Time-Multiplexing Optical Interconnection Networks; Why Does it Pay Off?

Rami G. Melhem
Department of Computer Science
The University of Pittsburgh
Pittsburgh, PA 15260

Abstract

Time Division Multiplexing (TDM) is typically applied whenever the capacity of a resource is larger than the individual requirements of any of its clients. This is the case in optical interconnection networks in which the bandwidth of an optical path is much larger than the rate at which a processor can generate data. However, in addition to closing the gap between the bandwidth of electronic processing and that of optical transmission, I argue in this paper that multiplexing optical networks is essential to reduce the overhead of controlling the network. This reduction is a result of increasing the number of connections that do not require control operations. I present a preliminary model to specify the communication requirements of applications and use this model to gain some intuition about the advantages of multiplexing interconnection networks, especially in cases where communications exhibit strong locality properties.

1 Introduction

The evolution of interconnection networks has, for the most part, been driven by the increasing needs for high throughput and low latency. Optical networks have the potential for achieving these two goals by offering unprecedented bandwidth in a medium that is free from inductive and capacitive loadings, thus relaxing the limitations imposed on the bandwidth-distance product [5, 8]. Optical fibers may be used for transmitting terahertz signals in each of 1.3 mm and 1.5 mm low attenuation passbands. Moreover, optical transmission has a low error rate and low sensitivity to noise.

Many different types of photonic switching networks have been reported and demonstrated. The control of such networks, however, is usually performed in the electronic domain, and thus is very slow, relative to the large bandwidth supported by the network. This is especially true in *all-optical* networks, where optical signals travel from their sources to their destinations without any optical-to-electronic conversion at intermediate nodes or switches.

Packet switching techniques, which are usually used in electronic multicomputer and multiprocessor interconnection networks become at a disadvantage when optical transmission is used. The absence of suitable photonic logic devices makes it extremely impractical to process packet routing information in the photonic domain. Moreover, conversion of this information into the electronic domain increases the latency at intermediate nodes relative to the internode propagation delay. Although this optical/electronic conversion may be acceptable for large distributed networks, it represents a disadvantage for multiprocessor networks in which internode propagation delays are very small.

The establishment of all-optical paths from sources to destinations places a severe demand on the control and arbitration of the interconnection network. This is especially true if this control is to be done at the aggregate traffic rate of the entire network rather than at the traffic rate of the individual nodes. Fortunately, interprocessor communications exhibit both spatial and temporal locality. The former implies that, at any given time, only a subset of all possible connections is used, and the latter implies that this subset tends to change slowly with time. Thus, if the interconnection capacity of the network is large enough to accommodate the subset of connections required by the spatial locality, the control of the network will have to be done only at the rate of the change in this locality rather than at the network aggregate traffic rate. In other words, if the network can simultaneously support, without conflict, all the connections needed at any one time, then the control overhead, amortized over all connections, will yield acceptable latencies.

For practical reasons, the number of connections that an interconnection network can support without conflict at any one time is limited. This number may not be enough to support the local set of connections, especially in applications where communication demands are not uniform over time. Time division multiplexing provides a means for increasing the number of connections established concurrently. Specifically, by cycling through a series of network configurations, the local set of connections may be accommodated over consecutive time slots.

If used moderately, time multiplexing all-optical networks has little effect on the bandwidth of individual connections. This is because the very high bandwidth offered by dedicated all-optical paths exceeds, sometimes by orders of magnitudes, the bandwidth needed for most point to point connections. In essence, time multiplexing allows the large optical bandwidth to be shared by a number of individual connections.

In the next section, Time Division Multiplexing (TDM) of interconnection networks is described. In Section 3, a preliminary model for communication requests is presented and used to express the notion of communication locality. In Sections 4 and 5, it is argued, with the help of the communication model, that the multiplexing of interconnection networks is beneficial, especially for applications that exhibit locality properties. Finally, some concluding remarks are given in Section 6.

