Design and Implementation of Abacus Switch: A Scalable Multicast ATM Switch

H. Jonathan Chao, *Senior Member, IEEE*, Byeong-Seog Choe, *Member, IEEE*, Jin-Soo Park, *Student Member, IEEE*, and Necdet Uzun, *Member, IEEE*

***Abstract*—This paper describes a new architecture for a multicast ATM switch scalable from a few tens to a few thousands of input ports. The switch, called the Abacus switch, has a nonblocking switch fabric followed by small switch modules at the output ports. It has buffers at input and output ports. Cell replication, cell routing, output contention resolution, and cell addressing are all performed in a distributed way so that it can be scaled up to thousands of input and output ports. A novel algorithm has been proposed to resolve output port contention while achieving input buffers sharing, fairness among the input ports, and call splitting for multicasting. The channel-grouping mechanism is also adopted in the switch to reduce the hardware complexity and improve the switch's throughput, while the cell sequence integrity is preserved. The switch can also handle multiple priority traffic by routing cells according to their priority levels. The performance study of the Abacus switch in throughput, average cell delay, and cell loss rate is presented. A key ASIC chip for building the Abacus switch, called the ARC (ATM routing and concentration) chip, contains a two-dimensional array (32 $\times$ 32) of switch elements that are arranged in a crossbar structure. It provides the flexibility of configuring the chip into different group sizes to accommodate different ATM switch sizes. The ARC chip has been designed and fabricated using 0.8 μm CMOS technology and tested to operate correctly at 240 MHz.**

***Index Terms*—Asynchronous transfer mode, contention resolution, large-scale switches, multicast switches, switching systems.**

I. Introduction

THERE are several approaches to building a large-scale ATM switch. The first is using small ATM switch modules (e.g., 32 $\times$ 32) as building blocks and connecting them in a multistage structure (e.g., Clos-type interconnection) [1]–[5]. The problem with this approach is the performance degradation due to the internal blocking between the switch modules. Although the performance can be improved by speeding up the internal links or providing more interconnection links between modules, this approach has not convinced us that it is capable of providing satisfactory performance for a large-scale ATM switch.

Manuscript received May 1, 1996; revised December 1, 1996. This work was supported by NSF Grant NCR-9216287, the Center of Advanced Technology for Telecommunications in New York State, the Electronics and Telecommunications Research Institute in Korea, and the Ministry of Information and Communication in Korea under Grant U96-186. This paper was presented in part at GLOBECOM'96, London, England, November 1996.

H. J. Chao, J.-S. Park, and N. Uzun are with the Department of Electrical Engineering, Polytechnic University, Brooklyn, NY 11201 USA.

B.-S. Choe is with the Department of Electronics Engineering, Dongkuk University, Seoul, Korea.

Publisher Item Identifier S 0733-8716(97)03373-8.

The second approach is using high-speed technology to switch cells at a multiple Gbit/s rate in a core switch. For instance, AT&T, Fujitsu, NTT, and BNR switches switch cells at 2.5 or 10 Gbit/s [6]–[9]. The advantage of this approach is that the buffer required in the core switch is minimized. There are two reasons. First, as users' traffic is multiplexed to a high-bandwidth link, each individual user's traffic looks more like random traffic (i.e., less bursty). Second, when cells are multiplexed and switched at high speed, the channel grouping technique is applied implicitly, and thus requires less memory for the same performance. However, demultiplexers at the output of the core switch require large buffers because high-speed cell streams are routed to lower speed output links. Since the speed required for the demultiplexer's memory is lower than that of the core switch's memory, the need for large buffers at the demultiplexer can be justified.

Output buffering (including shared-memory output buffering) has been proven to provide the best delay and throughput performance. As the switch grows up to a certain size (e.g., 256 input and output ports), memory speed may become a bottleneck or the technology used to implement such a kind of memory may become too costly. One way to eliminate the memory's speed constraint is to temporarily store some cells destined for the same output port at the input buffers. Input buffering's well-known head-of-line (HOL) blocking drawback can be improved by speeding up the internal links' bandwidth (e.g., two–four times the input lines') and buffering excessive cells at the output ports. The input-and-output buffering approach thus provides satisfactory performance and eliminates memory speed limitation. Examples of input-and-output buffered ATM switches are NTT's and BNR's 160 Gbit/s switch. The challenge for implementing input-and-output buffered switches is the output port contention resolution of the input cells destined for the same output port (or the same module). Such a kind of function is usually handled by an arbiter. The bottleneck caused by the memory speed is now shifted to the arbiter. If parallel processing and pipeline techniques can be intelligently applied to implement the arbiter, a large-scale switch will be feasible.

In [10], we proposed a recursive modular architecture to implement a large-scale ATM switch. It was then modified to cope with the multicast capability [11], in which we showed that a switch that is designed to meet the performance requirement for unicast calls will also satisfy multicast calls' performance. Both architectures employed the generalized knockout concept [12] with output buffers. However, both

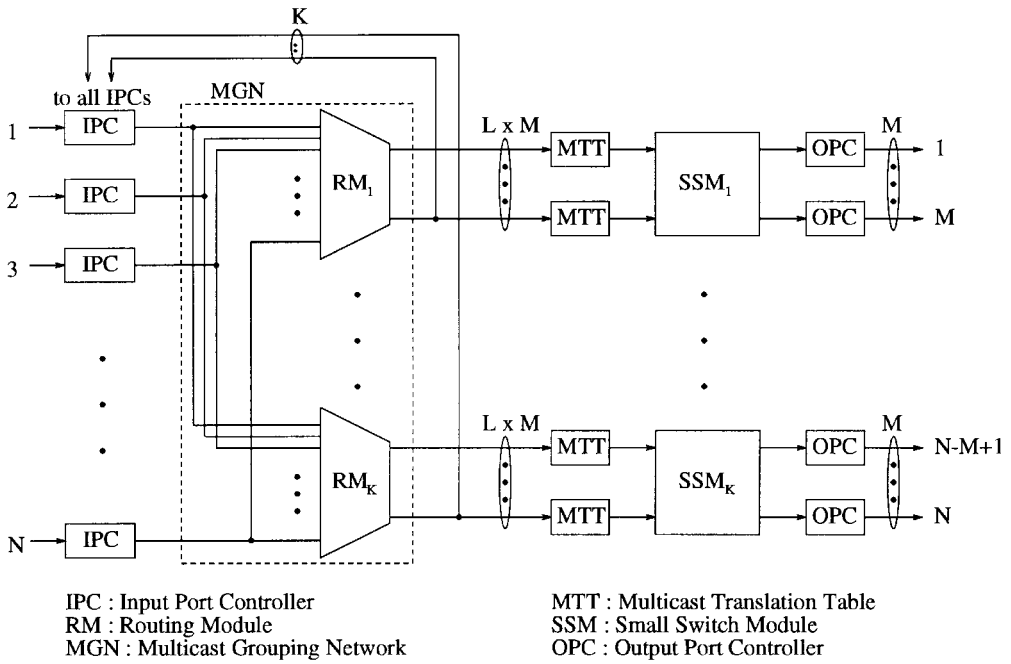

Fig. 1. Architecture of Abacus switch.

switch fabrics are a lossy system, where cells may be discarded when the number of routing links is less than the number of incoming cells destined for the same output port (or output group).

Here, we propose a new architecture eliminating the possibility of cells being discarded due to the loss of contention in the switch fabric. The new scalable multicast ATM switch has input and output buffers. It is called the *Abacus* switch because its switch fabric looks like an abacus. The Abacus switch consists of a nonblocking switch fabric followed by small switch modules at the output ports. Cell replication, cell routing, output contention resolution, and cell addressing are all performed distributedly in the Abacus switch so that it can be scaled up to thousands of input and output ports. The switch can be implemented with traditional economic CMOS technology while achieving compatible performance to those switches in [6]–[9]. Furthermore, the Abacus switch can be engineered in such a way that it requires small input buffers (e.g., 100 cells per input port) compared with large output buffers (e.g., a few tens of thousands of cells per output port). When implementing call admission control or buffer management, we will just need to focus on the output buffer rather than both input and output buffers, which reduces implementation complexity significantly.

We have proposed a novel algorithm to resolve the contention of multicast cells destined for the same output port (or output group). The new algorithm also has the following nice features: achieves input buffers sharing, provides fairness among the input ports, and supports call splitting for multicasting. The call-splitting function allows a multicast cell to be delivered to subsets of destined output ports in multiple cycles, thus increasing the system throughput [13]. We have also applied distributed and parallel processing techniques in the contention resolution to accommodate a large-scale switch. Several output contention resolution algorithms have been proposed, such as the recirculation algorithm [14], the three-phase algorithm [15], the ring reservation algorithm [16], and the centralized contention resolution device [17]. Most of them can only handle unicast calls (i.e., point-to-point communication) and N-to-1 selection (N is the switch size), while our algorithm can handle multicast calls, call splitting, and N-to-multiple selection when the channel grouping mechanism is applied. The channel grouping mechanism [18] is adopted in our switch to reduce the hardware complexity and improve the switch's throughput. It bundles multiple output ports, and permits them to share routing links among them.

This paper is organized as follows. Section II describes the architecture and operations of the Abacus switch. Section III presents the novel multicast contention resolution algorithm, which plays a major role in our switch architecture. Section IV describes the implementation of the input port controller. Section V shows an architecture to build a large-scale ATM switch with thousands of input and output ports. Section VI presents the performance study of the Abacus switch in throughput, average cell delay, and cell loss rate. Section VII briefly describes the design of the ARC (ATM routing and concentration) chip and its testing results [19]. Section VIII gives conclusions.

