JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 22, 268–278 (1994)

Dynamic Reconfiguration of Optically Interconnected Networks with Time-Division Multiplexing*

CHUNMING QIAO,† RAMI MELHEM, DONALD CHIARULLI, AND STEVEN LEVITAN

Departments of Computer Science and Electrical Engineering, University of Pittsburgh, Pittsburgh, Pennsylvania 15260

Routing performance of optical interconnection networks is limited by both switch complexity and network connectivity. One way to overcome these limitations is to allocate the network bandwidth in a time-division multiplexed (TDM) fashion. With this technique, an appropriate subset of input-to-output connections can be established during a time slot and all possible connections can be established over several time slots. Emulating a fully connected network, however, requires a large *multiplexing degree*, and thus introduces latencies which may be prohibitive. As a solution, we propose a technique called reconfiguration with time-division multiplexing (RTDM). With RTDM, only a subset, as required by applications, of all possible connections needs to be multiplexed in the network by letting the network go through a sequence of configurations. Network reconfiguration with TDM can be done either *statically* or *dynamically*. Static RTDM is applied when communication requirements of an application are known a priori. This paper deals mainly with dynamic RTDM, which requires run time control to accommodate dynamic connection requests. We show that reconfiguration overhead can be amortized over a sequence of configurations. In particular, we describe how the complexity of a dynamic reconfiguration control algorithm can be reduced through pipelined processing of requests. Our simulations show that dynamic RTDM allows for fair and fast allocation of network resources to connection requests. As a result, network service time decreases and communication efficiency increases. © 1994 Academic Press, Inc.

1. INTRODUCTION

Optical fiber communication technology has been successfully applied to telecommunications and local area networking. The same technology may be used to alleviate the communication bottlenecks in multiprocessor systems. With advances in VLSI technology, processing speed has grown faster than communication capacity, thus creating a communication bottleneck in multiprocessor interconnection networks [9]. It seems logical to use high bandwidth optical channels to implement these interconnects. Hybrid opto-electronic multiprocessor systems, with optical communication subsystems and electronic processing and control, has been the subject of a large body of recent research [7, 8, 16, 18]. For such a hybrid system to be successful, two design issues must be resolved. The first is the mismatch between the high bandwidth of optical communication and the relatively slow electronic data processing rate. For this problem, multiplexing techniques can be applied [3, 13], in which high bandwidth channels are shared by a number of transmitters and receivers. With time-division multiplexing, processors can send packets of data at the high optical rate, but at intervals dictated by their data processing rates [5, 10].

The second design issue that has to be dealt with in hybrid systems arises from the fact that control algorithms for the interconnection subsystems must be implemented in electronics, and thus cannot effectively match the bandwidth of the optical channels. Specifically, electronically controlled optical switches, such as lithium niobate directional couplers [1, 4], can have switching speeds in the range from hundreds of picoseconds to tens of nanoseconds. This is much faster than the speed of algorithms used to control circuit switching. As a result, there will be times when new connections cannot be established as soon as the network resources become available, resulting in wasted network bandwidth. Note that packet switching can be used to reduce the complexity of the control algorithms. However, in electro-optical systems, the address decoding required for packet switching usually means electronic–optical (E/O) conversions and the use of optical memory [15]. This may be justifiable if the propagation delay of a link is comparable to the address decoding overhead or when centralized control is impossible due to the geometric spreading of the switching elements, as, for example, in local area networks [16]. However, both E/O conversions and the use of optical memory could be costly in multiprocessor systems.

Reconfiguration with time-division multiplexing (RTDM), proposed in [14], is a connection paradigm which takes into consideration both of the above two design issues. With RTDM, the interconnection network is reconfigured by sequencing through a number of network configurations, each establishing a set of connections. Switches do not need to have message buffering or routing arbitration mechanisms. This makes electro-optical switching devices suitable for implementing the network. More importantly, because a configuration can be modified independently of the others in the sequence,

* This work is supported, in part, by the Air Force Office of Scientific Research under Grant AFOSR-89-0469.

† The author is now with the Dept. of ECE, State Univ. of New York at Buffalo, Amherst, NY 14260.

0743-7315/94 $6.00

concurrent processing of requests and overlapping of communications with network reconfigurations are achieved. In addition, a sequence of configurations can capture communication locality. This is analogous to a set of *pages* capturing memory reference locality in a virtual memory system. As a result, reconfiguration overhead can be amortized over the sequence of configurations, resulting in higher network bandwidth utilization and communication efficiency.

In Section 2, we formally describe RTDM as it applies to general interconnection networks. We give the background for static and dynamic reconfigurations with TDM. We then, in Section 3, turn our focus to dynamic reconfiguration with TDM. We discuss several design issues related to dynamic reconfiguration with TDM and show how the complexity of control algorithms can be reduced by pipelined processing. In addition to dynamic reconfiguration with a fixed multiplexing degree, an adaptive approach with a variable multiplexing degree is proposed. In Section 4, we determine criteria for the evaluation of network performance, and based on these criteria, we analyze, using simulation, the performance of TDM networks. We show how and why dynamic reconfiguration with TDM can result in fair and fast allocation of the network resources, thus increasing the communication efficiency and alleviating communication bottlenecks. In Section 5, we devise a way to determine the optimal length of a control cycle, during which a sequence of network configurations is determined. This optimal length of a control cycle, in turn, determines the frequency of dynamic reconfiguration that leads to high network performance. Finally, in Section 6, we conclude the paper.

2. BACKGROUND

In this section, we give the background for the proposed connection paradigm, called Reconfiguration with time-division multiplexing (RTDM). This paradigm is applicable to a general class of interconnection networks where a physical connection between a source and a destination can be established by properly setting the switches in the network. The RTDM paradigm is particularly suitable to photonic switching networks with electro-optical switches.

2.1. General Interconnection Network: INET

Let INET denote an $N \times N$ interconnection network connecting a set, I, of N input ports to a set, O, of N output ports. We refer to a connection as a physical path between an input port and an output port in the network. Ideally, one would like to have a completely connected network, in which all possible N^2 connections are established simultaneously. However, the cost of building such a network with size $N \times N$ for a reasonably large N may be prohibitive. In practice, most multiprocessor interconnection networks may establish any of the N^2 connections but have limited connectivities such that not all of them can be established at the same time. Given an INET with limited connectivity, only a subset of all possible connections may be established at a time without conflict. Call such a subset a *mapping*, M, and denote the network configuration that establishes the connections in the mapping by $C(M)$. We will use $|C|$ to denote the maximal number of connections in a mapping in the INET.

