Time-Division Optical Communications in Multiprocessor Arrays

Chunming Qiao, *Student Member, IEEE,* and Rami G. Melhem, *Member, IEEE*

***Abstract*—In this paper, we propose an optical communication structure for multiprocessor arrays which exploits the high communication bandwidth of optical waveguides. The structure takes advantage of two properties of optical signal transmissions on waveguides, namely unidirectional propagation and predictable propagation delays per unit length. Because of these two properties, time-division multiplexing of messages has the same effect as message pipelining on optical waveguides. Two time-division multiplexing approaches are proposed and the combination of the two is used in the design of the optical communication structure. Analysis and simulation results are given to evaluate the communication effectiveness of the system. A clock distribution method is also proposed to address potential synchronization problems. Finally, feasibility issues with current and future technologies are discussed.**

***Index Terms*—clock distribution, coincident pulse addressing, optical communications, pipelining, reservation scheme, time-division multiplexing, waveguide connections.**

I. Introduction and Background

OPTICAL interconnections for communication networks and multiprocessor systems including both free-space [3], [7], [9], [11], [14] and guided wave [12], [18], [29], [31], [37] approaches have been studied extensively in the literature. In this paper, we propose a waveguide interconnection system with time-division communication. Readers are referred to [8], [13], [35], [39] for other classes of interconnection systems with space-division, wavelength-division and hybrid (or multiple) communications.

Time-division communication is especially useful in systems with optical interconnections where high communication bandwidth can be exploited. Optical pulse transmissions on a single-mode waveguide have two distinct properties that are not shared by electronic signal transmissions, namely unidirectional propagation and predictable propagation delays per unit length. The second property states that the propagation delay is directly proportional to the length of a fiber. Various effects on the propagation delay are discussed in [5] and [30]. It has been found that the dominant effect is the change in temperature and that effects due to dispersion and polarization losses are negligible. The variability of the propagation delay in the fiber versus temperature is on the order of 40 ps/km-C° for a single-mode fiber at 1300 nm. This represents a very minor variation in effective optical path length.

In a multiprocessor system connected with an optical waveguide (or bus), relationships between the spatial and temporal positions of transmitted pulses can be established. For example, if two processors transmit a pulse on the waveguide at the same time, the difference between arrival times of these two pulses at any checkpoint downstream, is equal to the propagation delay between the two processors. In other words, the spatial separation of the two processors determines the temporal separation between the pulses they transmit. By arranging spatial separations and controlling transmission (or receiving) times of processors, time-division multiplexing (demultiplexing) can be done without using multiplexors (or demultiplexors).

Several time-division switching approaches can be applied in a multiprocessor system connected by optical buses. In the first approach, each processor is assigned a fixed time slot and transmits or receives a message during that particular time slot. A sequence of time slots formed on the transmitting segment of a bus is rearranged via a time-slot interchanger [26], [29], [34] and then forwarded to the receiving segment. Each time slot of the output sequence contains a message destined to the processor corresponding to that slot. In the second approach, each processor is assigned a fixed transmitting time slot. A sequence of time slots formed on the transmitting segment is directly forwarded to the receiving segment without interchanging the time slots. Instead of assigning a fixed receiving time slot to each processor, a *SIMD* environment is assumed where each processor knows which processor is sending a message to it and therefore knows the time slot that contains the message. Since there is a one-to-one mapping between a source processor and a time slot, we call this approach time-division source-oriented multiplexing (or TDSM). It has also been referred to as bus pipelining in [12] and [21].

TDSM may also be applied in a non-*SIMD* environment, where the source processors are not known to the destination processors. In this case, each message should contain address information so that each processor will be able to receive messages upon address decodings. Another approach, which is also applicable in a non-*SIMD* environment, assigns a fixed receiving time slot to each processor. Each message is transmitted during the receiving time slot assigned to its destination processor. Since there is a one-to-one mapping between a destination processor and a time slot, we call this approach time-division destination-oriented multiplexing

Manuscript received May 5, 1991; revised January 31, 1992. This work was supported in part by a grant from the Air Force Office of Scientific Research under Contract AFOSR-89-0469.

The authors are with the Department of Computer Science, University of Pittsburgh, Pittsburgh, PA 15260.

IEEE Log Number 9207202.

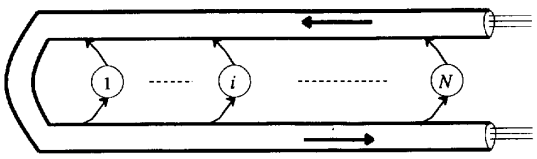


Fig. 1. A. linear optical bus system of N processors.

(or *TDDM*). Since each destination only has one dedicated receiving time-slot, contention can occur if several sources want to send messages to the same destination. One way to ensure exclusive access of a time-slot is to use a reservation scheme.

In this paper, we will discuss TDSM and TDDM approaches and apply the combination of these two in our system design. We use coincident pulse techniques [4], [18], [28] to encode address information that is contained in messages. In order to explain coincident pulse addressing, we consider a linear system of N processors which are connected by an optical bus as shown in Fig. 1. Each processor transmits on the upper half segment of a bus and receives from the lower half segment. The optical bus consists of three identical waveguides, one for carrying messages (the *message waveguide*) and two for carrying address information (the *reference waveguide* and the *select waveguide*). Messages are organized as *message frames,* which have a certain fixed length.

Let w be the pulse duration in seconds, and let c_b be the velocity of light in these waveguides. Define a unit delay to be the spatial length of a single optical pulse, that is $w \times c_b$. Initially, processors are connected to these three waveguides such that between any two given processors, the same length of fiber is used on all three waveguides. Hence, the propagation delays between two processors are the same for all three waveguides. We then add one unit delay, shown as a loop in Fig. 2, between any two processors on the receiving (lower half) segments of the reference waveguide and of the message waveguide. Each loop is an extra segment of a fiber and the amount of delay added can be accurately chosen based on the length of the segment [30]. As a result, the propagation delays on the receiving segments of the select waveguide and the reference waveguide are no longer the same. Fig. 2(a) shows the receiving segments of the select waveguide and the reference waveguide. For the purpose of simplicity, the transmitting segments of these waveguides, which do not have added delays, have been omitted from the figure. The message waveguide resembles the reference waveguide and is also omitted. Note that, in general, any amount of delay that is a multiple of the pulse width may be added between two adjacent processors. However, adding unnecessary amount of delay does increase the end to end propagation time on the waveguide.

A source processor sends a reference pulse and a select pulse at appropriate times, so that after these two pulses propagate through their corresponding waveguides, a coincidence of the two occurs at the desired destination. The source processor also sends a message frame which propagates synchronously with the reference pulse. Whenever a processor detects a coincidence of a reference pulse and a select pulse, it reads the message frame. More specifically, Let t_{ref} be the time when processor i transmits its reference pulse and $t_{sel}(j)$ be the time when it transmits a select pulse. These two pulses will coincide at processor j if and only if

$$t_{sel}(j) = t_{ref} + j \tag{1.1}$$

where $0 \leq i, j < N$. This means that for a given reference pulse transmitted at time t_{ref}, the presence of a select pulse at time $t_{ref} + j$ will address processor j while the absence of a select pulse at that time will not. In essence, the address of a destination processor is unary encoded by the source processor using the relative transmission time of a reference pulse and a select pulse.

Call the duration of each pulse, w, a pulse slot. A sequence of pulse slots on the select waveguide, each with either the presence or the absence of a select pulse relative to a given reference pulse, is called an *address frame.* Fig. 2(b) shows a snapshot of a reference pulse and an address frame just after they have been transmitted. At the transmission time, the select pulse is j units behind the reference pulse. Since the reference pulse will be delayed by one unit each time it propagates through a processor on the receiving segment of the reference waveguide, these two pulses will coincide at processor j.