II. Architecture and Operations of Abacus Switch

As shown Fig. 1, the proposed Abacus switch consists of input port controllers (IPC's), a multicast grouping network (MGN), multicast translation tables (MTT's), small switch modules (SSM's), and output port controllers (OPC's). The switch performs cell replication and cell routing simultaneously. Cell replication is achieved by broadcasting incoming cells to all routing modules (RM's), which then selectively route cells to their output links. Cell routing is performed

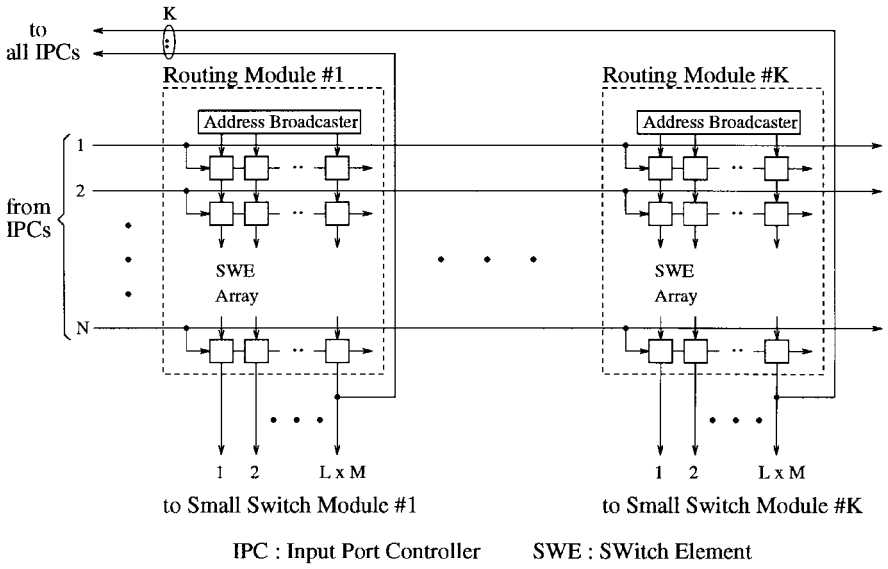

Fig. 2. Multicast grouping network (MGN).

distributedly by an array of switch elements (SWE's). The concept of sharing routing links (also called channel grouping [18]) is also applied to construct the MGN in order to reduce hardware complexity, where every M output ports are bundled in a group. For a switch size of N input ports and N output ports, there are K output groups $(K = N/M)$. The MGN consists of K routing modules; each of them provides $L \times M$ routing links to each output group. L is defined as group expansion ratio: the ratio of required routing links to the group size. Cells from the same virtual connection can be arbitrarily routed to any one of the $L \times M$ routing links, and their sequence integrity will be maintained. Based on a novel arbitration mechanism to be described in Section III, up to $L \times M$ cells from N IPC's can be chosen in each RM. Cells that lose contention are temporarily stored in an input buffer and will retry in the next time slot. On the other hand, cells that are successfully routed through RM's will further be routed to proper output port(s) through the SSM's.

We can engineer the group expansion ratio L in such a way that the required maximum throughput in a switch fabric can be achieved. Performance study (in Section VI) shows that the larger M is, the smaller L is required to achieve the same maximum throughput. For instance, for a group size M of 16 and input traffic with an average burst length of 15 cells, L has to be at least 1.25 to achieve a maximum throughput of 0.96. But, for a group size M of 32 and the same input traffic characteristic, L can be as low as 1.125 to achieve the same throughput.

The IPC's terminate input signals from the network, look up necessary information in a translation table, resolve contention among cells that are destined to the same output group, buffer those cells losing contention, and attach routing information in front of cells so that they can be routed properly in the MGN. Its implementation can be found in Section IV.

Each routing module (RM) in the MGN contains a two-dimensional array of switch elements and an address broadcaster (AB), as shown in Fig. 2. The SWE routes cells from the west and north to east and south, respectively, when it is at cross state, or to south and east, respectively, when it is at toggle state. The SWE's state is determined from the comparison of address bits and priority bits of cells from west and north. The AB generates dummy cells that carry proper output group addresses. This permits the SWE not to store the information of output group address, which simplifies the circuit complexity of the SWE significantly and results in higher VLSI integration density. The detailed operations of the SWE and the AB can be found in [10], [11], and [19]. In addition to routing cells, the RM's also sort cells' priority at the output links, which facilitates the new multicast contention resolution algorithm (see Section III for more details). Each routing module in the MGN has N horizontal input lines and $L \times M$ vertical routing links. These routing links are shared by the cells that are destined for the same output group (i.e., the same small switch module). Each input line is connected to all routing modules so that cells from any input line can be broadcast to all K output groups.

Cells from multicast calls are first replicated and routed by the MGN to multiple SSM's. Before the copied cells are further replicated and routed by the SSM's, their routing field will be updated by the multicast translation tables (MTT's) with proper routing information that is to be used by the SSM's. Each SSM has $L \times M$ inputs and M outputs. The SSM's must have multicast capability and output buffering structure. The latter is required to maintain the cell sequence for cells distributed among the $L \times M$ links. One example for such SSM's is Hitachi's 32×32 shared-buffered ATM switch [1]. The output port controller (OPC) updates each multicast cell with a new VCI/VPI and sends the cell to the network.

Fig. 3 shows an example illustrating how a cell is replicated in the MGN and the SSM's. Suppose a cell arrives at input port 3 and is to be multicast to three output ports: 1, M, and $(N-M+1)$. The cell is first broadcast to all K RM's in the MGN, but only the RM_1 and RM_K will accept the cell. Note that only one copy of the multicast cell will appear at one of

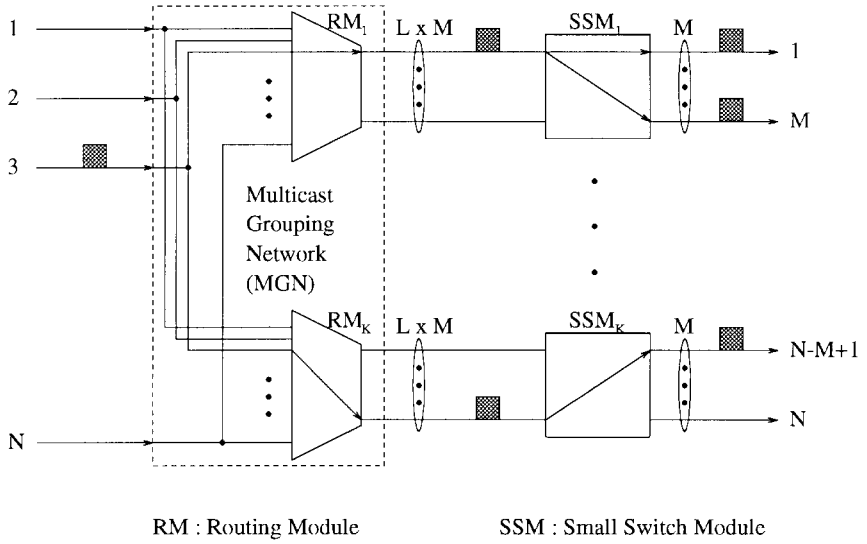

Fig. 3. Example of routing a multicast cell.

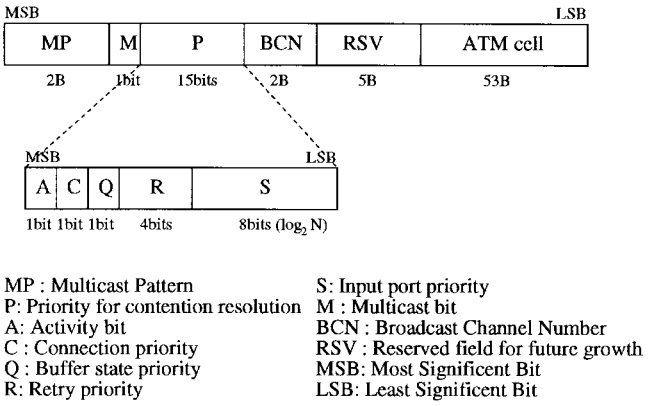

Fig. 4. Routing information used by Abacus switch with $N = 256$, $M = 16$.

the $L \times M$ links of the RM. The copied cell at the output of RM_1 is further replicated into two copies by SSM_1. There are a total of three replicated cells at the output ports.

Fig. 4 shows routing information for a multicast ATM switch with $N = 256$ and $M = 16$, which consists of several fields, multicast pattern (MP), priority field (P), and a broadcast channel number (BCN). A multicast pattern is a bit map of all of the output groups and is used in the MGN for routing cells to multiple output groups. Each bit indicates if the cell is to be sent to the associated output group. For instance, if the ith bit in the MP is set to "1," the cell is to be sent to the ith output group. The multicast pattern MP has K bits for an MGN that has K output groups (16 in this example). For a unicast call, its multicast pattern is basically a flattened output address (i.e., a decoded output address) in which only one bit is set to "1" and all other $(K - 1)$ bits are set to "0." For a multicast call, there is more than one bit set to "1" in the MP, corresponding to the output groups for which the cell is destined.

A priority field (P), used to assist contention resolution, can be flexibly set to any value to achieve desired service preference. For instance, the priority field may consist of an activity bit (A), a connection priority (C), a buffer state priority (Q), a retry priority (R), and an input port priority (S). Let us assume the smaller the priority value, the higher the priority level. The activity bit (A) indicates the validity of the cell. The activity bit (A) is set to "0" if the cell is valid and set to "1" otherwise. The connection priority (C) indicates the priority of the virtual connection, which can be determined during the call setup or service provisioning. The buffer state priority (Q) provides a sharing effect among N input buffers by allowing the HOL cell in an almost-overflowed buffer (e.g., exceeding a predetermined threshold) to be transmitted sooner so that the overall cell loss probability is reduced. The retry priority (R) provides global first-come, first-served (FCFS) discipline, allowing a cell's priority level to move up by one whenever it loses contention once. The retry priority (R) can initially be set to "1111" and decreased by one whenever losing contention once. In order to achieve fairness among input ports, the priority levels of the head-of-line cells at the input ports dynamically change at each time slot. The input port priority (S) can initially be set to its input port address with $\log_2 N$ bits and decreased by one at every time slot, thus achieving round-robin fairness.