Buses, multistage interconnection networks (MINs), and crossbars are a few examples of INETs. A bus may establish only one connection at a time and hence $|C| = 1$. Arbitration is used to resolve conflicts between multiple connection requests and to ensure exclusive access of the bus. A MIN may establish N connections at a time, which corresponds to one of a few, but not all possible $N \times N$ permutations. A crossbar may establish N connections corresponding to any $N \times N$ permutation at a time. Nevertheless, in both networks, $|C| = N$.

In order to overcome the limited connectivity of an INET, proper connection paradigms need to be developed. In general, the complexities of the network hardware and control algorithms depend on both the topology of the network and the connection paradigms [6]. Therefore, it is important to develop a connection paradigm that is both suitable for the implementation technology of the interconnection networks and efficient for the applications.

2.2. Reconfiguration with Time-Division Multiplexing (RTDM)

Let $M_1, M_2, \ldots, M_K$ be mappings in an INET. Define a *time slot* to be a fixed period of time and let the time domain be divided into a repeated sequence of K time slots. Using the RTDM connection paradigm, the connections are established by letting the INET go through a sequence of network configurations, $C(M_1)$, $C(M_2)$, ..., $C(M_K)$ repeatedly, using one time slot for each configuration. More specifically, the INET is reconfigured into $C(M_i)$ in time slot i for $1 \le i \le K$. As a result, each connection in $\bigcup_{i=1}^{K} M_i$ is established at least once in every K time slots. The number of configurations in the sequence, K, is called the *multiplexing degree*. Note that, when $K = 1$, the behavior of the network degenerates to that of a network which uses conventional circuit switching without time division multiplexing.

In previous work [21], a multistage interconnection network called *dilated slipped banyan* (DSB) was proposed by Thompson as a central switching hub for an optical local area network. The DSB emulates a completely connected network in the time domain. That is, switches in the DSB are set during the tth time slot such that an input port i is connected to output port $j = i \text{ xor } t$, where $0 \le i, j, t \le N - 1$, and xor is the bit-wise *exclusive-or* operation. Within every N time slots, any input port is connected to every output port once. By reconfi-

guring an INET into a configuration sequence $C(M_1)$, $C(M_2)$, ..., $C(M_K)$, a RTDM network can emulate a completely connected network if $\bigcup_{i=1}^{K} M_i = I \times O$. The DSB discussed above is a special case of RTDM networks which implements a *completely connected* (CC) configuration sequence with the multiplexing degree $K = N$.

Obviously, in an INET, a CC configuration sequence may be used to support any application. However, since latency is directly proportional to the multiplexing degree, such an approach is not efficient for large systems. In addition, communication locality implies that not all possible connections are required by an application program all the time.

To improve the network bandwidth utilization and reduce connection latency, either *static* or *dynamic* RTDM may be performed at an appropriately reduced multiplexing degree. With static reconfiguration, a network configuration sequence is predetermined and does not change at run time. Therefore, it is effective when no connection requests are made dynamically at run time. Otherwise, dynamic reconfiguration with TDM may be performed.

Figure 1 is a schematic diagram of a communication subsystem in which static RTDM can be applied. In other words, communication requirements are known a priori, having been derived either from compile time analysis, or from target communication structures to be embedded in the INET. Based on these requirements, the *control unit* constructs a configuration sequence and determines the state of switches for each of the configurations in the sequence. The logical control signals (0s and 1s) that can be used to set the switches in the network into the desired states are then forwarded to and stored in the *mapping generator,* which may consist of a set of shift registers, one for each switch in the INET. More specifically, control signals to be issued at the beginning of the ith time slot, which set the network to configuration $C(M_i)$ during that time slot, are stored as the ith bit of these registers. When execution begins, the mapping generator shifts cyclically all shift registers bit by bit so that the INET goes through each configuration in the sequence in a round-robin fashion.

Once the control unit determines the configuration sequence, both sources and destinations are informed of that sequence. As a result, each source or destination

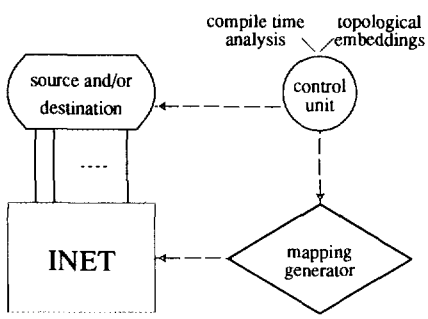

FIG. 1. Static reconfiguration with TDM.

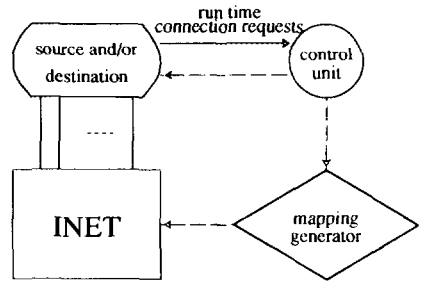

FIG. 2. Dynamic reconfiguration with TDM.

knows which destination or source, respectively, it is connected to during any given time slot. By synchronizing all sources and destinations at each time slot, high bandwidth utilization may be achieved due to the absence of run time control overheads and the bounded maximum connection latency.

Static reconfiguration with TDM in a multistage interconnection network was studied in [12]. Embeddings of regular communication structures such as rings, meshes, hypercubes, cube-connected cycles, and binary trees have been studied. In addition, algorithms have been developed to determine configuration sequences with minimal multiplexing degree based on arbitrary sets of connection requests. Having provided this background, we now turn our focus to dynamic reconfiguration with TDM in the rest of this paper.

3. DYNAMIC RECONFIGURATION WITH TDM

If the connection requirements of an application program are not known a priori or change dynamically during execution, *dynamic reconfiguration* may be performed to accommodate run time requests and improve communication efficiency. In this section, we discuss issues related to dynamic reconfiguration with TDM. Figure 2 is a schematic diagram of dynamic reconfiguration with TDM in an INET. It differs from Fig. 1 in that the input to the control unit in this figure is a set of run time connection requests. There are two ways to specify when a connection will be *released,* namely, to issue an explicit release request or to specify the duration of a connection at the time a request is made. In the rest of this paper, we adopt the latter approach and thus no explicit release requests are needed. The duration of a connection is specified by the transmission time of the *message* over the connection. A message is divided into a number of data packets, with each data packet being transmitted during one time slot.

Dynamic reconfiguration can be performed with either a variable or a fixed multiplexing degree. One way to dynamically reconfigure an INET with a variable multiplexing degree is to *periodically* perform static reconfigurations to accommodate as many connection requests as needed. We may trade the flexibility of this approach, however, with simple, uniform control by fixing the mul-

tiplexing degree. With a fixed multiplexing degree, any configuration sequence has a fixed number of configurations. Thus, the size of each shift register is fixed and synchronization control is easier. We will defer the detailed description of an adaptive approach with a variable multiplexing degree to Section 3.3. In the next two sections, we will assume dynamic reconfiguration based on a fixed multiplexing degree. In Section 3.1, we discuss overhead due to time division multiplexing, and in Section 3.2, we show that the complexity of the control algorithm used for dynamic reconfiguration with TDM can be reduced by pipelined processing.