2 A General Model for TDM

An interconnection network which provides the simultaneous establishment of all n^2 paths between n input nodes to n output nodes have a high hardware complexity, and thus is not scalable beyond a small number of processors. Most networks allow only a subset, $C = \{< x, y >; 0 \leq x, y < n\}$, of the possible n^2 connections to be established at any one time. If the subset C is fixed, then messages between nodes not directly connected in C have to be routed through intermediate nodes. I will not consider this type of networks in this work since, as discussed in the introduction, it does not allow all-optical paths to be established between nodes. I will only consider switching networks in which the set of connections that may be established simultaneously can be changed by changing the state of the network. A set of connections, C, that can be established in a given state will be called a *configuration*.

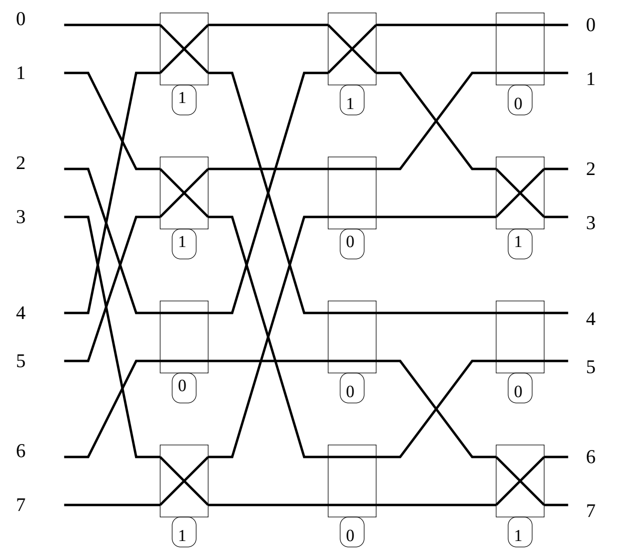

Figure 1: A multistage network.

For example, consider the multistage network shown in Figure 1, where each switch is an electro-optical switch (Ti:LiNbO$_3$ switch, for example [7]) which can be in either a *straight* or a *cross* state [10, 13]. Although each switch connects optical inputs to optical outputs without optical/electronic conversion, the state of the switch is controlled by an electronic signal. In other words, the switch state is set by loading a one bit electronic register (in the figure, 0 and 1 set the switch to *straight* and *cross*, respectively). The optical transmission rate is not affected by the relatively slow speed of changing the state of the switches (typically tens of nano-seconds) since the later is performed only when the state of the network is to be changed. In Figure 1, the switches are set to the states corresponding to the configuration $\{< 0, 4 >, < 1, 5 >, < 2, 0 >, < 3, 6 >, < 4, 3 >, < 5, 1 >, < 6, 7, >, < 7, 2 >\}$. Note that any subset of the above eight connections is also a configuration.

Another example is the mesh connected system shown in Figure 2. Each processor in the system has an input and an output connection to a 5×5 switch, which, in turns, is connected to four other switches in the north, south, east and west directions [11]. The 5×5 switch may be constructed from 2×2 electro/optical switches, and thus its state may be set by setting a few electronic bits. In Figure 2, the switches are set to states that realize the configuration $\{< 4, 1 >, < 5, 3 >, < 6, 10 >, < 8, 9 >, < 11, 2 >\}$.

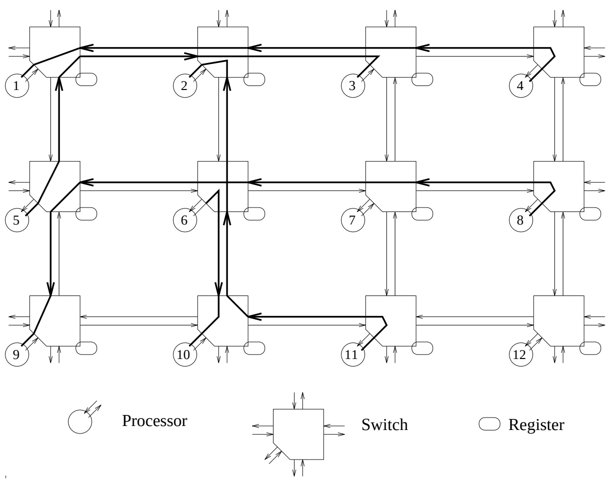

Figure 2: A mesh connected network.