Note that, in order for the above scheme to work, it is necessary to ensure that, between any two adjacent processors, propagation delay on the receiving segment of the reference waveguide is a unit delay longer than that of the select waveguide. It is not necessary, however, to have the same propagation delays between any two processors on either one of the waveguides. For the purpose of simplicity, however, we will assume that all processors are equally apart on any waveguide in terms of their spatial separations determined by the fiber length. The tolerance of the coincident pulse addressing mechanisms to minor variations in the fiber length, the diameter and other optical properties of all three waveguides depends on the transmission rate. Experiments have shown that at 250 MHz, synchronization errors up to one half of the pulse width can be tolerated among the reference and the select pulses without causing errors in the detection of coincident pulses [5].

With the above unary addressing, an address frame contains N pulse slots and is essentially a time-multiplexed sequence of pulse slots, each corresponding to a destination processor. As the address frame propagates through the receiving segment of the bus, demultiplexing of each pulse slot is performed with respect to a reference pulse via unit delays added on the reference waveguide. Address decoding, which could be a bottleneck with traditional addressing mechanisms, is done through the detection of a coincidence at the destination.

Define a *packet* to be a collection of information including a message frame, an address frame and a reference pulse. Let P be the length of a packet in time units and D be the spatial separation of any two adjacent processors on an optical bus. Because an address frame length is N time units, we have the condition $P \geq N$. With TDSM communication, if all processors transmit packets simultaneously, then the condition $D \geq P$ has to be satisfied in order to prevent packet

Fig. 2. Unary addressing using coincident pulse techniques. (a) The receiving segments of the reference and the select waveguides. (b) An address frame relative to a reference pulse.

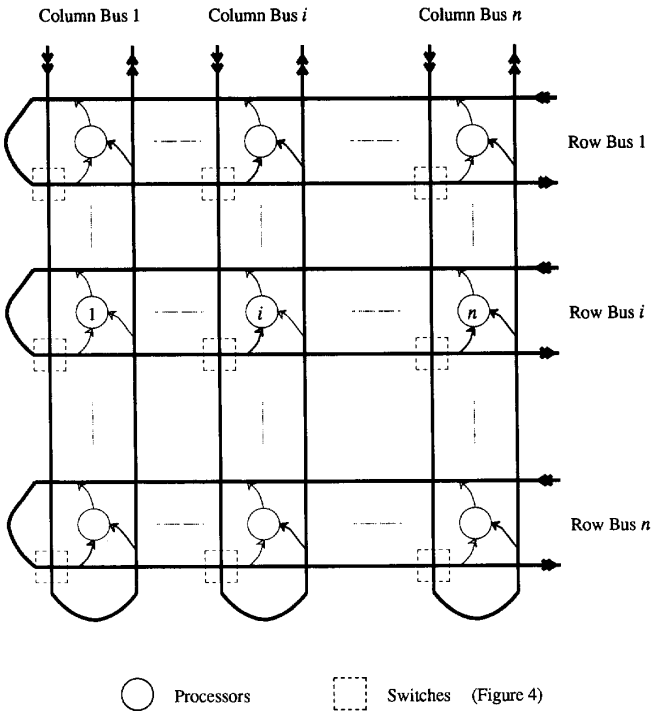


Fig. 3. System configuration of the *ASOS*.

overlappings. This condition, together with the condition that $P \geq N$, limits the system size N. In addition, the address frame length could be longer than the message frame length if N is large, resulting in inefficiency. Another factor that limits the system size relates to the power distribution. Specifically, the system size is limited by the minimum power that can be detected at the last processor in a linear system [5].

Because of these shortcomings in a linear system, we propose a two-dimensional $n \times n$ array called *ASOS* for *Array structure with Synchronous Optical Switches*. Note that, a synchronous optical hypergraph architecture proposed for computer network environments has been studied in [24]. Our communication structure, on the other hand, is proposed for multiprocessor communications. In Section II, we present the system configuration of the proposed *ASOS*. We also show how communication in the *ASOS* is done with TDDM, TDSM and the combination of the two. In Section III, we describe three reservation schemes for resolving time-slot contention in TDDM communication. In Section IV, we give analysis and simulation results that relate to communication effectiveness of the system. In Section V, we address the potential synchronization problem and propose a global clock distribution model as it relates to the packet size limitation. And finally in Section VI, we determine the merits of the design and conclude the paper.

II. Array Structure with Synchronous Optical Switches

A. System Configuration

The communication structure of an Array structure with Synchronous Optical Switches (or *ASOS*) is shown in Fig. 3. Processors shown may be just communication coprocessors that serve as interfaces between host processors and optical buses. These processors are connected with a set of folded horizontal (row) buses and vertical (column) buses, each consisting of three waveguides as in a linear system. All row buses are assumed to be identical, so are all column buses. During the course of the following presentation, rows are numbered from top to bottom and columns and processors are numbered from left to right.

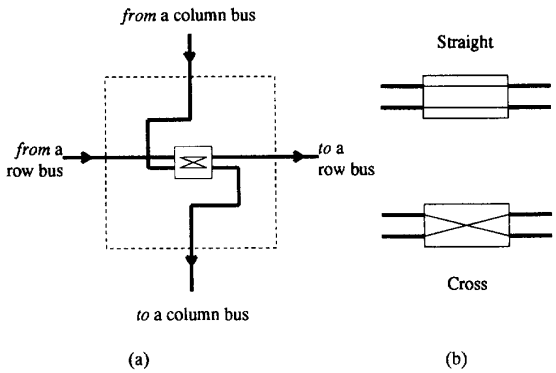


Fig. 4. Switch connections and its two states. (a) Switch connections. (b) Two switch states.

A processor can transmit on the upper segment of a row bus while receiving from the lower segment of a row bus and the right segment of a column bus concurrently. As shown in Fig. 3, an electro-optical switch is placed at each intersection of the lower segment of a row bus and the left segment of a column bus. The switch connects a row and a column bus as shown in Fig. 4(a). Each switch is a 2×2 electronically controlled optical device [1], [2], [32], which can be in one of the two states: straight or cross, as shown in Fig. 4(b). If a switch is in the straight state, a message packet propagating on a row bus will continue propagating on it. However, if a switch is in the cross state, a message packet propagating on a row bus will be switched over to a column bus. With the presence of all-optical data paths and electronic processors in the ASOS, conversions between optical signals and electronic signals by processors are needed. However, optical data can be transmitted on all-optical paths without intermediate conversions. By concurrenterizing optical data transmission and electronic signal processing, high bandwidth of optical paths may be attained.

A global clock is used for synchronization purposes and it is assumed that all processors and switches receive identical copies of the synchronization clock. How synchronizations can be retained without the above assumption is discussed in Section V. Note that host processors may execute at their own paces and submit messages to the communication structure, which delivers these messages under synchronized control.

B. Row And Column Communications

Communications between processors at the same row, which we call *row communications,* use each row bus for transmitting and receiving messages with switches set to the straight state. Since packets can not be transmitted directly on column buses, communications between processors at different rows, which we call *column communications,* use both row and column buses with switches connecting them set to the cross state. Row communications and column communications alternate, in what we call *row phases* and *column phases,* respectively. Switches alternate their two states accordingly. All switches are set to the same state simultaneously.

Let the optical path length between any two adjacent processors (or two adjacent switches) on a row bus be D units long. In order to allow simultaneous transmissions or switchings of packets without overlappings, we require $D \geq P$. We further let the folded optical path length on a row bus be D units (see Fig. 5). Defining T to be the end-to-end propagation delay of a row bus, we have $T = (2n-1)D$. Similarly, let the optical path length between two adjacent processors (or two adjacent switches) on a column bus be also D units long and the end-to-end propagation delay of a column bus be T units.

Time-division multiplexing in the *ASOS* can be best explained using the *train loading* model described below. Each time-slot is P units long and is called a *packet slot* (PS). Imagine that a train of n packet slots is originated on a row bus, as shown in Fig. 5. When the beginning of the train is at processor n, a communication phase begins. Packet slots in a train are numbered as $n, n-1, \cdots, 1$ from left to right. Two adjacent packet slots in a train are separated by D units. That is, there is a gap of $D-P$ units between two slots. We call a train originated at the beginning of a row phase or a column phase a *row train,* or a *column train,* respectively. Note that column buses, switches and taps are omitted from the figure. Further, a row or a column train should be regarded as three separate trains on the message, select and reference waveguides, respectively. Nevertheless, when there is no confusion, we will view these three trains as one on either a row or a column bus.