The broadcast channel number (BCN) in Fig. 4 will be used to find a new multicast pattern in the MTT, allowing the copied cell to be further duplicated in the SSM. The BCN will also be used by the OPC to find a new VCI/VPI for each copy of the replicated cell. The BCN can be either assigned during call setup or a combination of input port number and the VPI/VCI value.

III. Multicast Contention Resolution Algorithm

Here, we describe a novel algorithm that resolves output port contention among the input ports in a fair manner. It can also do call splitting for multicasting, and thus improves the system throughput. By applying distributed and parallel processing techniques, our contention resolution algorithm is able to accommodate a large-scale switch. The output port contention resolution is often implemented by a device

called an arbiter. Most proposed arbiters can only handle unicast calls (i.e., point-to-point communication) and N-to-1 selection, for example: three-phase [15], ring reservation [16], and centralized contention resolution device [17].

Implementing an arbiter capable of handling call splitting and N-to-multiple selection is much more challenging in terms of timing constraint. At the beginning of the cell time slot, the arbiter receives N multicast patterns, one from each input port, and returns acknowledgment to those input ports whose HOL cells have won contention. These cells are then allowed to transmit to the switch fabric. Let us consider these N multicast patterns (each with K bits in our architecture example) being stacked up, and there are K columns with N bits in each column. Each column associates with each output group. The arbiter's job is to select up to, for example, $L \times M$ bits that are set to "1" from each column and repeat the operation for K times, which must be finished in one cell time slot. In other words, the arbitration's timing complexity is on the order of $O(N \times K)$. The arbiter may become the system's bottleneck when N or K is large.

The arbitration scheme we propose here performs N-to-$L \times M$ selection in a distributed manner using the switch fabric and all IPC's, thus eliminating the speed constraint. Another difference between our arbitration scheme and others is that in our scheme, the HOL cell is repeatedly sent to the switch fabric to compete with others until it has successfully transmitted to all necessary output groups for which the cell is destined. Unlike other arbitration schemes, our scheme does not wait for an acknowledgment before transmitting the cell. When a cell is routed in a switch fabric without waiting for an acknowledgment, two situations are possible. It could be successfully routed to all necessary output groups, or only routed to a subset of the output groups (including an empty set). The latter case is considered a failure, and the HOL cell will retry in the next time slot. When a cell is transmitted to the switch fabric, since it does not know if it will succeed, it must be stored in a one-cell buffer for possible retransmission.

Now, the question is: How does the IPC know whether or not its HOL cell has been successfully transmitted to all necessary output groups? In our implementation, the RM's are responsible for returning the routing results to the IPC. One possible way is to let each RM inform IPC's about the identification (e.g., the broadcast channel number) of the cells that have been successfully routed. However, since a cell could be routed to multiple output groups (for instance, up to K output groups for a broadcast situation), one IPC may receive up to K acknowledgment from K RM's. The complexity of returning the identification of every successfully routed copy to all IPC's is too high to be practical for a large-scale switch. In the following, we introduce a scheme that significantly simplifies the complexity of the acknowledgment operation.

The RM not only can route cells to proper output groups, but also, based on cells' priority levels, can choose up to $L \times M$ cells that are destined for the same output group. The HOL cell of each input port is assigned a *unique* priority level that is different from the others. After cells are routed through an RM, they are sorted at the output links of the RM according to their priority levels from left to right in a descending order (see Fig. 2). The cell that appears at the rightmost output link has the lowest priority level among the cells that have been routed through this RM. This lowest priority information is broadcast to all IPC's. Each IPC will then compare the local priority level (LP) of the HOL cell with a feedback priority, say FP_j, to determine if the HOL cell has been routed through the RM_j. Note that there are K feedback priorities, $\mathrm{FP}_1, \cdots, \mathrm{FP}_K$. If the feedback priority level (FP_j) is lower than or equal to the local priority level (LP), the IPC determines that its HOL cell has reached one of the output links of the RM_j. Otherwise, the HOL cell must have been discarded in the RM_j due to loss of contention and will be retransmitted in the next time slot. Since there are K RM's in total, there will be K lines broadcast from K RM's to all IPC's, each carrying the lowest priority information in its output group.

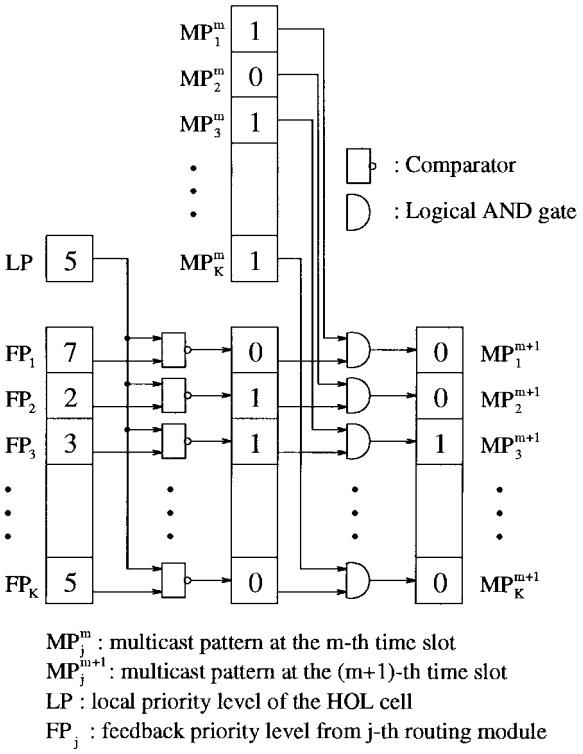

Fig. 5. Example of modifying a multicast pattern (MP).

The priority assigned to the HOL cells will be dynamically changed according to some arbitration policies, such as random, round-robin, state-dependent, and delay-dependent [20]. The random scheme randomly chooses the HOL cells of input ports for transmission; the drawback is it has a large delay variation. The round-robin scheme chooses HOL cells from input ports in a round-robin fashion by dynamically changing the scanning point from the top to the bottom input port (e.g., S field in Fig. 4). The state-dependent scheme chooses the HOL cell in the longest input queue such that input queue lengths are maintained nearly equal, achieving the input buffers sharing effect (e.g., Q field in Fig. 4). The delay-dependent scheme performs like a global FIFO, where the oldest HOL cell has the highest priority to be transmitted to the output (e.g., R field in Fig. 4). Since our arbitration is performed in a distributed manner by K RM's and in parallel by IPC's, we

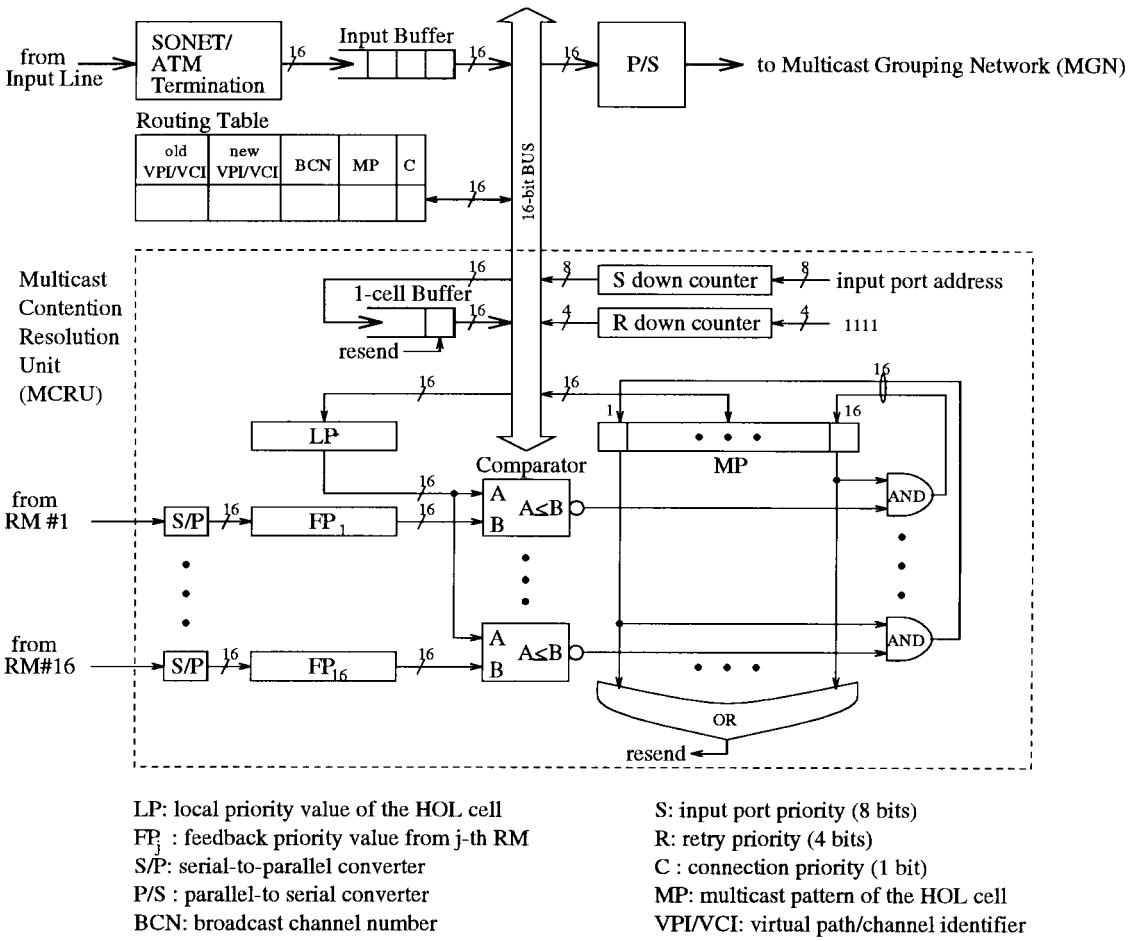

Fig. 6. Implementation of the input port controller (IPC) with $N = 256, M = 16$.

can implement any of the above policies, or a combination of them, by arbitrarily assigning a proper priority level to the HOL cell.