If a configuration, C, is established in a switching network, establishing a new connection, $< x, y >$, which is not already in C, requires changing the state of the network. The new state should establish a configuration, C', which includes $< x, y >$. Although it is preferable to have $C' = C \cup \{< x, y >\}$, the limited capacity of the network often necessitates the removal of some connections from C to accommodate $< x, y >$.

Setting the state of the network usually requires setting registers (hardware bits) which we will call *network status registers*. Although loading these registers can proceed at hardware speed, determining the new state of the network usually requires the execution of elaborate algorithms, which, even if implemented in hardware, will be much slower.

In general, changing the configuration of the network involves three operations that will be called *control, state loading* and *switching* operations. A control operation is the determination of the new state of the network using some control algorithm. A state loading operation is the transmission of the new state to the network's status registers, and a switching operation is the actual latching of the content of the status registers into the switch circuitry, thus establishing the communication paths. Typically, control operations are slower than state loading operations, which in turns are slower than switching operations.

Time multiplexing an interconnection network is achieved by periodically cycling through a number of network states to realize a sequence of configurations, $C_1, C_2, \ldots, C_k$. The connections in each C_i are established during a fixed time period, called a time slot, and the number of states in the sequence, k, is the multiplexing degree. When the network is not multiplexed, $k = 1$.

Going back to the networks shown in Figures 1 and 2. The k-way multiplexing of the networks may be achieved by replacing each one-bit register that controls the state of a 2×2 switch with a k-bit shift register. Sequencing through the network states requires the rotation of the bits in the shift registers. The shift to the next state can be done immediately after the bit for the current state is latched onto the switch circuitry, and thus will be considered part of the *switching operation*, In other words, sequencing through the states will not require any *control operation* or *state loading* operations, as long as the sequence is not changed.

Given that switching operations are faster than control and state loading operations, we define a set of *realizable connections*, R, as a set of connections in which any connection can be used without the need for control or state loading operations. Hence, when the network is multiplexed through $C_1, \ldots, C_k$, the set of realizable connections is $R = C_1 \cup \ldots \cup C_k$.

3 Modeling Communication Requirements

The demand that an application running on a parallel system puts on the interconnection network may be represented by a series of chronologically ordered connection requests $cr_1, \ldots, cr_m$. Each request, cr_i is to establish a connection, $< x, y >$, from node x to node y in the network. Denote the connection requested in cr_i by $c(cr_i)$.

The connections requested in $cr_1, \ldots, cr_m$ may be grouped into a series of sets $M_1, M_2, \ldots$, where each M_i contains the connections requested in consecutive requests starting at some cr_{s_i} and ending at cr_{e_i}, where $s_i \leq e_i = s_{i+1} - 1$. Any request, cr_j such that $s_i \leq j \leq e_i$, is said to be in the i^{th} *locality region*. Note that although $M_i = \{c(cr_j) : s_i \leq j \leq e_i\}$, the number of connections in M_i may be much smaller than $e_i - s_i$. This is the case if, due to communication locality, a small number of connections are requested repeatedly between cr_{s_i} and cr_{e_i}. Each set M_i is called a *locality set* and $M_1, M_2, \ldots$ is called a *locality series*.

Clearly, the grouping of connections into locality sets is not unique. Larger locality sets yield a shorter locality series. An application exhibits good communication locality if $cr_1, \ldots, cr_m$ can yield a short series of small locality sets.

In an attempt to characterize spatial and temporal locality in terms of the above grouping of communication requests, define an *optimal* locality series, $M_1^{opt}, M_2^{opt}, \ldots$, resulting from $cr_1, \ldots, cr_m$ as the one that minimizes both the size of the locality sets and the length of the locality series, according to some measure that combines these two criteria. Hence, for an application with connection requests $cr_1, \ldots, cr_m$, the sizes of the sets M_i^{opt} can be used to characterize the spatial locality. Moreover, the length of the optimal locality series $M_1^{opt}, M_2^{opt}, \ldots$ can be used to characterize the temporal locality, since a shorter series means that spatial locality changes slowly with time.