Assume that a train is originated at the beginning of a communication phase (at time t_0). Let $P\_Arr(i,p)$ be the time that the PS_i arrives at processor p on the upper segment of the bus. We have

$$\begin{aligned} P\_Arr(i,p) &= t_0 + (n-i)D + (n-p)D \\ &= t_0 + T - (i+p-1)D. \end{aligned} \tag{2.1}$$

The TDSM is used in a row phase. Each PS_i in a row train is assigned as a transmitting slot to processor i. Each processor can send out one packet during each row phase and each packet contains unary coincident pulse addressing information. More specifically, let t_0 be the time when a row phase begins. processor i transmits a packet, if it has one, at time $P\_Arr(i,i)$. That is, as the train propagates through the upper segment of a row bus, processor i loads PS_i with its packet.

One advantage of using TDSM is that a processor can receive more than one message frame in a single row phase, a capability which is usually referred to as m-to-1 communication. Another advantage with TDSM is that a processor can send out a message frame to several destinations in a single row phase efficiently, a capability which is usually referred to as multicasting. This is done by multiplexing several select pulses, each corresponding to one destination, in an address frame [18], [28].

Since a train is nD units long, the last packet slot of the train, namely PS_1, passes by processor n on the upper segment at the time $t_0 + nD$. At that time, the row phase ends and a column phase begins with a column train originated. In a column phase, TDDM is used. Each PS_i of a column train is assigned as a receiving slot to the ith switch at a row bus. That is, a packet transmitted during PS_i is to be switched to

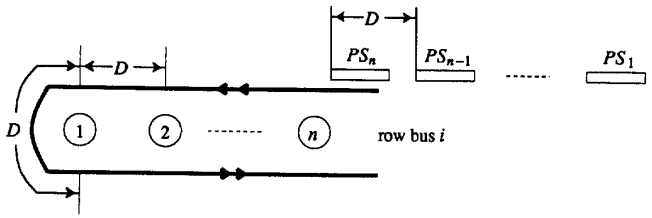


Fig. 5. A train of n packet slots when originated.

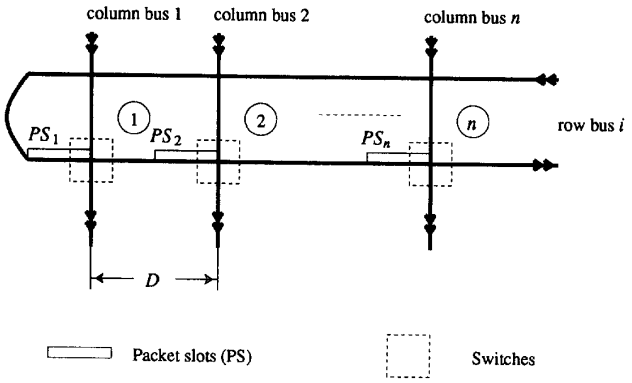


Fig. 6. Simultaneous switchings of packet slots.

column bus i. If more than one processor wants to send packets to the same column bus, packet slot contention will occur. Such contention is resolved by using packet slot reservation schemes, which will be described in Section III. For now, assume that reservations of every packet slot have been done and therefore only one processor will transmit a packet during any specific packet slot. Note however, that a processor could send several packets, each to a different column bus, if it has reserved the corresponding packet slots.

During a column phase, each processor loads packet slots that it has reserved while the train propagates through the upper segment of a row bus. More specifically, if processor p has reserved PS_i, it will transmit a packet which is to be switched to the column bus i at time $P\_Arr(i,p)$. T units after a column phase begins, each packet slot of a column train will arrive at its corresponding switch simultaneously as shown in Fig. 6. Every switch is set to the cross state for P units to switch a packet over to a column bus with which the destination processor of the packet is connected. That is, if a column phase begins at time t_1, then all switches will be in the cross state during the time period from $t_1 + T$ to $t_1 + T + P$. Note that, we have assumed a negligible switching time. In reality, the switching time ranges from a few hundreds of picoseconds to a few nanoseconds [32]. If S is the time needed for a switch to change from one state to the other, it is sufficient to let $D = P + S$ and let switches start switching to the cross state at the time $t_1 + T - S$. Each switch will stay in the cross state for only P units and start switching to the straight state at the time $t_1 + T + P$. Note that, in a non-SIMD environment, TDDM has the advantage of timely and orderly delivering messages to passive destinations, such as switches, without address information.

Before a column train is switched, a new row phase begins. A column phase ends as soon as the last packet slot of a column train, namely PS_1, passes by processor n on the upper segment. That is, a column phase also takes nD unit time. A row train will be originated and propagating on the upper segment while a column train is propagating on the lower segment. Note that the beginning of PS_n of the row train is also $D - P$ units away from the end of PS_1 of the preceding column train. Therefore, even with non-negligible switching time S, all switches can be set back to the straight state while the row train propagates on the lower segment of a bus.

When switches on a row bus are set to the cross state, switches at other row buses are also set to the cross state simultaneously. Since two adjacent switches on a column bus are D units apart, packet switched from different row buses will not overlap with each others on a column bus. Rather, these packets will form a train on the left segment of a column bus as shown in Fig. 7. As described earlier, a switch will stay in the cross state for only P units. After packets are switched onto a column bus, a switch is set to the straight state for the next T units until the next column train needs to be switched. By that time, however, all packets of the previous column train will have already passed through the left segment of the column bus. Therefore, no packets on any column bus will be switched back to a row bus even if switches will be set back again to the cross state.

Every packet contains an address frame encoded using coincident pulse techniques and a coincidence will occur at

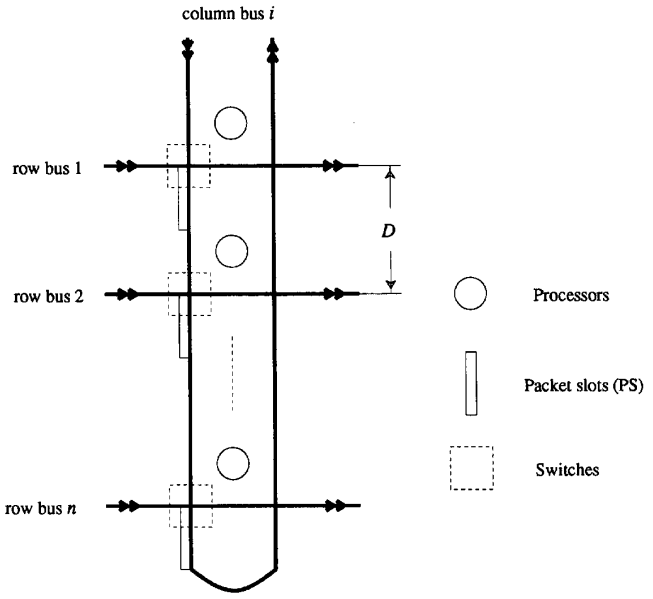


Fig. 7. A packet train formed on a column bus.

the destination processor, as will be described in the next subsection. In essence, TDDM is used to switch a packet onto a proper column bus and the destination processor on that column bus is then addressed using the coincident pulse techniques.

C. Coincident Pulse Addressing

As mentioned in the previous section, coincident pulse addressing is used in both row and column communications. Each row bus is similar to a bus used in a linear system [refer to Fig. 1 and Fig. 2(a)]. Each column bus can be viewed as a row bus rotated 90° degree anticlockwise. That is, on the right (receiving) segment of a column bus, one unit delay is added between any two adjacent processors on the message waveguide and the reference waveguide.