At the beginning of the time slot, each IPC sends its HOL cell to the MGN. Meanwhile, the HOL cell is temporarily stored in a one-cell size buffer during its transmission. After cells have traversed through the RM's, priority information FP_1 to FP_K (the priority of the rightmost link of each RM) is fed back to every IPC. Each IPC will then compare the feedback priority level $\mathrm{FP}_j, j = 1,2,\cdots,K$, with its local priority level LP. Three situations can happen. First, $\mathrm{MP}_j = 1$ and $\mathrm{LP} \leq \mathrm{FP}_j$ (recall that the smaller the priority value, the higher the priority level), which means the HOL cell is destined for the jth output group and has been successfully routed through the jth RM. The MP_j bit is then set to "0." Second, $\mathrm{MP}_j = 1$ and $\mathrm{LP} > \mathrm{FP}_j$, which means that the HOL cell is destined for the jth output group, but discarded in the jth RM. The MP_j bit remains "1." Third, $\mathrm{MP}_j = 0$, the jth bit of the HOL cell's multicast pattern can be equal to "0," which means that the HOL cell is not destined for the jth output group. Then, the MP_j bit remains "0."

After all MP_j bits $(j = 1,2,\cdots,K)$ have been updated according to one of the above three scenarios, a signal indicating whether the HOL cell should be retransmitted, *resend*, will be asserted to "1" if one or more than one bits in the multicast pattern remains "1." The *resend* signal is initially set to "0." If multicast pattern bits are all "0," meaning that the HOL cell has been successfully transmitted to all necessary output groups, the *resend* signal will be disasserted. The IPC will then clear the HOL cell in the one-cell buffer and transmit the next cell in the input buffer in the next time slot (if any).

Fig. 5 gives an example of how a multicast pattern is modified. Let us assume at the beginning of the mth time slot that the HOL cell is destined for three output groups: 1, 3, K. Therefore, the multicast pattern at the mth time slot MP^m has three bits set to "1." Let us also assume that the local priority value (LP) of the HOL cell is 5, and that the feedback priority values from 1, 2, 3, and K are 7, 2, 3, and 5, respectively, as shown in Fig. 5. The result of comparing LP with FP's is "$0110\cdots00$," which is then logically ANDed with the MP^m and produces a new multicast pattern "$0010\cdots00$" for the next time slot (MP^{m+1}). Since only the MP_3^{m+1} is set to "1," the IPC determines that the HOL cell has been successfully routed to RM's 1 and K, but discarded in RM 3 and will retransmit in the next time slot.

IV. Implementation of Input Port Controller (IPC)

Fig. 6 shows a block diagram of the IPC. For easy explanation, let us assume that the switch has 256 input ports and 256 output ports, and every 16 output ports are in one

group. A major difference between this IPC and traditional ones is the addition of the multicast contention resolution unit (MCRU), shown in a dashed box. It determines, by comparing K feedback priorities with the local priority of the HOL cell, whether or not the HOL cell has been successfully routed to all necessary output groups.

Let us start from the left where the input line from the SONET/ATM network is terminated. Cells 16 bits wide are written into an input buffer. The HOL cell's VCI/VPI is used to extract necessary information from a routing table. This information includes a new VPI/VCI for unicast connections, a broadcast channel number (BCN) for multicast connections, which uniquely identifies each multicast call in the entire switch, multicast pattern (MP) for routing cells in the MGN, and the connection priority (C). This information is then combined with a priority field to form the routing information, as shown in Fig. 4.

As the cell is transmitted to the MGN through a parallel-to-serial converter (P/S), the cell is also temporarily stored in a one-cell buffer. If the cell fails to successfully route through RM's, it will be retransmitted in the next cell cycle. During retransmission, it is written back to the one-cell buffer in case it fails to route through again. The S down-counter is initially loaded with the input address and decremented by one at each cell clock. The R down-counter is initially set to all "1's" and decreased by one every time the HOL cell fails to transmit successfully. When the R counter reaches zero, it will remain at zero until the HOL cell has been cleared and a new cell becomes the HOL cell.

K feedback priority signals FP_1 to FP_K are converted to 16 bit wide signals by the serial-to-parallel converters (S/P) and latched at the 16 bit registers. They are simultaneously compared with the HOL cell's local priority (LP) by K comparators. Recall that the larger the priority value is, the lower the priority level is. If the value of the FP_j is larger than or equal to the local priority value (LP), the jth comparator's output is asserted low, which will then reset the MP_j bit to zero regardless of what its value was ("0" or "1"). After the resetting operation, if any one of the MP_j bits is still "1," indicating that at least one HOL cell did not get through the RM in the current cycle, the "*resend*" signal will be asserted high, and the HOL cell will be retransmitted in the next cell cycle with the modified multicast pattern.

As shown in Fig. 5, there are K sets of S/P, FP register, and comparator. As a switch size increases, the number of output groups K also increases. In order to reduce hardware complexity, if we time-division multiplex the operation of comparing the local priority value LP with K feedback priority values, only one set of this hardware is required.

V. An Architecture for a Large-Scale Abacus Switch

The Abacus switch has employed several techniques to accommodate a large-scale size (e.g., 1024 $\times$1024). For instance, a crossbar structure of the SWE array permits short interconnection between SWE's. Input–output buffering allows lower speed memory chips at the input and output ports. Distributed and parallel processing techniques have been used to implement the multicast contention resolution. However, the timing requirement of routing cells and resolving contention also needs to be taken into consideration when building a large-scale switch.

Due to the timing alignment requirement for the signals of the vertical routing links and the horizontal lines in the RM, incoming cells and dummy cells from the address broadcasters (AB's) are skewed properly before they are sent to the SWE array. To implement the proposed multicast contention resolution algorithm, the time it takes to route cells through an RM and to feed back the lowest priority information from the RM to all IPC's must be less than one cell slot time. If the time is greater than one cell slot time, two situations can happen. First, if the HOL cell is held up in the one-cell buffer longer than a cell slot time, the system throughput will be degraded. On other hand, if a cell next to the HOL cell is allowed to transmit before the HOL cell has been successfully transmitted to output(s), it may cause a cell out-of-sequence problem. Although it can be resolved with a resequencing circuit at the output port, the complexity may be too high to be practical.

Here, we will ensure that the feedback priorities are returned to all IPC's within one cell time slot. Since each SWE in an RM introduces a 1-bit delay as the signal passes it in either direction, the sum of the maximum number of SWE's between the IPC and the rightmost link of the RM, and the number of bits in MP field and P field, should be less than the number of bits in a cell. In other words, $(N+L\times M-1)+(N/M+\log_2 N+8)$ should be less than the cell length in bits, where eight is chosen based on the partial routing information attached to the 53-byte cell in Fig. 4 (i.e., $M, A, C, Q,$ and R bits). For example, if we choose $M=16, L=1.25$, and a cell size of 64 bytes within the switch fabric, the equation becomes $N+20-1+N/16+\log_2 N+8<512$. Or, the maximum value of N is 448, which is not large enough for a large-scale switch.

Fig. 7 shows a proposed architecture to implement a large-scale ATM switch. In order to reduce the time spent on traversing cells through the RM and returning the lowest priority information to IPC's, the number of SWE's in the RM cannot be too large. For instance, if we partition N inputs into K_1 groups, each group with n inputs (i.e., $N=n\times K_1$), the MGN's size is reduced from N to n. In other words, one big MGN is divided into K_1 smaller MGN, and each MGN has n input lines, as shown in Fig. 7. Recall that each output group of M output ports requires $L\times M$ routing links to achieve an acceptable throughput. Now, there are K_1 MGN's; each has $L\times M$ routing links for each output group. Therefore, we need to further concentrate $K_1\times(L\times M)$ lines to $L\times M$ outputs by using concentration modules (CM's) at the second stage, where N/M CM's are required. The structure and implementation of the CM and the RM are identical, except that the function performed is a little different. Since cells that pass through the RM's to the CM always have correct output group addresses, we just need to perform concentration by using the priority field in the routing information.

Note that the feedback lines carrying the lowest priority are returned from the second-stage CM output links, instead of

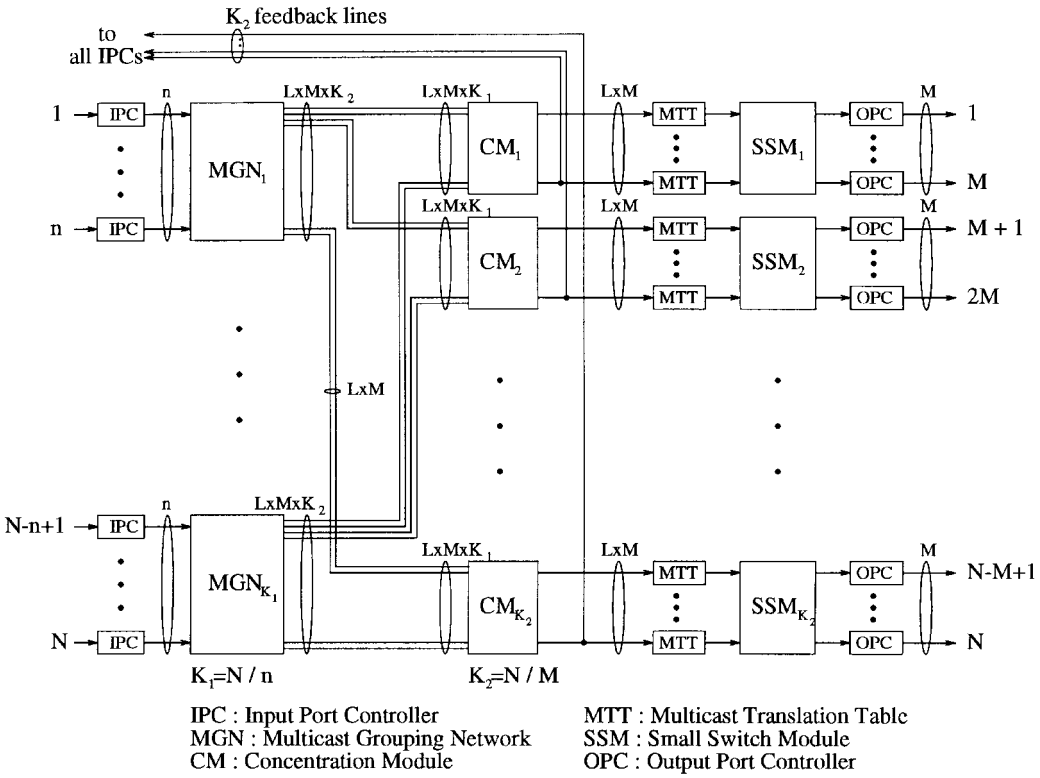

Fig. 7. Architecture for a large-scale Abacus switch.