When an application is executed on a parallel system, the number of network control operations needed to satisfy all the requests may be estimated by applying the same idea of grouping the connections requested in $cr_1, \ldots, cr_m$ into a locality series, $R_1, \ldots, R_q$ with the restriction that each R_i be a maximal set of realizable connections. In other words, when forming R_i starting at cr_{s_i}, connections are added to R_i as long as R_i remains a set of realizable connections. The next locality set, R_{i+1} starts at $cr_{s_{i+1}}$, where $c(cr_{s_{i+1}})$ is the first connection that renders R_i non-realizable. The series $R_1, \ldots, R_q$ will be called the *realizable series.* Given a realizable series, the number of control operations, σ, required to satisfy $cr_1, \ldots, cr_m$ is bounded by:

$$\sigma_{min} \leq \sigma \leq \sigma_{max} \tag{1}$$

where

$$\sigma_{max} = \sum_{i=1}^{q} |R_i| \tag{2}$$

$$\begin{aligned} \sigma_{min} &= \sum_{i=1}^{q} |R_i| - \sum_{i=2}^{q} |R_i \cap R_{i-1}| \\ &= |R_1| + \sum_{i=1}^{q-1} |R_{i+1} \sim R_i| \end{aligned} \tag{3}$$

where $|R|$ denotes the cardinality of set R and $\sim$ is the set difference operation.

The exact number of control operations, σ depends on the actual scheme used to deallocate existing connections in the network when new connections are established. Specifically, σ_{min} is obtained if the control algorithm can correctly predict the connections in each

R_i, such that, when a new request, $c(cr_j)$ is added to R_i, none of the connections in $R_i \cap R_{i-1}$ are deallocated. The maximum value, σ_{max}, is obtained if all the connections in $R_i \cap R_{i-1}$ are deallocated when establishing $c(cr_{s_i})$, which is the first connection in R_i. Clearly, if the sets $R_1, R_2, \ldots$ are disjoint, then

$$\sigma = \sigma_{max}. \tag{4}$$

In the following two sections, two different types of systems are considered. In the first, connection requests are completely dynamic, while in the second, most of the connection requests are known at compile time. The effect of time multiplexing on both types of systems is discussed.

4 TDM in Dynamic Environments

Two types of dynamic environments will be considered. In the first very poor locality properties are exhibited in the sense that the optimal locality series is a single large locality set. In the second very good locality properties are exhibited in the sense that the optimal locality series is composed of small, disjoint, locality sets. In both cases, it will be assumed that within a locality region, connections are requested randomly.

For a non-multiplexed network, the size of a realizable set of connections, R, is equal to the size of a single configuration that can be established simultaneously in the network. With time multiplexing, however, the size of R may be changed by changing the multiplexing degree. This added flexibility is expected to reduce the number of control operations in the network.

Consider the case in which the optimal locality series consists of a single locality set, M, of size μ, and assume that $R_1, ..., R_q$, for an even q, is the realizable series corresponding to a given network, If the network is multiplexed with degree two, then the size of realizable sets may be doubled. This results in a realizable series $R'_1, \ldots, R'_u$, where $u = q/2$ and each $R'_j = R_{2j-1} \cup R_{2j}$, is a realizable set. It can be shown that the bounds on the number of control operations given by (1) are smaller for $R'_1, \ldots, R'_u$ than for $R_1, ..., R_q$. Specifically, if σ'_{min}, σ'_{max} are the bounds for $R'_1, \ldots, R'_u$, and σ_{min}, σ_{max} are the bounds for $R_1, ..., R_q$, then $\sigma'_{max} \leq \sigma_{max}$ and $\sigma'_{min} \leq \sigma_{min}$

The result for the upper bound is obtained by applying the fact that $|A \cup B| \leq |A| \cup |B|$, for any two sets, A and B. From this fact, it is clear that $\sum_{j=1}^{u} |R'_j| \leq \sum_{j=1}^{u} |R_{2j-1}| + |R_{2j}| = \sum_{i=1}^{q} |R_i|$.