A packet in a row train propagates only on a row bus, that is, its reference pulse will encounter added delays only on the row bus. To cause a coincidence of the reference pulse and a select pulse at processor j of that row bus, the relative transmission times of the two pulses should satisfy the same equation as (1.1). A packet in a column train, however, propagates on a row bus and then a column bus. A reference pulse of such a packet will encounter additional unit delays on the right half segment of that column bus before arriving at a destination processor. More specifically, if a packet in a column train is destined to a processor at row i and column j, its reference pulse will first encounter j added unit delays on a row bus and then $n - i$ added unit delays on a column bus. Therefore, to cause a coincidence of the reference pulse and a select pulse at that destination, the relative transmission times of these two pulses should satisfy the following equation:

$$t_{sel} = t_{ref} + j + (n - i). \tag{2.2}$$

Since we have $1 \leq j + (n - i) \leq 2n - 1$, the length of an address frame should be $2n - 1$ units long and therefore $D \geq P \geq 2n - 1$. However, according to (1.1), we have $1 \leq t_{sel} - t_{ref} \leq n$. That is, in row communications, address information can be fit into n pulse slots. Therefore, any address frame of a row train will have $n - 1$ unused pulse slots.

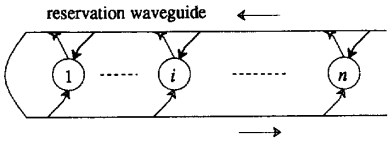


Fig. 8. A reservation waveguide in the *ASOS*.

III. Packet Slot Reservation Scheme

As mentioned earlier, when loading a column train using TDDM, reservations are required to resolve packet slot contention. Reservations can be made concurrently with message transmission using separate folded waveguides, called *reservation waveguides,* one at each row, as shown in Fig. 8. Each processor transmits on the upper segment and receives from either the upper or the lower segment of a reservation waveguide. It is also assumed that the optical path length between any two adjacent processors on each reservation waveguide is D units and the end-to-end propagation delay of each waveguide is T units.

There are n packet slots to be reserved for each column train before the train is loaded in a column phase. To reserve packet slots of a column train, a corresponding train, which we call a *reservation train,* is originated on a reservation waveguide. The reservation train also consists of n packet slots with the same separations between two slots as in a column train. Each of the packet slots in a reservation train is used to arbitrate

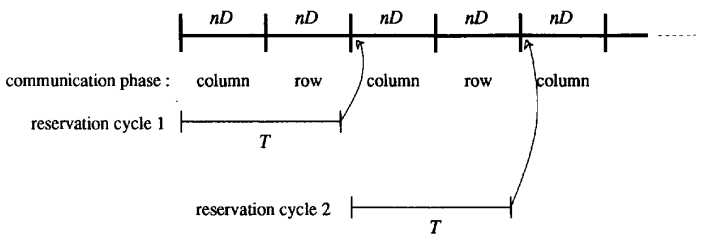


Fig. 9. Scheduling with the linear priority scheme.

the reservation of the corresponding packet slot in a column train. The time period from the origination of a reservation train to the end of reservation operations on its n packet slots is called a *reservation cycle.*

A. Linear Priority Scheme

The simplest reservation scheme is the linear priority scheme. Each processor at a row is assigned a unique priority for all packet slots. When competing for a reservation of a packet slot, the processor that has the highest priority among all competing processors will succeed while other will fail. This scheme can be implemented as follows. Processors upstream are assigned with higher priorities than processors downstream. For example, processor n is assigned with the highest priority and processor 1 with the lowest priority. After a reservation cycle begins, each processor monitors packet slots of a reservation train propagating on the upper segment of a reservation waveguide from right to left. If processor p wants to reserve packet slot i, it transmits pulses to the left while also detecting pulses coming from the right during that pulse slot. In other words, the processor transmits while also detecting pulses during the time period from $P\_Arr(i,p)$ to $P\_Arr(i,p) + P$. If no pulses are detected in that period, processor p has succeeded in making a reservation for PS_i since no processors upstream have attempted to reserve it. Otherwise, processor p fails and a processor upstream has succeeded in reserving PS_i.

Reservation operations on n packet slots will finish as soon as the train leaves the upper segment. In other words, a reservation cycle takes $P\_Arr(1,1) + D$ or T units. One way to schedule reservation cycles is shown in Fig. 9, where each reservation cycle starts $2nD$ units earlier than its corresponding column phase. In other words, a reservation train is originated $2nD$ units before the corresponding column train is originated.

However, it is important to note that two alternative schedulings are also possible. The first is to originate each reservation train only D units before its corresponding column train is originated. By the time that PS_i of the column train arrives at any processor, PS_i of the reservation train has already left the processor and therefore the processor knows if it has a reservation for PS_i or not. The second alternative is to originate another reservation train as soon as the previous reservation train leaves processor n, or equivalently, nD units after the previous train is originated. That is, reservation trains are pipelined on a reservation waveguide every nD units. Nevertheless, we only need one reservation train every $2nD$ units because reservations are not required in a row phase.

B. Restrained Linear Priority Scheme

The problem of the simple linear priority scheme is the possibility of starvations. Processors downstream with lower priorities may be indefinitely blocked because some higher priority processors at the right keep making reservations. This scheme is extremely unfair to those lower priority processors since they have much less chances to succeed in column communications than those higher priority ones.

A reservation cycle is called an *idle cycle* for a packet slot if no processor has made a reservation for that packet slot in the cycle. In the restrained linear priority scheme, linear priorities among processors are still enforced but we require a processor that succeeded in reserving a packet slot in a previous cycle to refrain from making another reservation for the same packet slot until after an idle cycle is detected for that packet slot. Since no processor can make reservations for a packet slot twice without an idle cycle in between, and there are only n processors competing at each row, any processor would be able to succeed in making a reservation for a packet slot within n cycles and no starvations are therefore possible.

An implementation of this scheme is very similar to that of the linear priority scheme except that each processor also detects an idle cycle for each packet slot on the lower segment of a reservation waveguide. More specifically, if processor p wants to reserve packet slot i, it transmits while detecting pulses on the upper segment of a reservation waveguide during the time period from $P\_Arr(i,p)$ to $P\_Arr(i,p) + P$. Once it has succeeded in reserving the packet slot, the processor raises an internal flag which indicates that it will have to wait for an idle cycle before it can make another reservation for PS_i. Processor p can detect an idle cycle of packet slot i by monitoring the lower segment during the time period from $P\_Arr(i,p) + (2p-1)D$ to $P\_Arr(i,p) + (2p-1)D + P$.

One way to schedule reservation cycles using this scheme is shown in Fig. 10. As in the first scheme, a processor knows if it has succeeded in reserving a packet slot as soon as the packet slot passes by it on the upper segment. Therefore, the first scheduling alternative mentioned before is also applicable. That is, a reservation cycle could start only D units earlier than the corresponding column cycle. However, the second alternative mentioned is not applicable in this scheme. This is because a processor must know if the previous reservation cycle is an idle cycle of a packet slot before it could attempt to

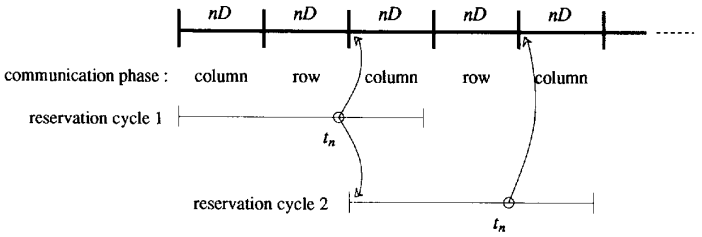


Fig. 10. Scheduling with the restrained linear priority scheme.

make a reservation for that packet slot. Nevertheless, $T + D$ or equivalently, $2nD$, units after a reservation starts, PS_n of a reservation train will pass by processor n on the lower segment of a reservation waveguide. Only at that time, denoted as t_n as in Fig. 10, processor n will know if the cycle is an idle cycle of PS_n and the next reservation cycle could then begin.

Although no starvations are possible in this scheme, a higher priority processor still has more chances of success than lower priority processors when competing for reservations. Assume that after an idle cycle, processor n, which has the highest priority, succeeds in making a reservation for a packet slot. Further assume that it is the only processor which wants to make reservations for that packet slot in the next cycle and thus the second cycle becomes an idle cycle. In the third reservation cycle, processor n will still succeed in making a reservation for the same packet slot, regardless of the fact that other processors which have not had their chances yet may also want to make reservations for the same packet slot. Note however, that if each processor wants to reserve a packet slot in every cycle, then each processor will succeed once before an idle cycle is forced. That is, under heavy loads, each processor will have equal chances of successful reservations using this scheme. In this case, this scheme is equivalent to a round-robin scheme with an idle cycle added every n cycles.