the first-stage RM outputs. Since there are K_2 (N/M) output ports, K_2 feedback lines are needed. Now, the maximum delay between the cell entering the switch fabric and the availability of the feedback priority is $n + K_1(L \times M) + L \times M - 1 + [N/M + \log_2 N + 8]$, which should be less than the cell length in the switch. The cell length in the switch is equal to $[N/M + \log_2 N + 8]$ bits plus 53 bytes, or $n + K_1(L \times M) + L \times M - 1 < 424$. For example, if we choose $M = 16, L = 1.25$, and $n = 256$, the maximum number of inputs that can be supported is 1907.

VI. Performance Analysis of Abacus Switch

In this section, the performance analysis of the Abacus switch is presented. Both simulation and analytical results are shown to compare with each other. Simulation results are obtained with a 95% confidence interval, not greater than 10% for the cell loss probability or 5% for the maximum throughput and average cell delay. In our analysis, we consider an ON–OFF source model in which an arrival process to an input port alternates between ON (active) and OFF (idle) periods. A traffic source, during the ON period, continues sending cells in every time slot, but stops sending cells in the OFF period. Both the durations of the ON and OFF periods are assumed to be geometrically distributed.

A. Maximum Throughput

In this section, we analyze the maximum throughput of the Abacus switch. The maximum throughput of an ATM switch employing input queueing is defined by the maximum utilization at the output port. An input-buffered ATM switch has the so-called HOL blocking problem which degrades the switch's throughput. However, the throughput can be improved by speeding up the switch fabric's operation rate or increasing the number of routing links with an expansion ratio L. Several other factors also affect the maximum throughput. For instance, the larger the switch size (N), burstiness (β), or input buffer size (B_i), the smaller the maximum throughput $(\rho_{\max})$ will be. However, the larger the group expansion ratio (L) or group size (M), the larger the maximum throughput will be.

Fig. 8 shows that the maximum throughput is monotonically increasing with the group size. For $M = 1$, the switch becomes an input-buffered switch, and its maximum throughput $\rho_{\max}$ is 0.586 for uniform random traffic $(\beta = 1)$ and $\rho_{\max} = 0.5$ for completely bursty traffic $(\beta \to \infty)$. For $M = N$, the switch becomes a completely shared memory switch such as Hitachi's switch [1]. Although it can achieve 100% throughput, it is impractical to implement a large-scale switch using such an architecture. Therefore, choosing M between 1 and N is a compromise between the throughput and the implementation complexity. Figs. 9 and 10 compare theoretical values and simulation values of the maximum throughput with different group expansion ratios (L) for $M = 1$ and $M = 16$, respectively. The theoretical values can be obtained from Liew's analysis [21].

B. Average Delay

A cell may experience two kinds of delay while traversing through the Abacus switch: input buffer delay and output buffer delay. To evaluate the delay at the output port, we

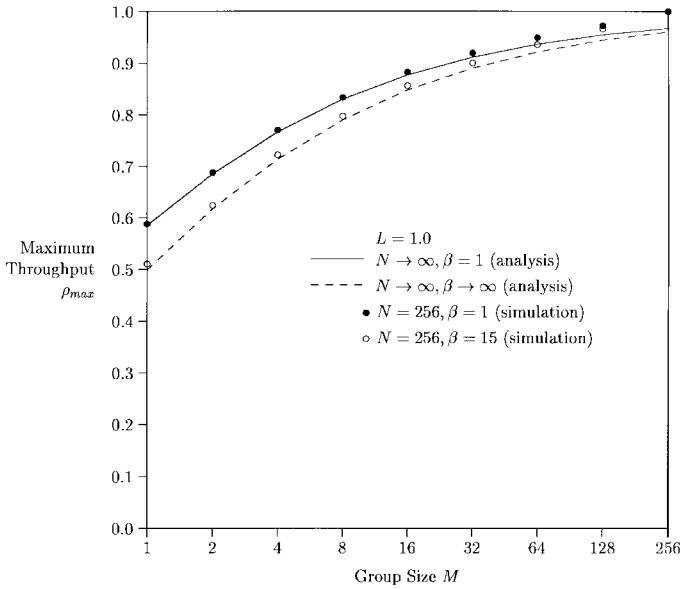

Fig. 8. Maximum throughput versus group size with $L = 1.0$.

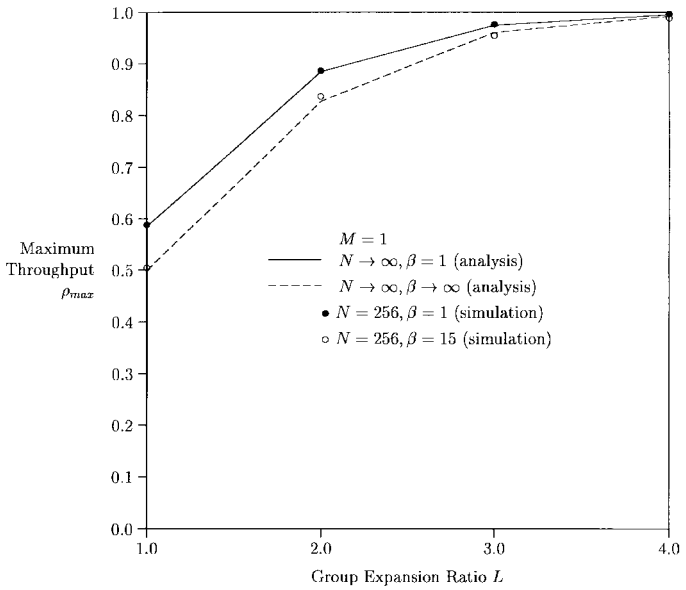

Fig. 9. Maximum throughput versus group expansion ratio with $M = 1$.

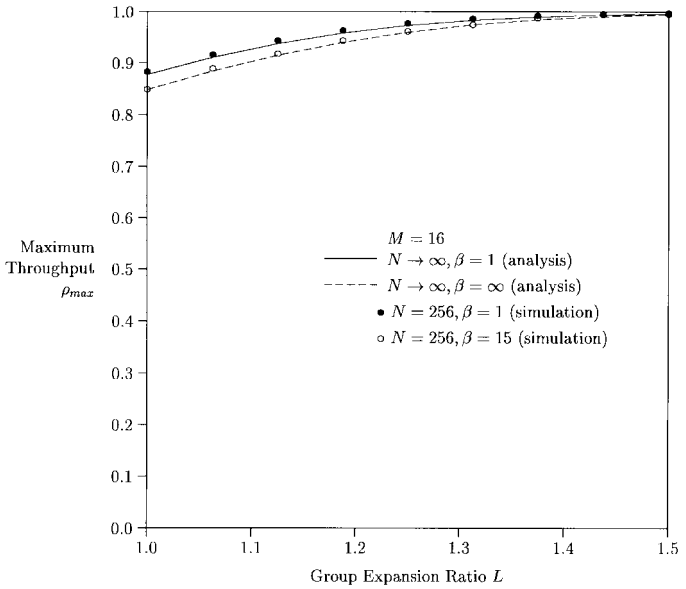

Fig. 10. Maximum throughput versus group expansion ratio with $M = 16$.

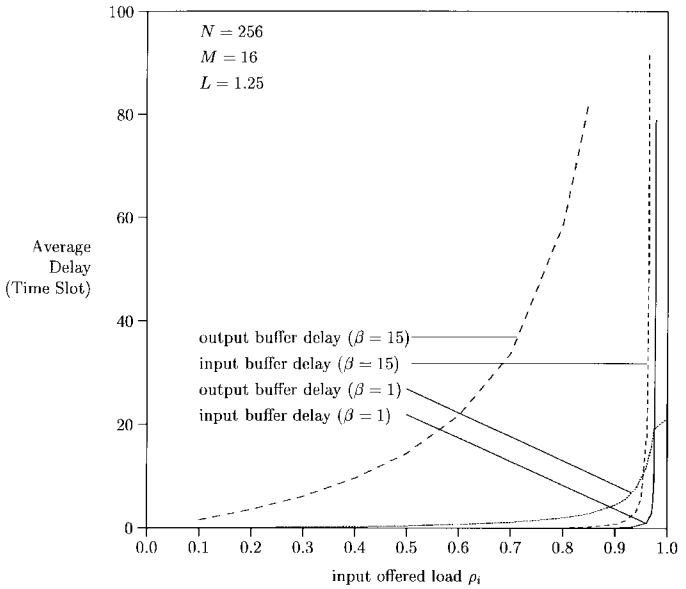

Fig. 11. Comparison of average input buffer delay and average output buffer delay versus input offered load (simulation).

assume that a small switch module in the Abacus switch is a shared-buffered switch, where all M output ports share a physical memory. Fig. 11 shows simulation results of the input and output buffer's average delay versus input offered load ρ_i. Note that the input buffer's average delay is much smaller than the output buffer's average delay at traffic load less than the saturated throughput. For example, for an input offered load ρ_i of 0.8 and an average burst length β of 15, the output buffer's average delay T_o is 58.8 cell times, but the input buffer's average delay T_i is only 0.1 cell time. It also shows that the impact of the burstiness of input traffic to the input buffer's average delay is very small when the traffic load is below the maximum throughput.