The result for the lower bound is obtained by observing that

$$\begin{aligned} |(R_{i+1} \cup R_i) \sim (R_{i-1} \cup R_{i-2})| \leq \\ |(R_{i+1} \cup R_i) \sim R_{i-1}| \leq \\ |R_{i+1} \sim R_i| + |R_i \sim R_{i-1}| \end{aligned}$$

and using this observation to show that σ'_{min} is equal to

$$\begin{aligned} |R_1 \cup R_2| \quad & + \sum_{i=1}^{u-1} |(R_{2i+2} \cup R_{2i+1}) \sim (R_{2i} \cup R_{2i-1})| \\ \leq \quad & |R_1| + |R_2 \sim R_1| + \sum_{i=2}^{q-1} |R_{i+1} \sim R_i| \\ = \quad & \sigma_{min}. \end{aligned}$$

At the expense of complicating the notation, the above argument may be extended to the case in which the size of the realizable sets are increased k fold by time multiplexing. In order to avoid complex notation, and at the same time, provide a more quantitative estimate of the reduction in the number of control operations, consider a non-blocking network which can realize any n connections and assume that the network is multiplexed with degree k ($k = 1$ if no multiplexing is used). Hence the size of each realizable set is nk. If $\mu \leq nk$, then M is realizable, and the number of control operations is reduced to its minimum. If, however, μ is larger than nk, then the probability that a connection request is in the current realizable set equals nk/μ. Hence, control operations need to be performed at the rate of one operation every $1 - (nk/\mu)$ connection requests. However, the maximum rate, ρ, at which control operations may be performed depends on the control algorithm and on the hardware executing this algorithm. Hence, the multiplexing degree should be large enough to reduce $1 - (nk/\mu)$ to ρ.

Besides reducing the rate of control operations to one that can be handled by the control algorithm, reducing the probability that a new connection will require a control operation reduces the latency of establishing this connection. However, the overall communication latency on a given connection depends on the bandwidth of the connection, as well as on the connection establishment latency.

Due to the large bandwidth of optical transmission relative to the rate at which the electronic processors can produce data, the bandwidth of individual connections is not affected by the multiplexing of the network. This, of course, is true up to a certain multiplexing degree, k_0, after which sharing the optical bandwidth reduces the effective transmission rate on individual connections. In other words, although increasing the multiplexing degree, k beyond k_0 reduces the control overhead, it also reduces the effective transmission bandwidth on individual connections by a factor of $(k - k_0)/k_0$. Hence, to preserve the quality of individual connections, the value of k should not be increased beyond the point at which the reduction in the connection establishment delays of requested connections (control operations) is offset by the reduction in the effective bandwidth of the connections.

When the connection requests do not exhibit any locality and the optimal locality series is a single locality set, the size of this set, μ, will probably be large, and it will not be possible, with multiplexing degree smaller than k_0, to have a realizable set which is equal to the locality set. However, in applications that exhibit strong locality properties, the optimal locality series, $M_1^{opt}, M_2^{opt}, \ldots$, consists of relatively small, disjoint, locality sets. In this case, according to equation (4), the number of control operations is minimized when $R_i = M_i^{opt}$. If M_i^{opt} is not a network configuration, then the number of control operations may be minimized by decomposing each M_i^{opt} into a sequence of configurations and time multiplexing these configurations to obtain $R_i = M_i^{opt}$. If the size, μ_i, of each M_i^{opt}, is small, then this may be accomplished with a multiplexing degree $k \leq k_0$.

The communication requirements of most applications do not fall into the above two locality extremes. Moreover, the optimal locality sets are not precisely defined. Even if such sets may be computed from connection requests that fall into one of the two extremes, these requests are not known in advance in dynamic systems. Hence, the optimal locality sets, M_i^{opt}, $i = 1, 2, \ldots$, are not known in advance, and it is not possible to determine the multiplexing degree that will give $R_i = M_i^{opt}$. The above analysis, however, strengthens the intuition that multiplexing reduces the number of control operations by increasing the size of the realizable sets of connections. This intuition is confirmed in various simulation studies reported in [2, 6, 12].