C. Round-Robin Scheme

The third scheme fully employs round-robin (or cyclic polling [16]) and requires a more complicated implementation. Unlike in the previous two schemes, each processor has a priority for each packet slot and that priority may change after each reservation cycle. The processor currently with the highest priority for a packet slot among all competing processors will succeed in making a reservation. In the next reservation cycle, it will become the lowest priority one when competing for the same packet slot. Meanwhile, the processor with the priority next to the one succeeded in the last cycle will have the highest priority and all other processors will adjust their priorities accordingly in a cyclic order. There will be an idle cycle for a packet slot only if no processors want to make a reservation, in which case, no priority changes will occur. As an example, assume that at the beginning of a cycle, processors, from the highest priority to the lowest for a packet slot, are $1, \cdots, i, i+1, \cdots, n$. If no processors want to reserve the packet slot, this cycle becomes an idle cycle and no priority changes will occur. Otherwise, assume processor i wants to reserve the slot, but none of processors from 1 through $i-1$ does. The processor i succeeds and processors, from the highest priority to the lowest in the next cycle, will be: $i+1, \cdots, n, 1, \cdots, i$.

As in the previous two schemes, a reservation train is originated at the beginning of each reservation cycle. However, to attempt to make a reservation for packet slot i, a processor no longer loads the entire PS_i. Instead, only the first n pulse slots of that packet slot are of interest. Assume these pulse slots are numbered as $1, 2, \cdots, n$ from the beginning to the end of a packet slot. Pulse slot p is assigned as the transmitting time-slot for processor p. That is, processor p only transmits a single pulse into the pth pulse slot of PS_i. More specifically, processor p transmits a pulse at the time $P\_Arr(i,p) + p$ if it wants to reserve packet slot i. In addition, each processor no longer monitors packet slots on the upper segment of a reservation waveguide. Instead, it detects every pulse slot of every packet slot on the lower segment of the waveguide. A presence (or an absence) of the jth pulse in packet slot i, which can be detected by processor p at the time $P\_Arr(i,p) + (2p-1)D + j$, indicates that processor j has (or has not) attempted to make a reservation for packet slot i. By time $P\_Arr(i,p)+(2p-1)D+D$ or $P\_Arr(i,p)+2pD$, processor p can decide which processor should succeed according to the round-robin scheme. In other words, each processor can obtain global information about reservation attempts made for every packet slot by all processors. Therefore, the decision concerning which processor should succeed in making a reservation for a packet slot according to the round-robin scheme can be made by each processor in a distributive way.

One way to schedule reservation cycles can be the same as what have already been shown in Fig. 10. Note however, that neither the idle cycle information of PS_n, nor its reservation result, will be available to processor n until after the time $P\_Arr(n,n) + 2nD$, or t_n, as defined above.

D. Reservations Without Additional Waveguides

Finally we note that there are two alternatives to the use of separate reservation waveguides for concurrent reservations and communications. Both alternatives eliminate the need for separate reservation waveguides by using one of the three waveguides in a row bus. The first alternative is to serialize

row phases, column phases and reservation cycles. Reservation can then be done using one of the three existing waveguides. A reservation cycle can be scheduled appropriately depending on the reservation scheme used. For example, if the first two schemes are used, it is sufficient to have each reservation train followed by a row train and a column train that corresponds to the reservation train. Further, a reservation train can be made as short as n pulse slots, each of which is used to reserve a packet slot of the column train.

The second alternative is to take advantage of the unused pulse slots available in address frames of row trains. As described in Section II-C, an address frame in each packet of a row train has $2n-1$ units, of which $n-1$ units are unused. More specifically, pulse slots numbered from $n+1$ to $2n-1$ in any address frame of a row train are not useful for coincident pulse addressing. These pulse slots in each of these address frames can be treated as a reservation packet. That is, reservation can be done on the select waveguide by incorporating each packet slot of a reservation train into an address frame of a row train. In other words, a reservation train can be embedded in a row train and a reservation cycle can overlap with a row cycle. Appropriate scheduling of reservation cycles can be achieved to ensure continuous row and column phases in similar ways as described in Section III-A through III-C.

IV. Analysis and Simulations

A. System Bandwidth

Let $B_{\max}$ be the maximum transmission rate at which a processor can drive an optical bus. Then the maximum bandwidth of a row bus is $B_{\max}$ and the maximum bandwidth of the $n \times n$ ASOS is $nB_{\max}$. Since during each packet slot which is $D = P + S$ units, at most P unit messages are transmitted. Therefore, the maximum efficiency, Θ, is

$$\Theta = \frac{P}{P+S}. \tag{3.1a}$$

And the maximum bandwidth achievable, B_a, is

$$B_a = n \times B_{\max} \times \Theta. \tag{3.1b}$$

Assume that on average, each processor generates L_r packets during each row phase and L_c packets during each column phase, where $0 \le L_r, L_c \le 1$. Denote the average number of packets generated per packet slot by L_s. Then $L_s = (L_r + L_c)/2$. The average throughput, or effective bandwith in a row phase, B_r, is

$$B_r = L_r \times B_a. \tag{3.2}$$

If we assume that the destinations of generated packets are uniformly distributed among n column buses, the *maximum* effective bandwidth in a column phase, B_c, may be approximated by

$$B_c = L_c \times B_a. \tag{3.3}$$

By taking the average of (3.2) and Eq. (3.3), the effective system bandwidth of an ASOS, denoted by B_e, is

$$B_e = \frac{B_r + B_c}{2} = \frac{nB_{\max}P(L_r + L_c)}{2(P+S)}. \tag{3.4}$$

Note that, the maximum bandwidth in a column phase given in (3.3) may not be achievable due to packet slot contentions. Therefore, given fixed average communication load L_s, mappings that distribute more loads to row communications and less loads to column communications will improve the effective system bandwidth. On the other hand, one may schedule more column phases than row phases to increase column communication bandwidth when balancing bandwidth among row and column phases is more important. Note that in the ASOS, column communication does not involve message relays at intermediate processors. In other words, no costly conversions between optical and electronic data signals are needed in column communication. As a result, column communications may be as nearly efficient as row communications. Hence, mappings are not as crucial to the system performance as in [12], where message relays may be required.

B. Packet Delay

Packet slot contention in column communications not only can decrease the effective bandwidth but can introduce delay for packets. In a column phase, a processor will not be able to send out a packet unless it has succeeded in reserving the corresponding packet slot. The packet that cannot be sent out due to an unsuccessful reservation has to be delayed until a future column phase. Define packet delay to be the number of column phases that a packet is delayed due to unsuccessful reservations of the corresponding packet slot. Further, assume that during each column phase, the number of packets that each processor generates is a poison process with mean rate λ where $\lambda < 1$. The destinations of these packets are evenly distributed among n columns. That is, on the average, out of λ packets generated by a processor in each column phase, only $\lambda_i = (\lambda/n)$ packets are destined for column bus i.

We first examine the average packet delay using the round-robin reservation scheme. A natural analytic model to use is the model with multiple queues and a single server. Each processor is viewed as a station with an ideally infinite queue. The server selects a station to serve in the round-robin fashion. The service time is a constant unit time (which is one column phase). In [33], a similar model is analyzed where the *reply interval,* defined as the time for a server to switch from one station to another, is assumed to be nonzero. For models with zero reply interval such as ours, only approximate analysis are given [17]. However, if we view the above model with multiple queues as a single queue, M/D/1 model with the *FCFS* (First-Come First-Served) policy, we will have the same average packet delay in both models. In fact, with the same assumptions about message arrival rate and service rate in both models, the statistical characteristics of the unfinished work in both models should be identical. In other words, the total number of packets remaining, hence the queue length in both models, is independent of the service policy used.