Fig. 12 shows simulation results of the input buffer's average delay versus expanded throughput ρ_j for both unicast and multicast traffic. Here, we assumed that the number of replicated cell is distributed geometrically with an average of c. The expanded throughput ρ_j is measured at the inputs of the SSM and normalized at each output port. Note that multicast traffic has a lower delay than unicast traffic because a multicast cell can be sent to multiple destinations in a time slot, while a unicast cell can be sent to only one destination in a time slot. For example, assume that an input port i has ten unicast cells, and the other input port j has a multicast cell with a fan-out of ten. Input port i will take at least ten time slots to transmit the ten unicast cells, while input port j can possibly transmit the multicast cell in one time slot.

C. Cell Loss Probability

As suggested in [23], there can be two buffer control schemes for an input–output buffered switch: the queue loss

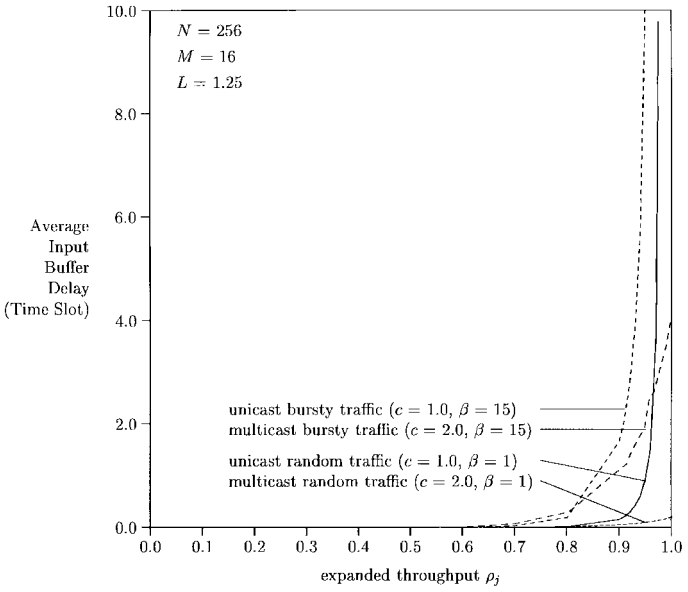

Fig. 12. Average input buffer delay versus expanded throughput for unicast and multicast traffic (simulation).

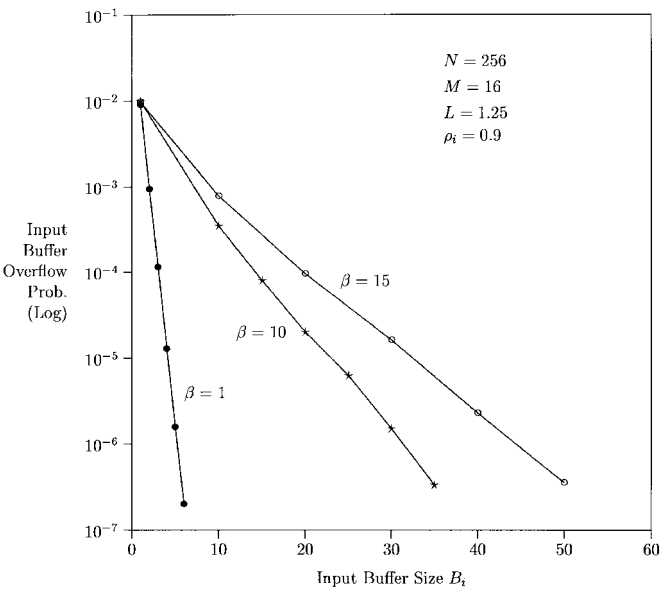

Fig. 13. Input buffer overflow probability versus input buffer size (simulation).

(QL) scheme and the backpressure (BP) scheme. In the QL scheme, cell loss can occur at both input and output buffers. All of the simulation results shown in previous sections are based on the QL scheme.

In the BP scheme, by means of backward throttling, the number of cells actually switched to each output group is limited not only to the group expansion ratio $(L \times M)$, but also to the current storage capability in the corresponding output buffer. For example, if the free buffer space in the corresponding output buffer is less than $L \times M$, only the number of cells corresponding to the free space are transmitted, and all other HOL cells destined for that output group remain at their respective input buffer. The Abacus switch can easily implement the backpressure scheme by forcing the address broadcaster (AB) in Fig. 2 to send the dummy cells with the highest priority level, which will automatically block the input cells from using those routing links. Furthermore, the number of blocked links can be dynamically assigned based on the output buffer's congestion situation.

Here, we only consider the QL scheme (cell loss at both input and output buffers). In the Abacus switch, cell loss can occur at input and output buffers, but not in the MGN. Fig. 13 shows input buffer overflow probabilities with different average burst lengths β. For uniform random traffic, an input buffer with a capacity of a few cells is sufficient to maintain the buffer overflow probability to be less than 10^{-6}. As the average burst length increases, so does the cell loss probability. For an average burst length β of 15, the required input buffer size can be a few tens of cells for the buffer overflow probability of 10^{-6}. By extrapolating the simulation result, the input buffer size is about 100 cells for 10^{-10} cell loss rate.

Fig. 14 shows output buffer overflow probabilities with different average burst lengths. Here, B_o is the normalized buffer size for each output. We notice that the required output buffer size is much larger than the input buffer size for the same cell loss probability.

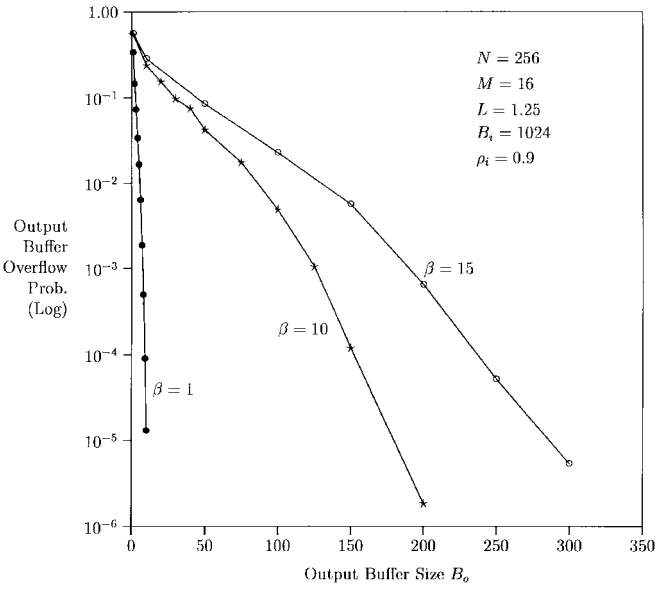

Fig. 14. Output buffer overflow probability versus output buffer size (simulation).

VII. The ATM Routing and Concentration (ARC) Chip

A. Chip Block Diagram

Fig. 15 shows the ARC chip's block diagram. Each block's function and design are explained briefly in the following sections. Details can be found in [24]. The ARC chip contains 32 × 32 switch elements (SWE's), which are partioned into eight SWE arrays, each with 32 × 4 SWE's. A set of input data signals $w[0:31]$ comes from IPC's. Another set of input data signals $n[0:31]$ either comes from the output $s[0:31]$ of the chips on the above row, or is tied to high for the chips on the first row (in the multicast case). A set of the output signals $s[0:31]$ either goes to the north input of the chips one row below or goes to the output buffer.

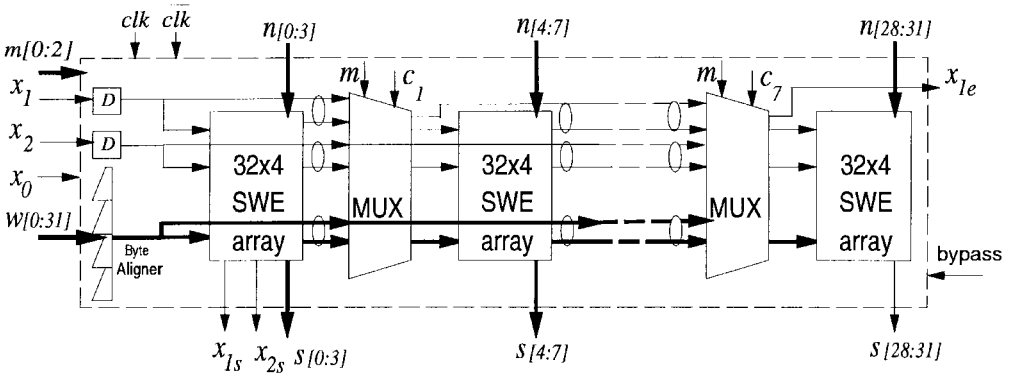

Fig. 15. Block diagram of the ARC chip.

