASIC dedicated to the management of message traffic to and from the CE, and part of the main memory. CEs can transparently access remote memory locations in the system. Information transfer takes place by means of fixed-size packets known as RACEway packet (2048 bytes or less).

The basic data transfer element in the RACEway network, the RACEway crossbar switch, is a 6-port, point-to-point cross connection component. Each port has a 32-bit wide data path plus 5 control lines and has a data bandwidth of 160 MB/s. Each of the 6 ports on a given crossbar can be internally connected to one or more of any of the other 5 ports on that crossbar. These port-to-port connections are defined by headers in each data packet transmitted over the RACEway and can be changed dynamically in a switching time of 125 nanoseconds. Consequently, three independent port-to-port data transfer connections can be established simultaneously, providing an aggregate data bandwidth of up to 480 MB/s.

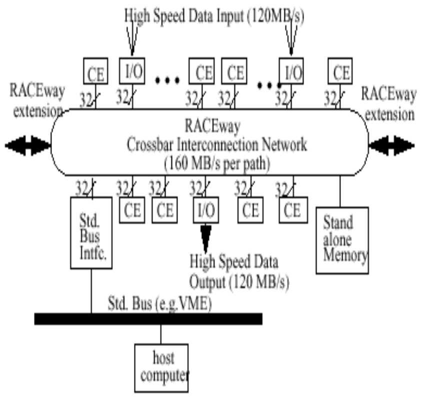

**Figure 3**. Typical RACEway Architecture [8].

The RACEway crossbar network supports concurrent, priority-based, preemptable data transfer between system nodes. The data transfer over RACEway network can be either CPU-controlled or DMA-controlled. Transactions across the network can be either Write or Read. The system supports each mechanism with different performance parameters such as path setup time, routing delay, etc. Each transaction has a user-defined priority. When two transactions require a common port in their paths, contention occurs. The crossbar makes the priority arbitration, the winning transaction goes through and the losing transaction is killed/blocked until the completion of the other one.

The system parameters that are considered in the RACEway simulation model are:

- number of processors in the system (6 per crossbar switch),
- bus transmission speed, the time to send unit data through the bus (typically – 480 MB/s),
- transmission unit, the maximum amount of data transferred in a single operation (2056 bytes).

## SIMULATION RESULTS

### VME Model Assumptions

In the VMEbus model in SES/workbench (Fig. 4), the active agents or processors make bus requests. The bus arbitration submodel is responsible for controlling and granting the bus resources to the agents in a timely and organized manner. Once a processor has control of the bus it performs the transmission. The size of data is the function of the workload.

The VME model assumes the following:

- each processor in the system has identical clock rate,
- each processor has identical bus request frequency,
- packets on VMEbus can be divided into smaller subpackets (not exceeding 256 bytes, determined by the VMEbus hardware characteristics),
- transmission of the data packet in progress is not interrupted,
- memory access time is assumed constant,
- a processor cannot be granted the bus if another request is presently waiting to be serviced by the bus, unless it has higher priority,
- errors due to invalid data or time-outs are negligible to the overall system performance.

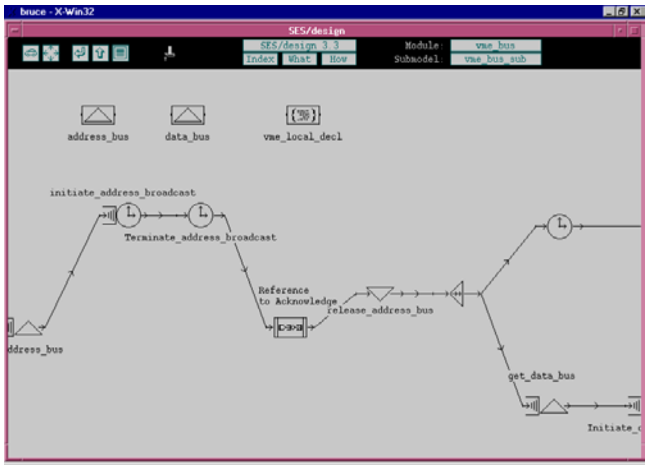

**Figure 4**. VMEbus Submodel in SES/workbench.

### Basic RACEway Model Assumptions

The RACEway multicomputer model in SES/workbench corresponds to the Mercury computer [8] and consists of three components: a CPU, a Communications Agent and a Crossbar (Xbar) Switch (Fig. 5). The CPU component models the function of the microprocessor in the Computing Element (CE). It provides only a "Compute, Send and Receive" type of functionality used in performance modeling. The communications agent models the functions of the CE's ASIC, specifically, the management of message traffic to and from the CE. It maintains the send and receive message queues, sets up and tears down the message path when messages are sent, handles message preemption, and handles the reception and response to remote read and write messages without involving the CPU.