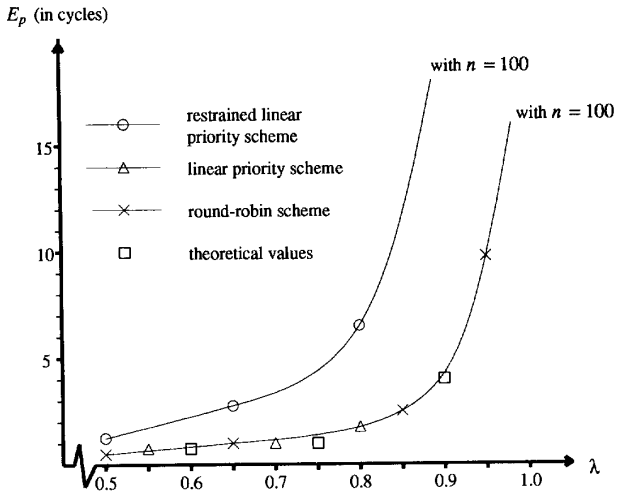


Fig. 11. Average packet delay versus arrival rate.

It follows from the Little's law [19] that the average packet delay, denoted by E_p, is also independent of the policy used. That is, the average packet delay using the round-robin scheme should be the same as that of a M/D/1 model. Namely [6], [15]

$$E_p = \frac{n\lambda_i}{2(1-n\lambda_i)} = \frac{\lambda}{2(1-\lambda)}. \tag{3.5}$$

It is also straightforward to have the same analysis result for the average packet delay using the linear priority scheme. However, we are unable to analyze the average packet delay using the restrained linear priority scheme. Fig. 11 shows simulation results of average packet delays for all three different reservation schemes when $n = 100$. We note that the results shown for both the round-robin and the linear priority schemes conform to the theoretical values as outlined in (3.5). However, the restrained linear priority scheme has longer average packet delay since some idle cycles are artificially introduced in this scheme.

If either the linear priority or the round-robin scheme is used, column communications will have an average packet delay of two (column phases) when the load reaches 80% of the capacity (this is equivalent to the case when $\lambda = 0.8$). Note however that according to the definition of the packet delay, row communication have zero packet delay.

C. Fairness of a Reservation Scheme

For each processor i, define its expected response time, denoted by $E_r[i]$, to be the average number of column phases that the processor has to delay transmissions due to its unsuccessful reservations. For example, in the linear priority scheme, processor n will always succeed in reservations and therefore its expected response time is 0 (column phase). On the other hand, processor 1 may, theoretically, have an infinite expected response time. Let E_r be the average expected response time of the n processors that are in the same row. Define the standard deviation of processors' expected response time, SD_r, to be the deviations of every $E_r[i]$ from E_r for $1 \leq i \leq n$. Different reservation schemes will yield different SD_r values. We use SD_r as a measure of fairness of a reservation scheme. A small value of SD_r indicates that the reservation scheme used is fair and a large value of SD_r indicates otherwise. Fig. 12 shows simulation results of SD_r for the three schemes discussed above, assuming $n = 100$. As expected, the linear priority scheme has the largest SD_r value and is regarded as being the least fair. In the restrained linear priority scheme however, the difference between the expected response time of any two processors could never exceed n. Therefore its SD_r value is smaller than that of linear priority scheme. The round-robin scheme has the least SD_r value and is regarded as being the most fair.

The restrained linear priority scheme may be preferred for its simplicity of implementation and reasonable fairness. The round-robin scheme requires extensive processing although it results in the least packet delay and is considered to be the most fair. According to the characteristics of the ASOS, several mapping strategies can be developed in conjunction with reservation schemes. In general, higher bandwidth and lower packet delay can be achieved if frequently communicated nodes can be mapped to same rows rather than different rows. After a mapping is done, if communication with nodes that are mapped to other rows is evenly distributed among nodes at the same row, we can use a fair reservation scheme. On the other hand, a less fair reservation scheme may be used if some nodes will communicate with nodes at different rows more frequently than others do. Higher priorities may be given to these nodes for making reservations in order to eliminate communication bottlenecks.

V. Clock Distribution and Packet Size Limitation

One of the major concerns in designing a large system is the synchronization problem. So far, we have assumed that all processors and switches are connected to a global clock with separate waveguides (called clock waveguides) of equal length. Therefore all processors and switches virtually share

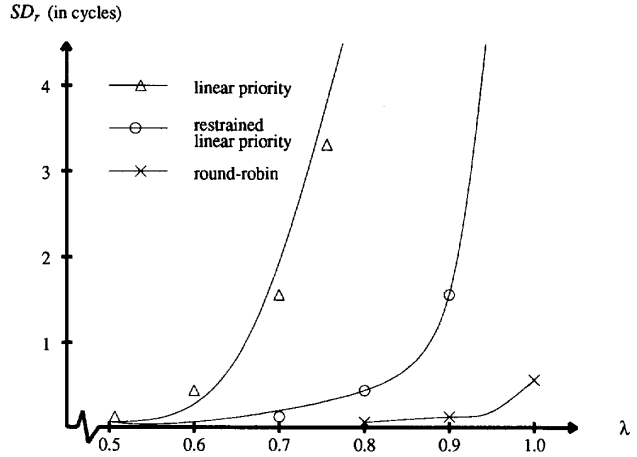


Fig. 12. Standard deviation of expected response time versus arrival rate.

Fig. 13. Two clock distribution models. (a) The identical time model. (b) The skewed time model.

an identical global time. More specifically, let T_g be a global time, and let $PT_L(r,c)$ and $ST_L(r,c)$ be a local time of the processor and the switch, respectively, at row r and column c. We have assumed that at any instant, $PT_L(r,c) = ST_L(r,c) = T_g + C$ for all r and c, where C is a constant. We call such a clock distribution model *identical-time* model.

For example, consider a system with two nodes, denoted by n_1 and n_2. Let $T_L(1)$ or $T_L(2)$ be their local time, respectively. To perform an operation *simultaneously* at a given instant T_g, each node starts the operation when its local time equals T_g. Here, an operation could be the transmission of a packet if the nodes are processors, or could be the change from one state to the other if the nodes are switches. In the identical-time model shown in Fig. 13(a), two separate clock waveguides with equal length are used to connect two nodes with the global clock. Since $T_L(1)$ is always identical to $T_L(2)$, the two nodes will start at the same time to perform a simultaneous operation on two packets. If these two nodes are separated by D units on a waveguide and a packet has a length of P units, the condition $P \leq D$ is necessary to prevent the two nodes from performing the same operation on the same packet. For instance, if these two nodes are processors, the above condition prevents overlappings of transmitted packets. If these two nodes are switches, the above condition prevents a packet from being switched (partially) by two switches.

Another model, which we call *skewed-time* model, is shown in Fig. 13(b). Only one clock waveguide is used to connect these two nodes to the global clock. In Fig. 13(b), it is assumed

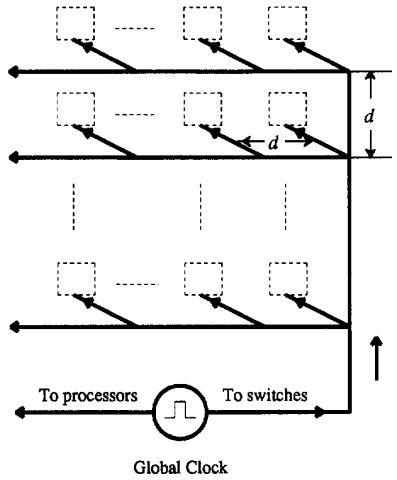


Fig. 14. Distributing the global clock to switches with skewed-time.

that clock pulses and packets propagate in the opposite directions. Assume that the propagation delay between n_1 and n_2 on the clock waveguide are d units, we have $T_L(1) = T_L(2) + d$. If the two nodes are to perform an operation simultaneously according to their local times, then the node n_1 will start d units earlier than the node n_2. Therefore, these two nodes will not perform the operation on the same packet as long as $P \leq D + d$. That is, given a fixed D, the packet size can be

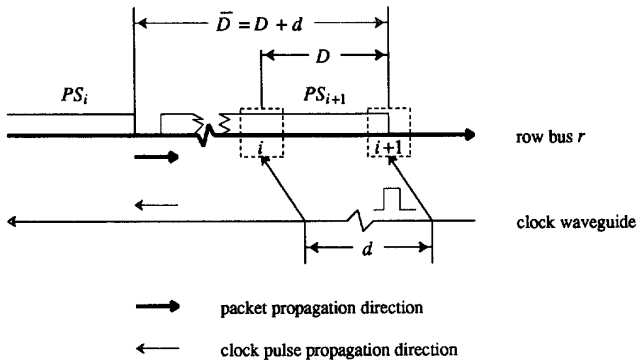


Fig. 15. Switching packets using skewed-time.

increased by d units using the skewed-time model. Note that, if clock pulses and packets propagate in the same direction, the packet size will have to be decreased. We are interested in increasing the packet size for a given D to overcome current technology restraints on the pulse width.