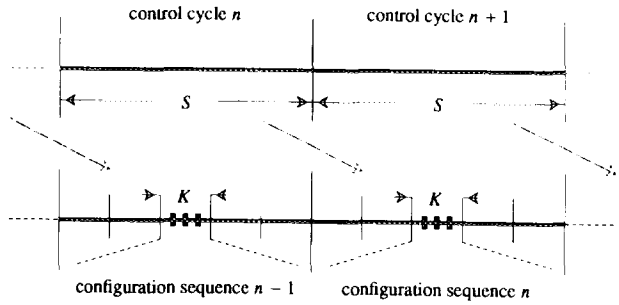

FIG. 3. Determining a configuration sequence in a control cycle.

3.1. Overhead Due to Time-Division Multiplexing

Before we show the network performance improvement due to dynamic reconfiguration with TDM, we examine the overhead involved in time division multiplexed communications. Assume that the length of a time slot is L and let L_d denote the transmission time of a data packet. With time-multiplexing, $L > L_d$. More specifically, $L = L_d + L_g$, where $L_g(>0)$ is the time needed for guard bands, which should be at least as large as the network end-to-end propagation delay plus the time to change the states of switches. For example, an L_g equal to 24 ns would be adequate if the switching time is less than 10 ns and the end-to-end propagation distance is less than a few feet.

The percentage of overhead due to time-multiplexing during each time slot is given by L_g/L, that is, $L_g/(L_d + L_g)$. Usually, L_g is determined by the implementation technology. Hence, at a fixed transmission rate, a smaller packet size leads to a lower efficiency. On the other hand, when a data message is divided into multiple packets, the last packet to be transmitted over a connection may contain less data than the maximal allowable packet size, causing fragmentation in the corresponding time slot. The larger the packet size is, the bigger the fragmentation may become, resulting in more wasted bandwidth. At a 1-Gbit transmission rate, 256 ns are needed to transmit a packet of 256 bits. This makes the width of a time slot $L = L_d + L_g = 256 + 24 = 280$ ns. The percentage of overhead during each time slot is thus less than 10%. If higher transmission rate (e.g., 2.4 Gbit) is used, the packet size has to be increased to maintain this low percentage overhead assuming the same L_g. Note that, optimal choices of the packet size and the width of a time slot depend on many other factors as well and will not be discussed here.

3.2. Reducing Control Complexity with Pipelined Processing

Centralized control algorithms used in circuit switching are usually too slow in handling connection requests to be practical for use in any large network. In this section, we show how the effective complexity of a centralized control algorithm can be reduced when time division multiplexing is used.

Assume that the control unit invokes a control algorithm every S time slots to determine the mapping M_i and assign source-destination pairs for each time slot i, $i = 1, 2, \ldots, K$. Let the period of time between two consecutive invocations of the algorithm be called a *control cycle*. During control cycle n, the control algorithm determines configuration sequence n, which will take effect during the next control cycle $n + 1$. Meanwhile, the INET is reconfigured into the configuration sequence $n - 1$, which is determined in the previous control cycle. This is illustrated in Fig. 3. Note that, during a control cycle, the INET is reconfigured into a sequence of K configurations S/K times. Here is it assumed that S is a multiple of K.

Let τ_n be the time that control cycle n begins and let $\tau_{n+1} = \tau_n + S \times L$ be the time that control cycle $n + 1$ begins. When the control algorithm is invoked at time τ_n, it first determines which existing connections will be released by time τ_{n+1} based on the time of its establishment and its duration. If an existing connection will not be released by time τ_{n+1}, it will be established in the new configuration sequence. That is, there will be no preemption of the existing connections.

Having examined the existing connections, the control algorithm examines each request in a queue maintained by the control unit. A request will be *granted* if the requested connection can be established in the new configuration sequence and *rejected* otherwise. A rejected request will be put back in the queue at the end of the control cycle. It will not be considered again until the next control cycle begins. Note that, a request will be considered *satisfied* only if message transmission over the requested connection is completed. We will refer to requests that have been *granted* but not yet *satisfied*, as *E-type* (established) requests, and those that are in the request queue as *B-type* (blocked) requests. A B-type request will change to an E-type request after it has been granted in a control cycle.

Figure 4a shows the flow of requests in the control unit, in which a *resource allocator* (RA) runs the control algorithm. In a network using conventional circuit-switching without time-division multiplexing (e.g., when

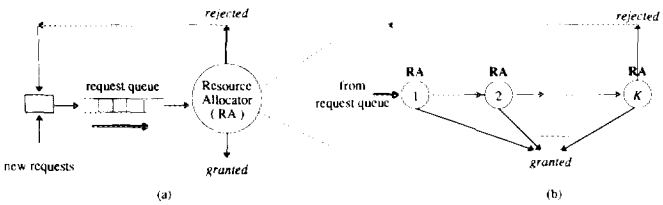

FIG. 4. Examining connection requests: (a) flow of requests in the control unit and (b) pipelined processing with multiple resource allocators.

$K = 1$), requests have to be examined one at a time to ensure that the INET is conflict-free when establishing connections in the network. Let the queue length be the number of requests in the request queue at the beginning of a control cycle and denote it by Q. If it takes S_c time slots to examine a request, then in order to examine all requests in the queue in a control cycle, S has to be larger than or equal to $S_c \times Q$.

Since a configuration in a sequence can be modified independently of the others, we may examine requests in a pipelined fashion to reduce the length of a control cycle when RTDM is used (i.e., when $K > 1$). More specifically, we may use K resource allocators (RA) in the control unit. Each RA executes the control algorithm and is capable of determining one of the configurations in the new sequence. As illustrated in Fig. 4b, when a control cycle begins, the first request in the queue is examined by RA_1 to see if it can be granted in the first configuration of the new sequence. If not, the request is forwarded to RA_2 for further examination while RA_1 starts to examine the next request in the queue. Eventually, all K resource allocators may examine requests concurrently in a pipelined fashion. The time needed to examine all Q requests with K resource allocators is thus $(Q + K - 1) \times S_c$ time slots. When $Q \gg K$, this can be approximated by $Q \times S_c$. Therefore, when using either circuit switching without RTDM (that is, $K = 1$), or RTDM ($K > 1$), S has to satisfy the following equation:

$$S \geq S_c \times Q. \tag{3.1}$$

However, when using RTDM, the amortized reconfiguration time for each configuration, which is S/K, is reduced. This is because within a control cycle of S time slots, a sequence of K configurations is determined. Note that the overhead involved in multiplexing, as discussed in the previous section, is low compared to the complexity of the control algorithm. Thus, the overall control complexity can be reduced through reconfiguration with TDM.