Next, systems in which the connection requests may be known in advance are considered.

5 TDM for Compiled Communications

Studies on numerical algorithms [3] and other benchmark parallel programs [1] show that 80% of the communication requirements of large scale parallel applications are static. That is, can be determined at compile time. An additional 19% are parametric. That is, can be expressed at compile time in terms of a parameter which is specified only at run-time. Only 1% of the communication requirement are dynamic references that cannot be predicted by the compiler.

The above observations strongly advocate the notion of compiling the communication requirements of an application [1]. Specifically, if the connection requests $cr_1, \ldots, cr_m$ are known at compile time, then the realizable series $R_1, R_2, \ldots$ and the corresponding values of the network status registers can also be determined at compile time. The compiler can then insert, in the program, instructions to load the network status registers with the predetermined values whenever R_i is to be changed to R_{i+1}. The goal is to avoid run-time network control operations and reduce the run-time overhead to that of *state loading operations* and *switching operations* (as defined in Section 2). The number of state loading operations will be equal to the number of realizable sets $R_1, R_2, \ldots$.

As was discussed in the previous section, time multiplexing of the network increases the size of the realizable sets, thus reducing their numbers. This in turns, reduces the number of state loading operations that need to be inserted in the program.

Optimal locality regions, $M_1^{opt}, M_2^{opt}, \ldots$ occur naturally in parallel programs, especially large scientific programs. They usually result from parallel "Doall" or "Doacross" loops. The size of each optimal locality set, however, may be larger than the size of the realizable sets of the network. In such cases increasing the sizes of the realizable sets to match the sizes of the locality sets may reduce drastically the number of state loading operations.

For instance, consider the following simple example that demonstrates an extreme case. Let $M_i^{opt} = \{c_1, c_2, c_3, c_4, c_5\}$, and assume that the network can support at most four connections at a time. That is the maximum size of a realizable set is 4. If the five connections are repeatedly used within a loop that executes 1000 times, then without time multiplexing, the network registers will have to be loaded twice within each execution of the loop to alternate between the two realizable sets $\{c_1, c_2, c_3, c_4\}$ and $\{c_2, c_3, c_4, c_5\}$ resulting in 2000 state loading operations. Using a two-way multiplexing will allow the five connections to be included in one realizable set, thus requiring only one state load operation.

Compiling the communication is very efficient whenever the communication patterns can be determined statically. However, provisions must be made to accommodate communication requests that are not known at compile time. Given that these requests represent only a very small percentage of the total communication, less efficient techniques may be applied to satisfy these requirements. A possible technique is to combine time-multiplexing of the network with routing through multiple traversals of the network. This will require routing information to be attached to messages whose destinations become known only at run-time. In order, however, to have a systematic algorithm to route each message to its destination, the network should sequence through configurations which emulate an efficient routing network. For example, if a hypercube interconnection may be emulated in the network through a sequence of *logn* configurations, then each message is guaranteed to reach its destination within *logn* steps. In this case, however, the routing information should be decoded after each step.

Whenever the compiler fails in determining the communication pattern in some section of an application, it inserts code to load the network registers with a sequence that emulates a routing network and sets up a *mode* flag to invoke a run-time routing routine. It also inserts code to reset the *mode* flag at the end of this section. This allows statically determined time multiplexed configuration sequences to be used for both static and dynamic communications within the same application.

6 Summary and Conclusions

The overhead of controlling all-optical interconnection networks can be greatly reduced by time multiplexing, especially when locality is manifested in the communication requirement of an application. In this paper, examples are given for the multiplexing of networks in which configurations are established by setting electro/optical switches. The same time multiplexing techniques may be equally applied to networks in which configurations are established by arbitrating passive optical couplers [6], assigning wavelengths to transmitters and receivers [4] or assigning transmitting slots on pipelined optical busses [9].

The time multiplexing approach is very suitable to applications in which most of the communication requirements are known at compile time. The majority of applications that require massive parallelism fall into this category. For these application, the communication requirements are divided into phases, and at the beginning of each phase, the network is set to sequence through configurations that satisfy all the communication requirements in that phase. In other words, the network is configured to satisfy the communication for that phase. This paradigm was called Reconfiguration with Time Division Multiplexing in [10].