In order to have a regular connection between switches and the global clock, the skewed-time model is used to distribute global clock to switches in ASOS, as shown in Fig. 14. Clock pulses always propagate in the direction opposite to packets propagation. For example, packets propagate on the lower segment of a row bus from left to right. But clock pulses propagate from right to left on any row r, that is, at any instant,

$$ST_L(r,i) = ST_L(r,i-1) + d. \tag{5.1a}$$

Fig. 15 illustrates how packets are switched using the skewed-time model. The snapshot in the figure shows that switch $i+1$ has just been set to the cross state. The beginning of PS_i is $D+d$ units away from that of PS_{i+1} and d units away from switch i. After d units, switch i will be set to the cross state according to (5.1a). At the same time, PS_i will arrive at switch i just as desired. In addition, switched packets propagate top down on the left segment of a column bus but since clock pulses propagate bottom up at each column, we have

$$ST_L(r,c) = ST_L(r-1,c) + d. \tag{5.1b}$$

Similarly, we can show that no packet overlappings are possible on any column bus.

Fig. 16 shows a way to connect all processors to the same global clock using the skewed-time model. Note that, processors transmit their packets on the upper segment of row buses. These packets propagate from right to left at each row and therefore clock pulses propagate from left to right at each row as shown in the figure.

It is possible to recalculate (2.1) for the skewed-time model in terms of the local time of each processor. Specifically, let $\overline{D} = D + d$ and assume that a packet slot is $\overline{D}$ units. A row or a column phase is $n\overline{D}$ units since a train is $n\overline{D}$ units long. Assume that a train is originated when the local time of processor n at row r, namely $PT_L(r,n) = t_0$. In terms of the local time of processor p at the same row, the train is originated

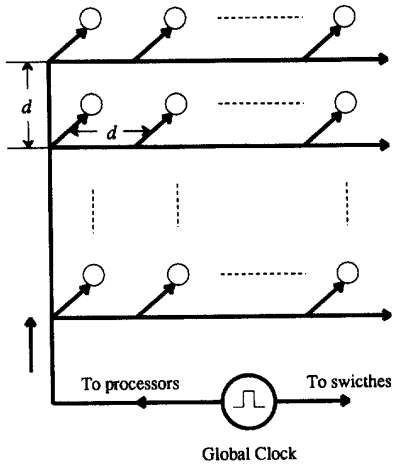


Fig. 16. Distributing the global clock to processors with skewed-time.

at time $PT_L(r,p) = PT_L(r,n) + (n-p)d$.Therefore,

$$\begin{aligned} P\_Arr(i,p) &= [t_0 + (n-p)d] + (n-i)\overline{D} + (n-p)D \\ &= t_0 + (n-p-i)\overline{D}. \end{aligned} \tag{5.2}$$

Note that, by replacing D in (2.1b) with $\overline{D}$, we can also derive (5.2). With this equation, processors can load or monitor packet slots as described in Section II. Reservations can also be made as described in Section III.

VI. Concluding Remarks

Usually, interconnection networks in multiprocessor systems provide limited connectivities between processors. Interprocessor communication in such systems may be enhanced by using the high bandwidth of optical interconnections. Communication with time-division multiplexing may emerge as a general solution to such a problem. On the one hand, TDM communication may help closing the gap between the speed of electronic devices and that of optical communication paths. On the other hand, the properties of optical signal transmission facilitate TDM communication in optical interconnected systems. As a result, the complexity of both

hardware and control in interconnection networks may be reduced and high bandwidth may be achieved with limited connectivities. This paper describes the application of the general TDM communication scheme to a two-dimensional spanning bus structure. Other applications of the scheme to multistage interconnection networks have been studied in [27] and [36].

It has been established that much higher bandwidth can be achieved with pipelined optical bus communication than with exclusive electronic bus access communication [12], [21]. The high bandwidth of optical waveguides can also be achieved using two time-division approaches, namely TDSM and TDDM, which are used in the design of the *ASOS*.

In addition to achieving high bandwidth, several other design goals have been met. For instance, the first-order crosstalk at the optical switches [25] is reduced since switches of the upper triangle ASOS array have only one input and output port active at any time. Moreover, the simplicity of the switch structures and control results in a low probability of switch failures. In other words, the proposed communication structure may be a robust component of a multiprocessor system.

Given available technologies, the suggested design is feasible and flexible. For example, with current technology, it is possible to drive an optical bus at 20 GHz [31]. This results in a pulse width w, or a time unit as defined in this paper, of 50 ps. Assuming that the speed of light in the waveguide is $c_g = 2 \times 10^8$ m/s, the spatial length of a pulse is 1 cm. With 100 ps switching speed [31], [32] and a 16-bit message frame, the switching time S is 2 time units and the packet length P is 16 time units in an 8×8 ASOS. Therefore, as long as the spatial separation of two processors, D, satisfies the condition $\overline{D} = D + d \geq P + S = 18$, packets can be pipelined without overlappings. That is, the design will apply to systems having spatial separation larger than 18 cm. If communication loads are 80% of the communication capacity in both row and column phases, that is, $L_r = L_c = 0.8$, the maximum effective bandwidth for the 8×8 ASOS is about 113.8 Gb/s according to (3.4), given the above P and S.

Note that, if a system has smaller spatial separation, for example, $D = 7$ cm, the above condition can also be satisfied by using the skewed time model with the skewing distance d of at least 11 time units. This 7 cm spatial separation may be appropriate for backplane connections or for intermodule connections, in which each module may contain several processors. In addition, when technology advances to the stage at which optical buses are implemented on GaAs wafers with 100 GHz transmission and at 10 ps switching speed, the skewed-time model may be no longer necessary. For example, with $D = 7$ cm, a packet could contain up to 34 bits without using the skewed-time model. Further advance in technology is expected to increase both the transmission and the switch speed, thus reducing the required minimum spatial separation to a point at which our techniques are suitable for chip-to-chip connections or connections within wafers. In such cases, the use of optical interconnections may still be advantageous when high frequency transmission is achieved [9], [38].

We note that our reservation schemes that relate to TDDM are different from any of the those schemes in [10]. Uses of random access schemes, such as those in [20] and [22], for resolving time-slot contentions would yield longer average packet delay and lower system bandwidth in this particular application in which the communication load is assumed to be moderately high. We note that the fairness issue, as we have defined, has not received much attention in the literatures and is important in eliminating communication bottlenecks. Finally, we note that some technology issues not mentioned in this paper, such as pulse generations, coincidence detections and power distributions, are discussed in [5] and [23].

REFERENCES

[1] R. Alferness, L. Buhl, S. Korotky, and R. Tucker, "High-speed $\Delta\beta$-reversal directional coupler switch," in *Top. Meeting Photon. Switching, Tech. Dig. Series*, vol. 13, 1987, pp. 77–78.

[2] A. Benner, H. Jordan, and V. Heuring, "Optically switched lithium niobate directional couplers for digital optical computing," *SPIE Proc., Digital Optical Computing II*, vol. 1215, pp. 343–352, 1990.

[3] K. Brenner, A. Huang, and N. Streibl, "Digital optical computing with symbolic substitution," *Appl. Opt.*, vol. 25, p. 3054, 1986.

[4] D. Chiarulli, R. Melhem, and S. Levitan, "Using coincident optical pulses for parallel memory addressing," *IEEE Comput. Mag.*, vol. 20, no. 12, pp. 48–58, 1987.