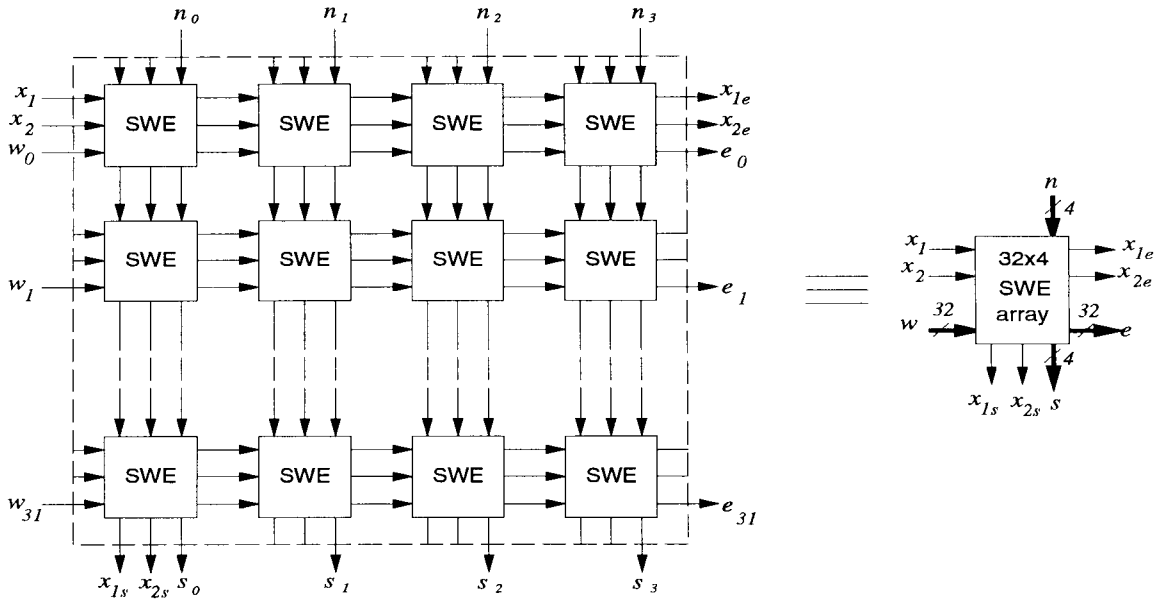

Fig. 16. 32 × 4 SWE array.

The x_0 signal is broadcast to all SWE's to initialize each SWE to a cross state, where the west input passes to the east and the north input passes to the south. The x_1 signal specifies the address bit(s) used for routing cells, while the x_2 signal specifies the priority field. Other x output signals propagate along with cells to the adjacent chips on the east or south side.

The $m[0:1]$ signals are used to configure the chip into four different group sizes as shown in Table I: 1) eight groups, each with four output links, 2) four groups, each with eight output links, 3) two groups, each with 16 output links, and 4) one group with 32 output links. The $m[2]$ signal is used to configure the chip to either unicast or multicast application. For the unicast case, $m[2]$ is set to zero, while for the multicast case, $m[2]$ is set to one.

TABLE I
TRUTH TABLE FOR DIFFERENT OPERATION MODES

m1	m0	Operation
0	0	8 groups with 4 links per group
0	1	4 groups with 8 links per group
1	0	2 groups with 16 links per group
1	1	1 group with 32 links per group

m2=1 multicast, m2=0 unicast

B. 32 × 4 *SWE Array*

As shown in Fig. 16, the SWE's are arranged in a crossbar structure, where signals only communicate between adjacent SWE's, easing the synchronization problem. ATM cells are propagated in the SWE array similar to a wave propagating diagonally toward the bottom right corner. The x_1 and x_2 signals are applied from the top left of the SWE array, and each SWE distributes the x_1 and x_2 signals to its east and south neighbors. This requires the same phase to the signal arriving at each SWE. The x_1 and x_2 signals are passed to the neighbor SWE's (east and south) after one clock cycle delay, as are data signals (w and n). The x_0 signal is broadcast to all SWE's (not shown in Fig. 16) to precharge an internal node in the SWE in every cell cycle. The x_{1e} output signal is used to identify the address bit position of the cells in the first SWE array of the next adjacent chip.

The timing diagram of the SWE input signals and its two possible states are shown in Fig. 17. Two bit-aligned cells, one from the west and one from the north, are applied to the

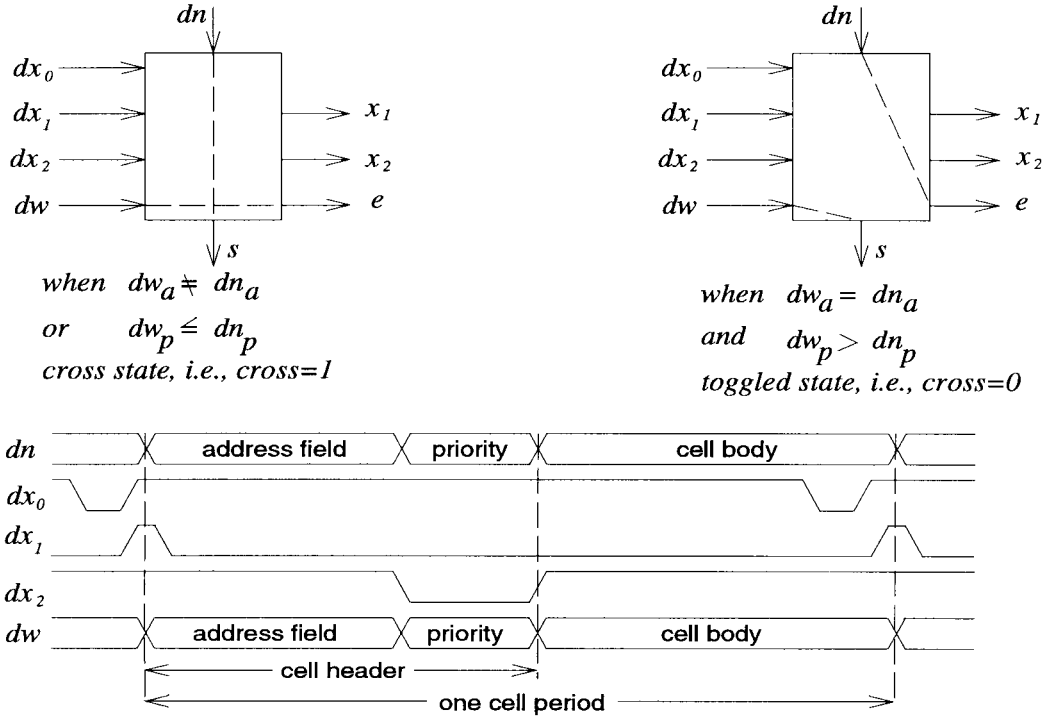

Fig. 17. Two states of the switch element.

SWE along with the dx_1 and dx_2 signals, which determine the address and priority fields of the input cells. The SWE has two states: cross and toggle. Initially, the SWE is initialized to a cross state by the dx_0 signal, i.e., cells from the north side are routed to the south side, and cells from the west side are routed to the east side. When the address of the cell from the west (dw_a) is matched with the address of the cell from the north (dn_a), and when the west's priority level (dw_p) is higher than the north's (dn_p), the SWE is toggled. The cell from the west side is then routed to the south side, and the cell from the north is routed to the east. Otherwise, the SWE remains at the cross state.

C. Testing Results

The 32 × 32 ARC chip has been designed and fabricated using 0.8 μm CMOS technology with a die size of 6.6 × 6.6 mm. Note that this chip is pad limited. The chip has been tested successfully up to 240 MHz by using a high-speed oscilloscope, timing analyzer, and a pattern generator capable of generating signals up to 1 GHz. The chip's characteristics are summarized in Table II. Its photograph is shown in Fig. 18.

Fig. 19 shows a testing result, where x_{2s}, s_0, s_1, and s_2 are shown from top to bottom, respectively. x_{2s} specifies the range of the priority field of the cells at the output, which is chosen to be seven bits in this test. Since x_{2s} is taken from the bottom left of the SWE from which s_0 comes out, it is aligned with s_0. s_1 is delayed by one clock cycle with respect to x_{2s}. Similarly, s_2 is delayed by two clock cycles with respect to x_{2s}. In this test, cells are applied to the west inputs, while north inputs are tied to VDD. It is observed that south outputs come out in a sorted priority order. The priority of s_0 is 1000100, which is the highest priority among all inputs. The priority of s_1 is 1000101, which is the second highest. The priority of s_2 is 1000110, which is the third highest. Note that the cell length used here is kept short in order to be able to see one cell cycle in the viewing window of the oscilloscope.

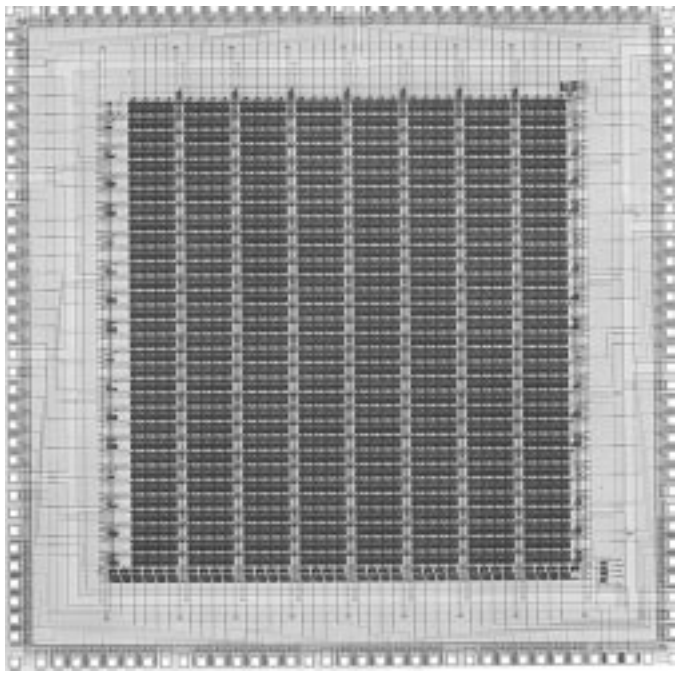

Fig. 18. Photograph of the ARC chip.