Note that if S is chosen to be smaller than $Q \times S_c$, not all requests in the queue can be examined during a control cycle. As a result, the network bandwidth may be underutilized and the queue length may keep increasing as requests accumulate. On the other hand, the network needs to be reconfigured into a new configuration sequence as soon as that configuration is determined in order to reduce connection latency. In other words, S should not be unnecessarily long. In this and the following sections (Sections 3 and 4), we assume that an S is chosen large enough to allow all the requests in the queue to be examined during a control cycle. In Section 5, we show how an appropriate S may be chosen to achieve high communication efficiency.

3.3. Reconfiguration with Variable Multiplexing Degrees

So far we have considered dynamic reconfigurations with a fixed multiplexing degree. Having a low multiplexing degree improves the connection latency of E-type requests but may have unfavorable effects on B-type requests. Based on a certain performance criterion, it will be shown in Section 4.2.1 that there is a multiplexing degree at which dynamic reconfiguration achieves overall optimal performance for a specific application. Such a multiplexing degree is thus called *optimal* for the application, and is denoted by K_o. Different applications may have different values of K_o. If the K_o is not known for an application, one may rely on a fixed multiplexing degree which may be different from K_o for that application. In doing so, there is a risk of severe performance degradation if the chosen multiplexing degree is not close enough to K_o. In order to avoid this degradation, a variable multiplexing degree may be used in dynamic reconfiguration. With this approach, the multiplexing degree is adapted to the run time communication requirements of the application. This may be done by *periodically* invoking static reconfiguration algorithms [12]. These algorithms construct a configuration sequence that establishes as many connections as required by using as many time slots as necessary. Note that, such a configuration sequence may contain many configurations in which only a few connections are established. These configurations may be excluded from the sequence to reduce the multiplexing degree.

To achieve this goal, we define the *utilization* of a configuration $C(M)$ to be the ratio of the number of connections in M over $|C|$. We propose an adaptive approach in which a configuration with utilization lower than a threshold value, denoted by MU ($0 \leq \text{MU} \leq 1$), becomes a candidate for exclusion from the sequence. Note that the multiplexing degree of each configuration sequence is inversely proportional to MU. By choosing an appropriate MU, the multiplexing degree can be adapted to the optimal one.

As with a fixed multiplexing degree, it is also assumed that established connections will not be preempted between different configuration sequences. This implies that not all configurations in a sequence should be required to maintain their utilizations above MU. More specifically, configurations in which E-type requests are

granted are included in the new configuration sequence no matter what their utilizations are. On the other hand, a configuration in which only B-type requests are granted will be included in the sequence only if its utilization is above MU. In other words, all E-type requests will be granted but a B-type request will be granted only if either of the following two conditions is satisfied.

(1) the request is granted in a configuration in which at least one E-type requests is granted *or*

(2) the request is granted in a configuration with utilization higher than MU.

For instance, if MU = 0, then all B-type requests will be granted. This is not the case if MU > 0. In Section 4.2.2, we show that a near optimal performance can be obtained using this adaptive approach with a variable multiplexing degree.

4. PERFORMANCE EVALUATION

In this section, we first develop performance measures of an INET, including network capacity, request waiting time and normalized service time. Based on these performance criteria, dynamical RTDM is compared with circuit switching without RTDM (which is similar to RTDM with $K = 1$).

4.1. Performance Measures

The amount of traffic in an INET may be measured in terms of the average number of requests generated by each source in the INET during each time slot. Let this number be p and define the *packet rate* per source to be $P = p \times D$, where D is the *average* connection duration in terms of number of data packets. Recall that $|C|$ is the maximal number of connections that can be established in the INET and thus it is the maximal number of packets that can be sent during each time slot. Therefore, P has to be less than $|C|/N$, which is usually not greater than 1.

A measure of the network performance is its maximal capacity, which is the amount of traffic that the network can sustain without saturation. A network will be saturated if the traffic is so heavy that the length of the request queue keeps increasing. Let the *critical rate* be the highest possible packet rate beyond which the network will be saturated. It can be seen that the network capacity is N times its critical rate. As will be shown later, the critical rate of a network depends on several parameters. In particular, the higher the multiplexing degree is, the higher the critical rate becomes, resulting in a higher network capacity.

Let $P_c(K)$ be the critical rate of a network reconfigured with multiplexing degree K. Since $P_c(1)$ equals the critical rate of a network without time-multiplexing, the ratio of $P_c(K)/P_c(1)$, denoted by $\alpha(K)$, represents the increase in the critical rate as well as in the network capacity due to RTDM. We will refer to the network capacity corresponding to $P_c(1)$ as the *base* capacity. For any given packet rate P, which is less than the critical rate, the ratio $\beta = P/P_c(1)$, represents the actual traffic as a percentage of the base network capacity.

Let W_r be the waiting time of a request, that is, the time it spends in the queue before the requested connection is established. The average waiting time of all requests, denoted by W, can be written as $ave(W_r)$ where $ave(\ )$ denotes the average over all requests. Note that, if the request queue is stable, W increases when the average length of the request queue increases. W may be used to measure the responsiveness of a network. In addition, the standard deviation of W, denoted by $\sigma(W)$, may be used to estimate how fairly the network responds to requests. As will be shown in the next section, one of the advantages of using RTDM is that both request waiting time and its standard deviation are reduced due to time sharing of the network bandwidth.

The overall quality of the service provided by the network may be measured in terms of the *normalized (request) service time*, denoted by NST [22]. For a single request, this is defined to be the ratio of the total service time over the connection duration. The total service time of a request is the time from the generation of the request until the time that the request is satisfied. This includes the request waiting time, W_r, and the time to complete the message transmission over the requested connection, which is denoted by T_r. Let D_r be the connection duration, measured in time slots, of a requested connection. Clearly, T_r depends on D_r since a message is divided into D_r data packets. The NST is defined to be the average of normalized request service time over all requests; that is,

$$\mathrm{NST} = ave[(W_r + T_r)/D_r]. \tag{4.1}$$

Similar to the standard deviations of the average waiting time, the standard deviations of NST, denoted by $\sigma(\mathrm{NST})$, may be used to estimate the fairness of the network in providing services.