So, why does it pay off to time multiplex optical networks?. First, without multiplexing, the control overhead of establishing all optical paths may be prohibitive. Second, applications exhibit communication locality which amplifies the benefits of multiplexing the network. Third the high bandwidth of optical paths allows for multiplexing the network with minimal effect on the transmission rate on individual connections. Finally, the nature of massively parallel programs allows compilers to determine, with current technology, the multiplexing sequences that satisfy most the communication requirements of a parallel program.

Acknowledgments

I would like to acknowledge that the formulation and analysis of the concepts discussed in this position paper are largely influenced by the results obtained in my joint work with Chunming Qiao, Donald Chiarulli, Steven Levitan, Greg Gravenstrater and Zincheng Guo. This work is in part supported by a grant from The Air Force Office of Scientific Research under contract F49620-93-1-0023DEF.

References

[1] F. Cappello and G. Germain, "Toward High Communication Performance through Compiled Communications on a Circuit Switched Interconnection Network" *Proc. of the First Int. Symp. on High-Performance Computer Architecture*, Raleigh, NC, 1995.

[2] D. Chiarulli, S. Levitan, R. Melhem and C. Qiao, "Locality Based Control Algorithms for Recomfigurable Optical Interconnection Networks", *Applied Optics,* vol 33, pp. 1528-1537, 1994.

[3] R. Cypher, A. Ho, S. Konstantinidou and P. Messina, "Architectural Requirements of Parallel Scientific Applications with Explicit Communication", *Proc. of the Int. Symp. on Computer Architecture*, San Diego, CA, 1993.

[4] P. Dowd, "Random Access Protocols for High-Speed Interprocessor Communications based on an Optical Passive Star Topology", *Journal of Lightwave Technology*, Vol 9, no. 6, pp 799-808, 1991.

[5] M Feldman, S Esener, C. Guest and S. Lee, "Comparison Between Optical and Electrical Interconnects Based on Power and Speed Considerations", *Applied Optics*, Vol. 29, pp. 1742-1751, 1990.

[6] G. Gravenstrater, R. Melhem, D. Chiarulli, S. Levitan and J. Taza, "The Partitioned Optical Passive Star Topology" *Proceedings of the 9th Int. Parallel Processing Symposium*, Santa Barbara, CA, 1994.

[7] H. Scott Hinton, "Photonic Switching Using Directional Couplers", *IEEE Communication Magazine*, Vol 25, no 5, pp 16-26, 1987.

[8] P. Lalwaney, L. Zenou, A. Ganz and I. Koren, "Optical Interconnects for Multiprocessors: Cost Performance Analysis" *Proceedings of the Symposium on Frontiers of Massively Parallel Computation* , pp. 278-285, 1992.

[9] R. Melhem, D. Chiarulli and S. Levitan, "Space Multiplexing of Waveguides in Optically Interconnected Multiprocessor Systems" *The Computer Journal,* Vol 32, no 4, pp. 362-369, 1989.

[10] C. Qiao and R. Melhem, "Reconfiguration with Time Division Multiplexed MINs for Multiprocessor Communications", *IEEE Transactions on Parallel and Distributed Systems*, Vol 5, no 4, pp. 337-352, 1994.

[11] C. Qiao and R. Melhem "Reducing Communication Latency with Path Multiplexing in Optically Interconnected Multiprocessor Systems", *Proc. of the First Int. Symp. on High-Performance Computer Architecture*, Raleigh, NC, 1995.

[12] C. Qiao, R. Melhem, D. Chiarulli and S. Levitan, "Dynamic Reconfiguration of Optically Interconnected Networks with Time-Division Multiplexing", *The Journal of Parallel and Distributed Computing*, Vol 22, no. 2, pp 268-278, 1994.

[13] R. Thompson, R. Anderson, J. Camlet and P. Giordano, "Experimental modular switching system with a time-multiplexed photonic center stage", *Proceedings of OSA Topical Meeting on Photonic Switching*, 1989.