VIII. Conclusion

We have described a new architecture to implement a multicast ATM switch scalable from a few tens to a few

TABLE II
CHIP SUMMARY

Process Technology	0.8-μm CMOS, triple metal
Number of switching elements	32x32
Configurable group size	4, 8, 16, or 32 output links
Pin count	145
Package	Ceramic PGA
Number of transistors	81,000
Die size	6.6 x 6.6 mm^2
Clock signals	Pseudo ECL
Interface signals	TTL/CMOS inputs, CMOS outputs
Maximum clock speed	240MHz
Worst-case power dissipation	2.8W at 240MHz

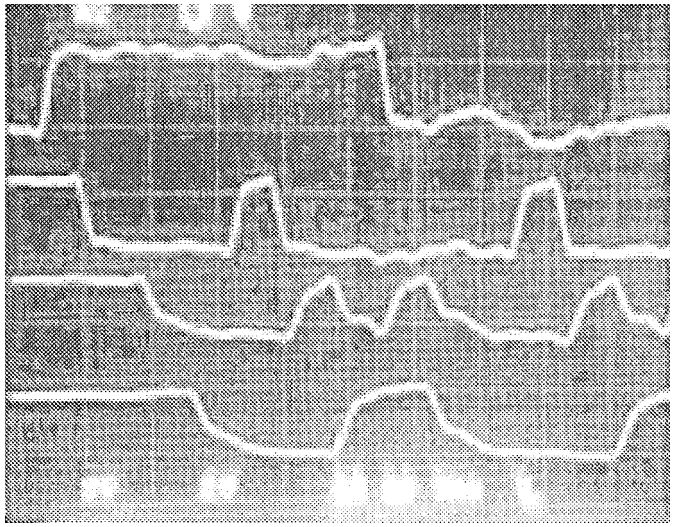

Fig. 19. Photograph of a testing result with unit grid of 5 ns.

thousands of input ports. The switch, called the Abacus switch, consists of a nonblocking switch fabric followed by small switch modules at the output ports, and has buffers at the input and output ports. The switch employs a novel algorithm to resolve the contention of multicast cells destined for the same output port (group). The algorithm also provides the capability of sharing input buffers, effectively achieving fairness among the input ports, and performing call splitting for multicasting. The channel grouping mechanism is adopted in our switch to reduce the hardware complexity and improve the switch's throughput, while the cell sequence integrity is preserved. The switch can also handle multiple priority traffic by routing cells according to their priority levels.

Cell replication, cell routing, output contention resolution, and cell addressing are all performed distributedly in the Abacus switch so that it can be scaled up to thousands of input and output ports. The cell replication is achieved by broadcasting incoming cells to multiple routing modules, which consists of a two-dimensional array of switch elements (SWE's). The regular structure permits us to implement a high-density VLSI chip and to have relaxed synchronization for data and clock signals. A key ASIC chip for building the Abacus switch, called the ARC (ATM routing and concentration) chip, contains a two-dimensional array (32×32) of switch elements that are arranged in a crossbar structure. The chip has been designed and fabricated using 0.8 μm CMOS technology and tested to operate correctly at 240 MHz.

The performance of the Abacus switch under bursty traffic was presented. By engineering the expansion ratio (number of routing links/group size), the head-of-line blocking probability can be lowered arbitrarily so that the throughput of the input-output buffered switch approaches the output buffered switch. For a given expansion ratio, as the group size increases, the maximum throughput also increases. It shows that when the traffic load is below the maximum throughput, the input buffer's average delay is much smaller than the output buffer's average delay, e.g., by one order of magnitude, and the impact of the burstiness of input traffic on the input buffer's average delay is very small. It was also shown that multicasting has better throughput and smaller input-buffer delay than unicasting under uniform traffic distribution.

REFERENCES

[1] T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano, "32×32 shared buffer type ATM switch VLSI's for B-ISDN's," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1239–1247, Oct. 1991.

[2] Y. Shobatake, M. Motoyama, E. Shobatake, T. Kamitake, S. Shimizu, M. Noda, and K. Sakaue, "A one-chip scalable 8×8 ATM switch LSI employing shared buffer architecture," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1248–1254, Oct. 1991.

[3] T. R. Banniza, G. J. Eilenberger, B. Pauwels, and Y. Therasse, "Design and technology aspects of VLSI's for ATM switches," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1255–1264, Oct. 1991.

[4] A. Itoh, W. Takahashi, H. Nagano, M. Kurisaki, and S. Iwasaki, "Practical implementation and packaging technologies for a large-scale ATM switching system," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1280–1288, Oct. 1991.

[5] W. Fischer, O. Fundneider, E.-H. Goeldner, and K. A. Lutz, "A scalable ATM switching system architecture," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1299–1307, Oct. 1991.

[6] K. Y. Eng, M. A. Pashan, R. A. Spanke, M. J. Karol, and G. D. Martin, "A high-performance prototype 2.5 Gb/s ATM switch for broadband applications," in *Proc. ICC '89*, pp. 111–117.

[7] Y. Kato, T. Shimoe, and K. Murakami, "A development of a high speed ATM switching LSIC," in *Proc. ICC'90*, pp. 562–566.

[8] K. Genda, Y. Doi, K. Endo, T. Kawamura, and S. Sasaki, "A 160-Gb/s ATM switching system using an internal speed-up crossbar switch," in *Proc. GLOBECOM'94*, Nov. 1994, pp. 123–133.

[9] E. Munter, "A high capacity ATM switch based on advanced electronic and optical technologies," in *Proc. ISS'95*, Berlin, Germany, pp. 389–393.

[10] H. J. Chao, "A recursive modular terabit/sec ATM switch," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1161–1172, Oct. 1991.

[11] H. J. Chao and B. S. Choe, "Design and analysis of a large-scale multicast output buffered ATM switch," *IEEE/ACM Trans. Networking*, vol. 3, pp. 112–138, Apr. 1995.

[12] K. Y. Eng, M. J. Karol, and Y. S. Yeh, "A growable packet (ATM) switch architecture: Design principles and applications," *IEEE Trans. Commun.*, vol. 40, pp. 423–430, Feb. 1992.

[13] C.-K. Kim and T. T. Lee, "Call scheduling algorithms in a multicast switch," *IEEE Trans. Commun.*, vol. 40, pp. 625–635, Mar. 1992.

[14] A. Huang and S. Knauer, "STARLITE: A wideband digital switch," in *Proc. GLOBECOM'84*, pp. 121–125.

[15] J. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE J. Select. Areas Commun.*, vol. SAC-5, pp. 1264–1273, Oct. 1987.

[16] B. Bingham and H. Bussey, "Reservation-based contention resolution mechanism for batcher-banyan packet switches," *Electron. Lett.*, vol. 24, pp. 772–773, June 1988.

[17] A. Cisneros and C. A. Bracket, "A large ATM switch based on memory switches and optical star couplers," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 1348–1360, Oct. 1991.

[18] A. Pattavina, "Multichannel bandwidth allocation in a broadband packet switch," *IEEE J. Select. Areas Commun.*, vol. 6, pp. 1489–1499, Dec. 1988.

[19] H. J. Chao and N. Uzun, "An ASIC for a scalable multicast ATM switch," in *Proc. Design SuperCon'96*, Santa Clara, CA.

[20] R. Handel et al., *ATM Networks: Concepts, Protocols, Applications*. Reading, MA: Addison-Wesley, 1994, ch. 7.
[21] S. C. Liew and K. W. Lu, "Comparison of buffering strategies for asymmetric packet switch modules," *IEEE J. Select. Areas Commun.*, vol. 9, pp. 428–438, Apr. 1991.
[22] C.-Y. Chang, A. J. Paulraj, and T. Kailath, "A broadband packet switch architecture with input and output queueing," in *Proc. GLOBECOM'94*, San Francisco, CA, Nov. 1994, pp. 448–452.
[23] A. Pattavina and G. Bruzzi, "Analysis of input and output queueing for nonblocking ATM switches," *IEEE/ACM Trans. Networking*, vol. 1, pp. 314–328, June 1993.
[24] H. J. Chao and N. Uzun, "An ATM routing and concentration chip for a scalable multicast ATM switch," *IEEE J. Solid-State Circuits*, vol. 32, no. 6, pp. 816–828, June 1997.

H. Jonathan Chao (S'83–M'85–SM'95), for a photograph and biography, see this issue, p. 771.

Byeong-Seog Choe (S'90–M'95) was born in Seoul, Korea, in 1957. He received the B.S.E.E. degree from Seoul National University in 1985, the M.S.E.E. degree from Fairleigh Dickinson University, Rutherford, NJ, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from Polytechnic University, Brooklyn, NY, in 1993 and 1994, respectively.

From 1992 to 1993, he was a Research Fellow in the Center for Advanced Technology in Telecommunications, Polytechnic University, where he was involved in the design and performance analysis of ATM switches. From 1994 to 1997, he was on the faculty of the Department of Electronics Engineering, Myongji University, Korea. He is now Assistant Professor of Electronics Engineering at Dongkuk University, Seoul, Korea. His current research interests include congestion control in B-ISDN and photonic ATM switches.

Jin-Soo Park (S'96) received the B.S. degree from Seoul National University, Seoul, Korea, in 1986, and the M.S. degree from the Korea Advanced Institute of Science and Technology, Taejon, in 1988, both in electrical engineering. He is currently working toward the Ph.D. degree in electrical engineering at Polytechnic University, Brooklyn, NY.

From 1988 to 1992, he was a member of the Technical Staff at the Korea Telecom Research Center, where he worked on standardization, deployment plan in Korea, technical specification, and operation of SONET/SDH systems. Since 1993, he has been working as a Research Assistant in the Center for Advanced Technology in Telecommunications, Polytechnic University. His current research interests include the design and analysis of large-scale multicast ATM switch architecture, output contention resolution algorithms in input-buffered ATM switches, and congestion control in ATM networks.

Necdet Uzun (S'91–M'93) received the B.S. and M.S. degrees in electrical engineering from the Technical University of Istanbul, Istanbul, Turkey, in 1983 and 1986, respectively, and the Ph.D. degree in electrical engineering from Polytechnic University, Brooklyn, NY, in 1993.

He is an Industry Associate Professor of Electrical Engineering at Polytechnic University. Previously, he was with Bellcore, Red Bank, NJ, from 1990 to 1992, as a Consultant in the Multiplex and Multiaccess Technology Group. His R&D activities in high-speed networking include electronic and photonic ATM switches, admission and congestion control, ATM WAN's, and high-speed VLSI architectures of ATM switching systems. He has also been involved in the analysis and modeling of quantization effects in filter banks.