If a fixed multiplexing degree K is used, then $T_r = D_r \times K$ since a data packet is transmitted over a connection every K time slots. Therefore, the above equation may be simplified to

$$\mathrm{NST} = ave[(W_r + D_r \times K)/D_r] = K + ave(W_r/D_r). \tag{4.2}$$

As the multiplexing degree increases, the first term increases and the second term decreases due to the decrease in W_r. The optimal multiplexing degree, K_o, is the value of K that minimizes NST in the above equation.

In the next section, the performance of a network that is dynamically reconfigured with a fixed multiplexing degree will be evaluated in terms of these measures. Some of the evaluation results are also applicable to the adaptive approach with a variable multiplexing degree. In particular, we will examine how the NST value obtained by using an optimal multiplexing degree can be approxi-

mated by using a variable multiplexing degree in Section 4.2.2.

4.2. Simulation Studies

We have conducted a detailed simulation study to address the following issues:

(1) the increase in the network capacity due to RTDM;

(2) the choice of the fixed optimal multiplexing degree K_0 which minimizes NST;

(3) the improvements due to RTDM, measured by the reductions in the average delay, the normalized service time (NST), and their standard deviations;

(4) the sensitivity of NST and K_0 to changes in the value of network parameters N and S, and that of application parameters D and P;

(5) the effectiveness of the adaptive approach with a variable multiplexing degree.

A multistage interconnection network (MIN) has been chosen as a specific INET to be studied. Such a MIN with N inputs and N outputs is shown in Fig. 5 for $N = 8$. It has $\log N$ stages with generalized cube connections between stages [17]. We assume that electro-optic devices such as lithium niobate directional couplers are used as switches in the MIN. These 2×2 switches can be set to one of the two states, namely, straight and cross. Note that, networks with redundant paths and those using switching elements of larger sizes or more states (such as broadcasting states) may improve network performances at the expense of higher hardware and control complexities. With reference to Fig. 2, the rectangular box marked by INET in that figure can now be replaced by the MIN just described. All simulation results have been obtained by limiting the packet rate to be less than the critical rate. The results obtained when a network is saturated are meaningless and thus ignored.

For simulation purposes, we have assumed that the destination of each requested connection is uniformly distributed among all possible destinations. Note that, for real applications which have localized communications, more positive results may be expected when RTDM is used for the reasons mentioned in Sections 1 and 2.2. Let the duration of connections be a uniform distribution with the average D. A 10% overhead for each time slot due to the guard bands is assumed whenever the multiplexing degree K exceeds 1. This assumption makes the results obtained for RTDM with $K = 1$ close to those obtainable for circuit switching without RTDM.

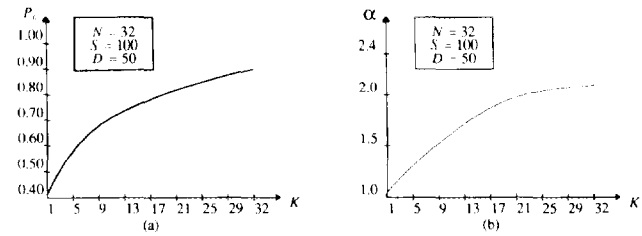

FIG. 6. Increases in (a) the critical rate P_c and (b) the network capacity (measure by α) with K.

4.2.1. Simulation of Networks with a Fixed Multiplexing Degree. In this subsection, we present simulation results assuming a fixed multiplexing degree. These results have shown that the network critical rate, P_c, increases as the multiplexing degree K increases. Figure 6a shows the network critical rate as a function of the multiplexing degree in a 32×32 network. From the figure, we have $P_c(1) \approx 0.42$ and $P_c(5) = 0.5$, That is, the network critical rate increases by approximately 1.2 times when $K = 5$. At the same time, the network capacity is also increased by $\alpha(5) = 1.2$ times, as shown in Fig. 6b. This increase is mainly due to the better utilization of the network bandwidth and reduced control complexity through time-multiplexing. Note that, with $K = 32$, a completely connected network can be emulated statically to satisfy all possible requests. In this case, the network critical rate is 1.0 (the dotted line in Fig. 6).

We have also found that the average waiting time, W, and its standard deviation, $\sigma(W)$, decrease as the multiplexing degree increases, as shown in Fig. 7a and 7b, respectively. That is, with time-multiplexing, requests are responded to faster and more fairly by the network than without time-multiplexing. Note that, this phenomenon is also common in other time-shared environments as long as no idle time slots are forced.

Figures 8a and 8b show, respectively, the standard deviation of the normalized service time and its average as a function of K. It can be seen that the standard deviation of NST also decreases with K, thus reinforcing the fact that the network provides services more fairly with a larger multiplexing degree.

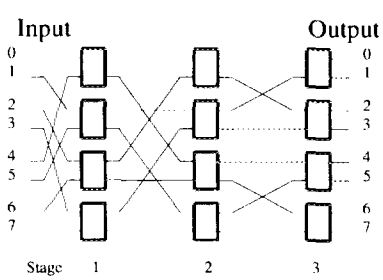

FIG. 5. A MIN with generalized cube connections.

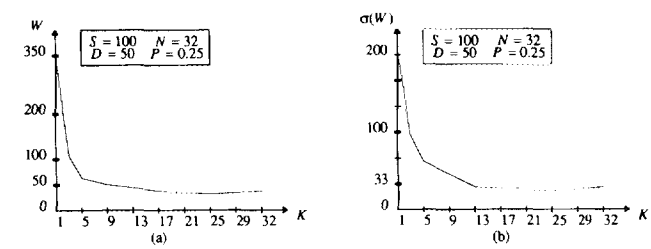

FIG. 7. Decreases in (a) the request waiting time W and (b) its standard deviations $\sigma(W)$ with K.

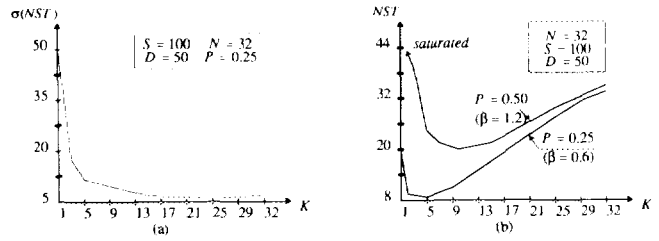

FIG. 8. (a) Decreases in the standard deviations of the normalized service time σ(NST) with K and (b) the effect of the packet rate P on the optimal multiplexing degree which minimizes NST.

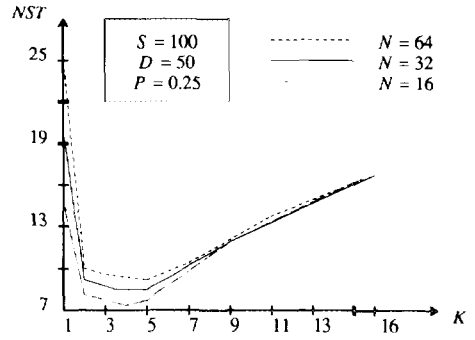

FIG. 10. The effects of the network size N.