The crossbar switch models the function of the RACEway 6-way crossbar in the RACEway interconnection network. This includes the modeling of normal setup and tear-down of communications paths for messages, adaptive routing, contention and priority arbitration, and path preemption. With this RACEway crossbar switch primitive, a variety of topologies of RACEway crossbar interconnection networks can be easily constructed by interconnecting the crossbar switches together.

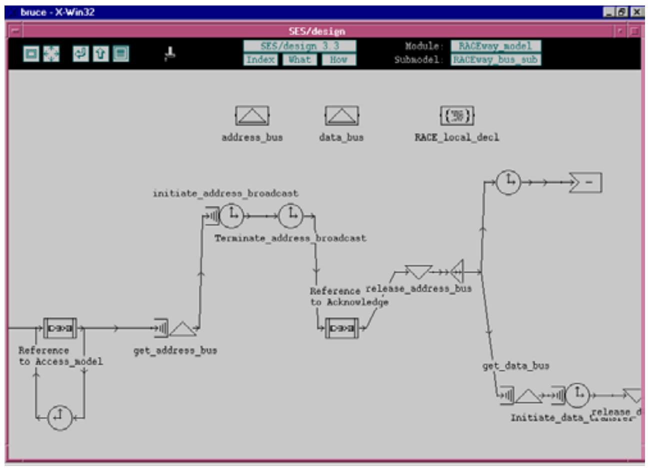

**Figure 5**. RACEway Submodel in SES/workbench.

### Server Access Delay

This case study compares RACEway and VMEbus with CSMA/CD segments on FDDI backbone [9]. The simulation models developed in this project are:

- FDDI backbone having four VMEbus segments.
- FDDI backbone with four RACEway segments.
- FDDI backbone interconnecting four CSMA/CD segments.

The workload parameters used in the VMEbus and RACEway models are as follows (for details, see [12]):

- inter-arrival time, an average time between two consecutive bus requests from the same processor (generated using a negative exponential distribution),
- packet size, the number of bytes to be sent in a request,
- protocol – priority based.

The first simulation, which is not bus-based but was developed for reference with more advanced systems, models an FDDI LAN connecting a four segment CSMA/CD LAN. Figure 6 shows the FDDI backbone and all the CSMA/CD bridges. The network consists of 70 to 85 active stations (workstations/clients) communicating with two server computers located on server segment #1.