[5] D. Chiarulli, R. Ditmore, R. Melhem, and S. Levitan, "An all optical addressing circuit: Experimental results and scalability analysis," *J. Lightwave Technol.*, vol. 9, no. 12, pp. 1717–1725, Dec. 1991.

[6] J. Cohen, *The Single Server Queues*. Amsterdam, The Netherlands: North Holland, 1969.

[7] A. Dickinson and M. Prise, "Free-space optical interconnection scheme," in *Top. Meeting Photon. Switch*, Salt Lake City, UT, Feb. 1989.

[8] P. Dowd, "High performance interprocessor communication through optical wavelength division multiple access channels," in *Proc. 18th Int. Symp. Comput. Architecture*, May 1991, pp. 96–105.

[9] M. Feldman, S. Esener, C. Guest, and S. Lee, "Comparison between optical and electrical interconnects based on power and speed considerations," *Appl. Opt.*, vol. 27, pp. 1742–1751, 1988.

[10] M. Fine and F. Tobagi, "Demand assignment multiple access schemes in broadcast bus local area networks," *IEEE Trans. Comput.*, vol. C-33, no. 12, pp. 1130–1159, Dec. 1984.

[11] J. Goodman, F. Leonberger, S. Kung, and R. Athale, "Optical interconnections for VLSI systems," *Proc. IEEE*, vol. 72, pp. 850–866, 1984.

[12] Z. Guo, R. Melhem, R. Hall, D. Chiarulli, and S. Levitan, "Array processors with pipelined optical busses," *J. Parallel Distributed Comput.*, vol. 12, no. 3, pp. 269–282, 1991.

[13] K. Huang and J. Pier,. "A study of 3-D optical multistage interconnection networks," *SPIE Proc. Digital Opt. Comput. II*, vol. 1215, pp. 143–154, 1990.

[14] J. Jahns and M. Murdocca, "Crossover networks and their optical implementation," *Appl. Opt.*, vol. 27, pp. 3155–3160, 1988.

[15] L. Kleinrock, *Queueing Systems, Vol. 1: Theory*. New York: Wiley, 1975.

[16] H. Kobayashi and A. Konheim, "Queuing models for computer communications systems analysis," *IEEE Trans. Commun.*, vol. COM-25, no. 1, pp. 2–29, Jan. 1977.

[17] A. Konheim, "Chaining in a loop system," *IEEE Trans. Commun.*, vol. COM-24, no. 2, pp. 203–210, Feb. 1976.

[18] S. Levitan, D. Chiarulli, and R. Melhem, "Coincident pulse techniques for multiprocessor interconnection structures," *Appl. Opt.*, vol. 29, no. 14, pp. 2024–2039, 1990.

[19] J. Little, "A proof for the queueing formula: $L = \lambda$," *Oper. Res.*, vol. 9, no. 4, pp. 204–209, July 1961.

[20] N. Maxemchuk, "Twelve random access strategies for fiber-optic networks," *IEEE Trans. Commun.*, vol. 36, pp. 942–950, Aug. 1988.

[21] R. Melhem, D. Chiarulli, and S. Levitan, "Space multiplexing of waveguides in optically interconnected multiprocessor systems," *Comput. J.*, vol. 32, no. 4, pp. 362–369, 1989.

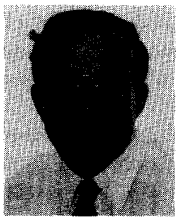
[22] R. Metcalfe and D. Boggs, "Ethernet: Distributed packet switching for local computer networks," *Commun. ACM*, vol. 19, no. 7, pp. 395–403, 1976.

[23] M. Nassehi, F. Tobagi, and M. Marhic, "Fiber optic configurations for local area networks," *IEEE J. Select. Areas Commun.*, vol. SAC-3, no. 6, pp. 941–949, Nov. 1985.
[24] Y. Ofek, "The topology, algorithms and analysis of a synchronous optical hypergraph architecture," Ph.D. dissertation, Univ. of Illinois at Urbana-Champaign, 1987.
[25] K. Padmanabhan and A. Netravali, "Dilated network for photonic switching," *Photon. Switching 1987,* also in *IEEE Trans. Commun.*, Dec, 1987.
[26] P. Perrier, P. Prucnal, and M. Chbat, "Demonstration of a self-clocked optical time-slot interchanger," *OSA Proc. Photon. Switching,* vol. 3, pp. 219–225, Mar. 1989.
[27] C. Qiao and R. Melhem, "Reconfiguration with time-division multiplexed MINs for multiprocessor communications," Tech. Rep. 91-20, Comput. Sci. Dep. Univ. Pittsburgh, Sept. 1991.
[28] C. Qiao, R. Melhem, D. Chiarulli, and S. Levitan, "Optical multicasting in linear arrays," *Int. J. Opt. Comput.*, vol. 2, pp. 31–48, 1991.
[29] S. Ramanan and H. Jordan, "Serial array shuffle-exchange architecture for universal permutation of time slots," *SPIE Proc., Digital Opt. Comput. II,* vol. 1215, pp. 330–342, Jan. 1990.
[30] D. Sarrazin, H. Jordan, and V. Heuring, "Digital fiber optical delay line memory," *Digital Opt. Comput. II, SPIE,* vol. 1215, pp. 366–375, Jan. 1990.
[31] J. Sauer, "A multi-Gb/s optical interconnect," *SPIE Proc., Digital Opt. Comput. II,* vol. 1215, pp. 198–207, 1990.
[32] F. Stone, J. Watson, D. Moser, and W. Minford, "Performance and yield of pilot-line quantities of lithium niobate switches," in *SPIE Conf. Proc. OE/Fibers '89,* Boston, MA, Sept. 1989.
[33] H. Takagi, *Analysis of Polling Systems.* Cambridge, MA: M.I.T. Press, 1986.
[34] R. Thompson and P. Giordano, "An experimental photonic time-slot interchanger using optical fibers as reentrant delay-line memories," *J. Lightwave Technol.*, vol. 1, pp. 154–162, 1987.
[35] R. Thompson, R. Anderson, J. Camlet, and P. Giordano, "Experimental modular switching system with a time-multiplexed photonic center stage," *OSA Proc. Photon. Switching,* vol. 3, pp. 212–218, 1989.
[36] R. Thompson, "Photonic time-multiplexed permutation switching using the dilated slipped banyan network," *J. Lightwave Technol.*, vol. 9, no. 12, pp. 1780–1787, Dec. 1991.
[37] F. Tobagi, F. Borgonovo, and L. Fratta, "Expressnet: A high-performance integrated-services local area network," *IEEE J. Select. Areas Commun.*, vol. SAC-1, no. 5, pp. 898–913, Nov. 1983.
[38] D. Z. Tsang, "One-gigabit per second free-space optical interconnection," *Appl. Opt.,* vol. 29, pp. 2034–2037, 1990.
[39] N. Whitehead and N. Parsons, *OSA Proc. Photon. Switching,* vol. 3, pp. 257–259, Mar 1989.

Chunming Qiao (S'89) was born in Suzhou, P.R. China, in 1965. He received the B.S. degree in computer science and engineering from the University of Science and Technology of China in 1985 and the M.S. degree in computer science from the University of Pittsburgh in 1990. He received the Ph.D. degree in computer science from the University of Pittsburgh, Pittsburgh, PA, in April 1993.

He will be joining the State University of New York at Buffalo in September 1993. His current research interests include parallel architectures, parallel processing and optical interconnection.

Mr. Qiao is a student member of the IEEE Computer Society.



Rami G. Melhem (S'82–M'84) was born in Cairo, Egypt, in 1954. He received the B.E. degree in electrical engineering from Cairo University, Egypt, in 1976, the M.A. degree in mathematics and the M.S. degree in computer science from the University of Pittsburgh in 1981, and the 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. He also coedited a special issue of the *Journal of Parallel and Distributed Computing* on Optical Computing and Interconnection Systems. He is a member of the IEEE Computer Society, the Association for Computing Machinery and the International Society for Optical Engineering.