As expected, Fig. 8b shows that there is an optimal multiplexing degree that minimizes the NST, namely, $K_o = 5$ and $K_o = 10$ in the two curves respectively. What is important is that the heavier the traffic is, the better the improvement on NST when the network is time multiplexed at an appropriate degree. For example, with $\beta = 0.6$, the maximal improvement on NST is about 2.2 times (obtained with K = 5). With $\beta = 1.2$, the maximal improvement would be at least 15 times (with $K = 10$). This is because when traffic becomes heavier, the reduction in the request waiting time will be larger than the increase in the transmission time of data packets with an increased multiplexing degree. Similar phenomena can also be observed from Figs. 9a, 9b, and 10. From these results, we may conclude that time-multiplexing can be *especially* effective in improving NST when traffic is heavy, the control complexity is high, or the network size is large.

It is also worth noting that the two curves in Fig. 8b become close and the value of the NST becomes dominated by K. Similar phenomena can be observed from Figs. 9a, 9b, and 10. The reason for the above phenomena is that request waiting time and overhead due to time-multiplexing become fairly small and almost constant when K is large. Note that, when $P = 0.5$ (or $\beta = 1.2$), the network is saturated if no time-multiplexing is used. As a result, the NST value keeps increasing during simulation and its value at $K = 1$ is not plotted in the figure.

The figure also shows the effects of the amount of traffic (which is proportional to packet rate P) on the optimal multiplexing degree. It can be seen that K_o is proportional to P. Interestingly, we have found that with N, S, and D unchanged, K_o becomes 1 when $P \leq 0.15$ ($\beta < 0.4$). In addition, we have found that if $P \geq 0.65$ ($\beta > 1.6$), the minimum NST value that dynamic reconfiguration could achieve is larger than 32. From these results, we may conclude that dynamic reconfiguration with an optimal multiplexing degree improves NST in a wide range of traffic, namely $0.25 < P < 0.65$ (or $0.4 < \beta < 1.6$). For applications having less traffic than 40% of the base network capacity, dynamic reconfiguration should be done without time-multiplexing (such as circuit switching). On the other hand, for applications having traffic heavier than 160% of the base capacity, the network should be statically reconfigured to emulate a completely connected network.

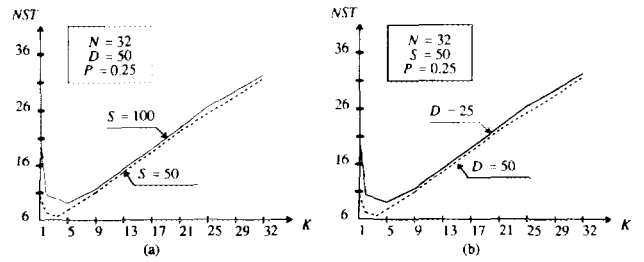

FIG. 9. The effects of (a) the control cycle length S and (b) the connection duration D.

In addition to traffic load, the optimal multiplexing degree K_o also depends on other parameters. Intuitively, it depends mostly on the ratio of S over D. In a simplistic model, all connections have the same, fixed duration D, which is less than the length of a control cycle S. In order not to waste network bandwidth during a control cycle, the network should be multiplexed with a degree of S/D. This intuition is confirmed by the simulation results. Figures 9a and 9b show respectively the effects of S and D on the value of NST and K_o. From Fig. 9, we can see that both NST and K_o are proportional to S, and inversely proportional to D. Furthermore, both solid curves in Figs. 9a and 9b have the same ratio S/D (which is equal to 2) and they are almost identical even when the parameters are different. Note that the dotted curves in both cases are exactly the same because of the same parameters used.

Finally, Fig. 10 shows the effects of the network size, N, on the NST and the optimal multiplexing degree. Three different network sizes, namely, 16, 32, and 64, are considered. From this figure, it can be seen that K_o is nearly independent of N.

We summarize in Table I the qualitative dependency of our measurements on the parameters. A "+" sign in an entry means that the measurement is proportional to the corresponding parameter, while a "−" sign means that the measurement is inversely proportional to the corresponding parameter. A double plus or minus sign (that is,

TABLE I
The Qualitative Dependencies of the Performance Measurements on the Network and Application Parameters

	P_s	NST	S(NST)	W	$S(W)$	K_o
N	×	×	+	++	+	×
S	−	+	+	++	+	+
D	+	−	−	+	+	−
P	n/a	++	++	++	++	++
K	+	−/+	−−	−−	−−	n/a

"++" or "−−") is used to indicate strong dependency, while an "×" is used to indicate negligibly weak dependency or no dependency. A "−/+" sign is used to indicate that NST decreases with K initially but increases with K when $K > K_o$. Finally, an "n/a" means *not applicable*. The last row in Table I shows that reconfiguration with time-multiplexing can improve all performance measures that we have discussed, while the last column can guide the choice of an appropriate multiplexing degree.

4.2.2. Simulation of Networks with Variable Multiplexing Degrees. As mentioned earlier, our objective in this section is to demonstrate that when the optimal multiplexing degree is unknown, an adaptive approach with a variable multiplexing degree can be used to achieve near optimal network performance. Simulation results presented in this section are obtained under the same assumptions as those made in the previous sections, except that a variable multiplexing degree is used. We illustrate the adaptive approach with two values of minimal utilization, namely, MU = 0 and MU = 1/3.

Figure 11 shows the NST value at different packet rates P. The results obtained by using the fixed optimal multiplexing degree are also shown for comparison. When the network traffic is extremely low (e.g., when $P \approx 0.1$), MU = 1/3 results in the worst performance. The reason for that is the minimal utilization required is too high thus limiting the multiplexing degree to well below the optimal multiplexing degree. At low traffic load, the NST value is dominated by the average request waiting time, which, in this case, becomes large. As the traffic load increases, MU = 1/3 results in better performance while MU = 0 results in worse performance. This is because with MU = 0, the multiplexing degree becomes higher than the optimal degree and many configurations have very low utilization (utilizations are found to be between 1/32 and 5/32). With MU = 1/3, these time slots are eliminated and the average multiplexing degree used is close to the optimal one. As a result, smaller NST values are obtained. Note that, by varying MU based on traffic loads, the adaptive approach with a variable multiplexing degree can be as effective as dynamic reconfiguration with an optimal multiplexing degree in reducing NST under all traffic load conditions.