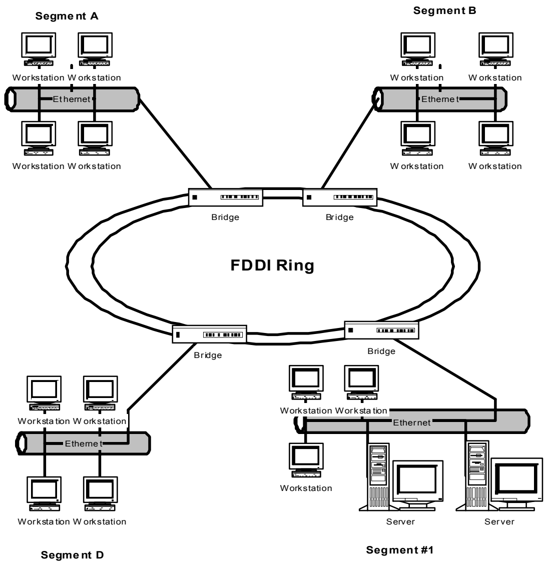

**Figure 6**. Configuration of the Server Network.

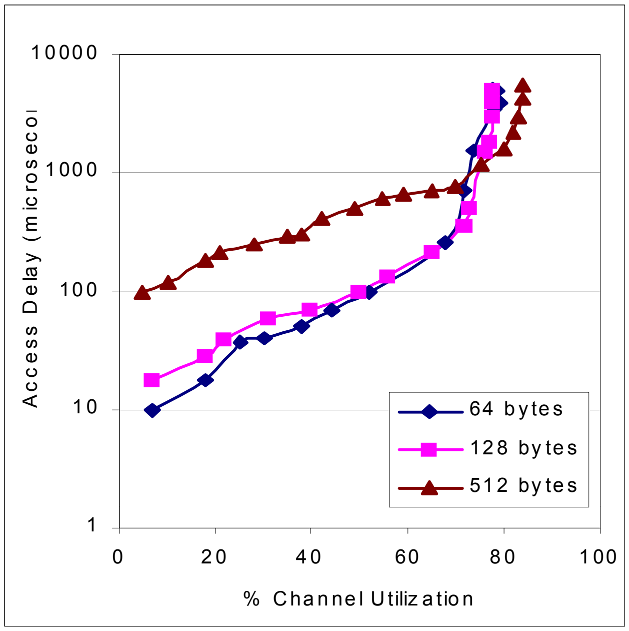

**Figure 7**. Server Access Delays for FDDI LAN with CSMA/CD Segments.

Figure 7 shows the results of increasing the load on segment #1, as access delay versus channel utilization for different frame sizes (64, 128 and 512 bytes). Access delay for shorter frames is, in general, shorter than for longer frames, however, only until about 76% utilization. The reason access delay becomes shorter for longer frames than for shorter frames is attributed to the increased number of total bits that must be transmitted with small packet sizes, due to a header/trailer overhead [10].

The second simulation models four VMEbus segments interconnected by an FDDI backbone. Comparing results from this and previous simulation shows that the migration to FDDI backbone with VME segments makes the access delay under lower loads 2-3 times smaller than when having CSMA/CD segments. The third simulation models four RACEway segments interconnected by an FDDI backbone. In this case, the access delay decreases 10 times compared to the VME-FDDI simulation, especially under high load conditions. The throughput of the network is also higher than in the four-segment VME on a FDDI backbone.

## Effect of Packet Length on Access Delay

Data for all studied configurations are shown collectively in Figures 8 and 9.

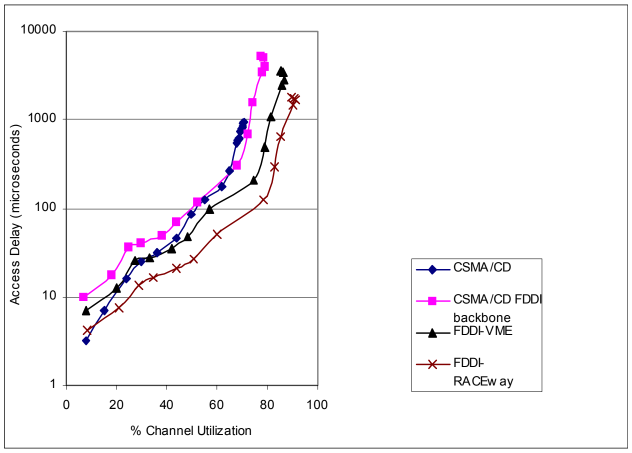

**Figure 8**. Server Access Delay for 64-byte Packets.

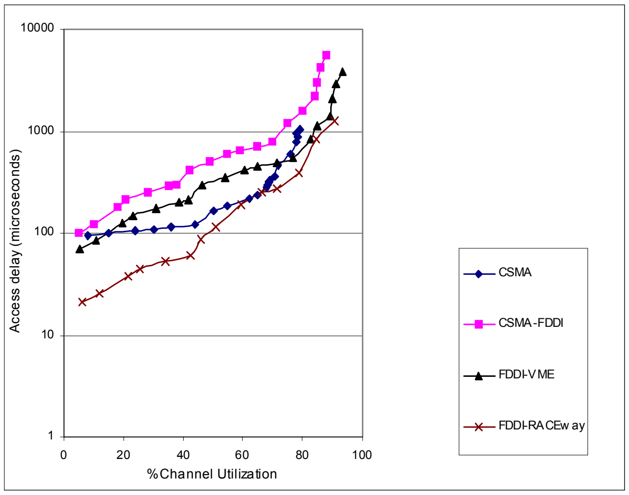

**Figure 9**. Server Access Delay for 512-byte Packets.

The packet lengths used are 64 bytes, 128 bytes and 512 bytes. Figure 8 shows the access delay versus channel utilization for a packet length of 64 bytes, for four network configurations (for the sake of comparison, we include also plain CSMA/CD network). Figure 9 shows the access delay versus channel utilization for a message length of 512 bytes.

Figures 8 and 9 show that the access delay in case of RACEway segments is the lowest. Channel utilization in all the cases is slightly higher for RACEway segments when compared to CSMA/CD and VMEbus segments. One more observation that can be made from the graphs is that under heavy load, the channel utilization reaches the highest values for 512-byte messages when compared to the other two message sizes (Figure 10). This is consistent with the results discussed in [11].