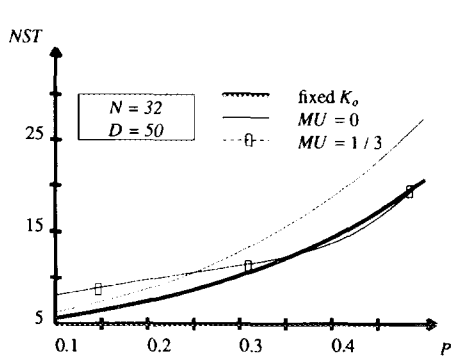

FIG. 11. Adaptive approach with variable multiplexing degrees.

5. OPTIMAL LENGTH OF A CONTROL CYCLE

The frequency of dynamic reconfiguration, be it with a fixed multiplexing degree or with a variable multiplexing degree, is determined by the length of a control cycle, S. In this section, we devise a way to determine the optimal length of a control cycle that leads to improved network performance.

So far we have assumed that all requests in the queue can be examined during a control cycle. That is, the condition expressed in Eq. (3.1), which is $S \geq S_c \times Q$, is satisfied. This condition is crucial to ensure high utilizations and the stability of the system. If not all requests can be examined during a control cycle, not only will the utilization be low, but also the request queue may keep building up.

Equation (3.1) shows that S depends on the length of the request queue, Q. Intuitively, however, Q should depend on S for a given application as well. Larger S leads to larger Q. Given different values of S, we have measured the queue length when the optimal multiplexing degree K_o is used. Let $Q(S)$ be the queue length at an optimal multiplexing degree for a given S when an application with parameter D and P is executed. It is observed that $Q(S)$ increases with S sublinearly. In other words, the function $Q(S)$ is sublinear.

In order to examine all requests in the queue during a control cycle, Eq. (3.1) must now be modified into

$$S \geq S_c \times Q(S). \tag{5.1}$$

As can be seen from the simulation results of Section 4.2.1, the NST value increases with S. In order to minimize NST, the smallest possible S should be used. This can be found by solving the equation

$$S = S_c \times Q(S), \tag{5.2}$$

where S_c is a machine-dependent constant and $Q(S)$ is a network characteristic that may be determined by simulations. This equation can be solved for S by finding the intersection of the curve $Q = Q(S)$ and the line $Q = S/S_c$, as illustrated in Fig. 12. Since $Q(S)$ is found to be a

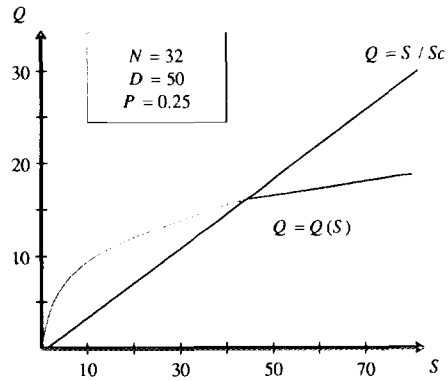

FIG. 12. Determining the optimal length of a control cycle.

sublinear function, such an intersection will exist. We call the value of S at the intersection the *optimal* S and denote it by S_0. Using an S larger then S_0 will result in a larger NST value while using an S smaller than S_0 would not satisfy Eq. (5.1).

Based on our control algorithm which may not be optimal, the time needed to examine a request is approximately the time to perform $\log N$ memory *reads*, $\log N$ memory *writes*, and 6 $\log N$ arithmetic operations for an $N \times N$ MIN. Assuming that each *read* or *write* operation takes about 45 ns and each arithmetic operation takes about 10 ns (with current microprocessors), S_c is about 750 ns when $N = 32$. If the width of a time slot is taken to be 280 ns, then $S_c \approx 2.7$ time slots. With these values, we may draw the line $Q = S/2.7$, as shown in Fig. 12. The curve $Q = Q(S)$ is obtained from simulation results with $D = 50$ and $P = 0.25$. At the intersection point of the line and the curve, $S \approx 45$.

Note that, since $S_0 \approx 45$, all the requests in the queue can be examined if $S \geq 45$. Therefore, simulation results obtained in Section 4.2 are valid because S was assumed to be either 50 or 100. However, simulation results obtained with a smaller S would be optimistic since not all the requests in the queue can be examined during a control cycle. This is the case for the dotted portion of the curve $Q = Q(S)$ in Fig. 12 before the intersection point. Note also, that the effective reconfiguration time in a time-multiplexed network can be reduced through pipelined processing, as discussed in Section 3.2.

6. CONCLUDING REMARKS

Reconfiguration with time-division multiplexing (RTDM) helps solve the problems of the relatively slower speed of data processing and the network control algorithms of the electronic processing elements as compared to the high speed of optical data transmission. In addition, it makes electro-optical switches such as lithium niobate switches suitable for the implementation of the interconnection network by eliminating the need for buffering data or making routing decisions in switches. Therefore, this technique matches current photonic switching technology and is promising in designing hybrid electrooptical multiprocessor systems. With RTDM, the network bandwidth is time-shared among different connections and can be utilized more efficiently.

As discussed in Section 2.2 and in [12], static reconfiguration with TDM enables a network with limited connectivity to efficiently support applications whose communication requirements are known a priori. It also enables such a network to efficiently emulate target networks having stronger connectivity. On the other hand, results obtained in Section 4.2 demonstrate that dynamic reconfiguration with time-division multiplexing leads to fair and fast allocations of network resources to run time connection requests. With dynamic TDM reconfiguration, the maximal capacity of a network, as well as its communication efficiency, can be increased. In supercomputing environments, large network sizes are used in which heavy network traffic and high control complexity cause unacceptable performance due to the communication bottlenecks. The simulation results (Figs. 8, 9, and 10) indicate that dynamically reconfiguring a photonic switching network with TDM is an effective solution to the problem. Path losses, crosstalks, synchronization, and polarization control are important issues in designing such photonic switching networks. These issues, which are investigated in [2, 11, 19, 20], are not discussed in this paper.

When dynamically reconfiguring a network with a fixed multiplexing degree, there is an optimal multiplexing degree which minimizes the average normalized service time for a specific application. We have shown that dynamic reconfiguration may be performed with a variable multiplexing degree. With this adaptive approach, network performances achieved with the optimal multiplexing degree can be approximated. Note that although our simulation results are obtained by assuming a multistage interconnection network, they are also indicative of the behavior of a large class of interconnection networks.

REFERENCES

1. Alferness, R., Buhl, L., Korothy, S., and Tucker, R. High-speed $\Delta\beta$-reversal directional coupler switch. *Topical Meeting on Photonic Switching*, Technical Digest Series, Vol. 13, pp. 77–78, 1987.
2. Chiarulli, D., Ditmore, R., Melhem, R., and Levitan, S. An all optical addressing circuit: Experimental results and scalability analysis. *IEEE J. Lightware Tech.*, **9**, 12 (Dec. 1991).
3. Dowd, P. High performance interprocessor communication through optical wavelength division multiple access channels. *Proceedings of 18th International Symposium on Computer Architecture*, May 1991, pp. 96–105.
4. Erickson, J., and Hinton, H. Implementing a Ti:$LiNbO_3$ 4 × 4 nonblocking interconnection network. *SPIE Integrated Opt. Circuit Eng.* **578**, (1985), 201–206.

5. Guo, Z., Melhem, R., Hall, R., Chiarulli, D., and Levitan, S. Array processors with pipelined optical Busses. *J. Parallel Distribut. Comput.* **12,** 3 (1991), 269–282.

6. Jenkins, B. K., and Giles, C. L. Parallel processing paradigms and optical computing. *SPIE Proc. Opt. Comput.* **625** (1989), 22–29.

7. Kiamilev, F., Esener, S., Ozgus, V., and Lee, S. Programmable optoelectronic multiprocessor systems. *Digital Opt. Comput.* **CR35** (July 1990), 197–220.

8. Levitan, S., Chiarulli, D., and Melham, R. Coincident pulse techniques for multiprocessor interconnection structures. *Appl. Opt.* **29,** 14 (1990), 2024–2039.

9. Markatos, E., and LeBlanc, T. Shared-memory multiprocessor trends and the implications for parallel program performance. Technical Report 420, Computer Science Department, University of Rochester, March 1992.

10. Melhem, R., Chiarulli, D., and Levitan, S. Space multiplexing of waveguides in optically interconnected multiprocessor systems. *Comput. J.* **32,** 4 (1989), 362–369.

11. Padmanabhan, K., and Netravali, A. Dilated network for photonic switching. *Photonic Switching 1987; IEEE Trans. Comm.* (Dec. 1987).

12. Qiao, C., and Melhem, R. Reconfiguration with time-division multiplexed MINs for multiprocessor communications. *IEEE Trans. Parallel and Distributed Systems* **5,** 4 (1994), 337–352.

13. Qiao, C., and Melhem, Time-division optical communications in multiprocessor arrays. *IEEE Trans. Comput.* **42,** 5 (1993), 577–590.

14. Qiao, C. A high speed interconnection paradigm for multiprocessors and its applications to optical interconnection networks. Ph.D. Dissertation, Computer Science Department, University of Pittsburgh, 1993.

15. Sarrazin, D., Jordan, H., and Heuring, V. Digital fiber optic delay line memory. *SPIE Proc. Digital Opt. Comput. II* **1215** (Jan. 1990), 366–375.

16. Sauer, J. A multi-Gb/s optical interconnect. *SPIE Proc. Digital Opt. Comput. II* **1215** (Jan. 1990), 187–207.

17. Siegel, H. J., and McMillen, R. J. The multistage cube: A versatile interconnection network. *Computer* **14** (Dec. 1981), 65–76.

18. Szymanski, T. A comparison of circuit and packet switching in a fiber-optic hypercube. *Proceedings of the International Conference on Parallel Processing,* 1990, pp. 1296–300.

19. Thompson, R. A. Architectures with improved signal-to-noise ratio in photonic systems with fiber-loop delay lines. *IEEE J. Selected Areas Comm.* **6,** 7 (Aug. 1988), 1096–1106.

20. Thompson, R. A., Anderson, R. V., Camlet, J. V., and Giordano, P. P. Experimental modular switching system with a time-multiplexed photonic center stage. *Proceedings of OSA Topical Meeting on Photonic Switching,* March 1989, pp. 212–218.

21. Thompson, R. A. The dilated slipped banyan switching network architecture for use in an all-optical local area network. *IEEE J. Lightwave Tech.* **9,** 12 (Dec. 1991), 1780–1787.

22. Wu, C.-L. and Lee, M. Performance analysis of multistage interconnection network configurations and operations. *IEEE Trans. Comput.* **14,** 1 (Jan. 1992), 18–27.

Received July 3, 1992; accepted December 4, 1992

CHUNMING QIAO was born in Suzhou, People's Republic of China, in 1965. He received a B.S. in computer science and engineering from the University of Science and Technology of China in 1985 and an M.S. and a Ph.D. in Computer Science from the University of Pittsburgh in 1990 and 1993, respectively. He is now an assistant professor of Electrical and Computer Engineering at the State University of New York at Buffalo. His current research interests include parallel computer architectures, distributed processing, and optical interconnects. Dr. Qiao is a member of the International Society for Optical Engineering and a member of the IEEE Computer Society.

RAMI G. MELHEM received a B.E. in electrical engineering from Cairo University, Egypt, in 1976, an M.A. degree in mathematics and an M.S. Degree in computer science from the University of Pittsburgh in 1981, and a Ph.D degree in computer science from the University of Pittsburgh in December 1983. Since 1989, he has been an associate professor of computer science at the University of Pittsburgh. Previously, he was an assistant professor at Purdue University and at the University of Pittsburgh. He has published numerous papers in the areas of systolic architectures, parallel computing, fault tolerant processor arrays, and optical computing. He served in program committees for several conferences and he is on the editorial board of the *IEEE Transactions on Computers*. Dr. Melhem is a member of the IEEE Computer Society, the Association for Computing Machinery, and the International Society for Optical Engineering.

DONALD CHIARULLI is an associate professor of computer science at the University of Pittsburgh. He received a B.S. degree in physics from Louisiana State University in 1976, an M.S. degree in computer science from Virginia Polytechnic Institute in 1979, and a Ph.D. in computer science from Louisiana State University in 1986. Dr. Chiarulli has been active in high performance computing research since 1980. His accomplishments include the design and construction of The Factoring Machine, a 256-bit reconfigurable VLIW machine for factoring large numbers. More recently, Dr. Chiarulli and his colleagues at the University of Pittsburgh have successfully demonstrated the first all-optical addressing circuit for use in high performance backplanes. His current research is focused on the use of locality in communication streams to reconcile the data and control bandwidth of reconfigurable optical systems.

STEVEN P. LEVITAN is the Wellington C. Carl Associate Professor of Electrical Engineering at the University of Pittsburgh. He received a B.S. degree from Case Western Reserve University in 1972, and an M.S. and Ph.D., both in computer science, from the University of Massachusetts, Amherst, in 1979 and 1984, respectively. He worked for Xylogic Systems designing hardware for computerized text processing systems and for Digital Equipment Corporation on the Silicon Synthesis project. He was in assistant professor from 1984 to 1986 in the Electrical and Computer Engineering Department at the University of Massachusetts. In 1987, he joined the electrical engineering faculty at the University of Pittsburgh. Dr. Levitan's research interests include computer aided design for VLSI, parallel computer architecture, parallel algorithm design, and VLSI design. He is a member of the IEEE-CS, ACM, SPIE, and OSA.