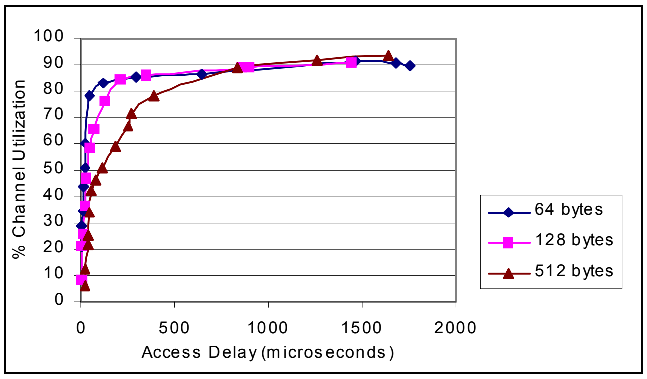

**Figure 10**. Effect of Increased Packet Length on Channel Utilization for RACEway.

## CONCLUSION

We built a model of a heterogeneous local area network, consisting of FDDI and CSMA/CD backbones with segments based on VMEbus and RACEway interconnects. Then we examined the effects that local bus traffic has on overall system performance, using server access time as a measure. We modeled a wide range of network types and traffic patterns in order to see the impact of the bus traffic and its associated protocols on the application performance.

The results show that local bus traffic on the CSMA/CD or FDDI LAN has a significant impact on network performance, thus studying bus traffic in the network is important to estimate overall performance. In the server access case study, RACEway with FDDI backbone demonstrated the best performance. It has the highest utilization and shortest access delay for long packets, as well as highest throughput (although the latter results are not discussed here due to space limits). Future work will involve studying switching technologies, rather than plain CSMA/CD.

## REFERENCES

[1] HyPerformix, Inc., *SES/workbench Reference Manual*, Austin, Texas, January 1999.

[2] J. Zalewski (Ed.), *Advanced Multi-microprocessor Bus Architectures*, IEEE CS Press, Los Alamitos, CA, 1995.

[3] M. Ajmone Marsan, G. Balbo, G. Conte, Comparative Performance Analysis of a Single Bus Multiprocessor Architectures, *IEEE Trans. Computers*, Vol. 31, No. 12, pp. 1179-1191, December 1982.

[4] C.W. McCarron, C.H. Tung, Simulation Analysis of a Multiple Bus Shared Memory Multiprocessor, *Simulation*, Vol. 61, No. 3, pp. 169-177, 1993.

[5] A. Kornecki, J. Zalewski, *Simulation of Multiprocessor Bus Systems for Real-Time Applications,* Proc. 1998 Conf. on Simulation Methods and Applications, Orlando, Fla., Nov. 1-3, 1998, SCS, San Diego, Calif, 1998, pp. 74-81.

[6] IEEE Std 1014-1987, *Versatile Backplane Bus: VMEbus,* IEEE, New York, March 1988.

[7] ANSI/VITA 5.1-1999 *American National Standard for RACEway Interlink*, VITA, Scottsdale, Ariz., 1999.

[8] *The RACE Multi-compute*r, Documentation Volume 1, Version 1.3, Mercury Computer Systems Inc., Chelmsford, Mass., 2001

[9] ANSI X3T9.5/83-15 REV 15, *FDDI Physical Layer Protocol (PHY),* American National Standards Institute, New York, 1987.

[10] D. Bertsekas and R. Gallager, *Data Networks*, Prentice-Hall, Englewood Cliffs, NJ, 1987.

[11] B. Albert, A.P. Jayasumana, Performance Analysis of FDDI LANs Using Numerical Methods, *IEE Proc., Comput. Digit. Tech.*, Vol. 144, No. 3, May 1997, pp. 149-154.

[12] V. Jonnalagadda, *Network Performance Simulation Involving Bus Traffic*, MSc Thesis, University of Central Florida, Orlando, Fla., 2002.

**Biographies**

Vinay Jonnalagadda and Mandar Mathure graduated with MSc in Computer Engineering, Software Engineering Specialization, from UCF, doing their theses under supervision of Dr. Zalewski. Andrew Kornecki is a professor of Computer Science at Embry-Riddle Aeronautical University. His research interests include modeling and simulation, performance evaluation, safety-critical systems, and real-time software engineering. He served on the SCS Board of Directors in the 1990s. Janusz Zalewski is an associate professor of Computer Science at Florida Gulf Coast University. His research interests include multiprocessor systems, real-time computing, safety-critical systems and software engineering. He consulted for several government organizations, such as FAA, NASA, NRC, and numerous software companies, including Lockheed Martin, Harris, and Boeing.