# CHAPTER 10

# THE ABACUS SWITCH

The switches based on the knockout concept suffer from cell loss due to the lack of routing links in the switch fabric, for example, the concentrator in the knockout switch or the multicast grouping network (MGN) in the MOBAS (see Chapter 9). Although we can engineer the group expansion ratio ($L$) to achieve a satisfactory cell loss probability, say $10^{-10}$, it is based on the assumption that the traffic from different input ports is uncorrelated and input traffic is uniformly distributed to all output ports. The latter assumption gives the worst case cell loss probability, while the former assumption may not be realistic for applications such as Internet Web services. There may be a lot of traffic destined for the same popular site at the same time, resulting in a so-called hot-spot situation and an unacceptable cell loss probability. In order to reduce the cell loss rate, excessive cells can be stored at the input buffers, which results in the switch having buffers at the input and output ports. The switch to be discussed in this chapter belongs to this category.

We describe a switch that has a similar architecture to that of the MOBAS but does not discard cells in the switch fabric. When the head-of-line (HOL) cells of the input ports are sent to the switch fabric, they are held at the input ports until they have been successfully transmitted to the desired output port. The switch fabric is a crossbar structure, where switch elements, with the capability of routing cells and resolving contention based on cells' priority levels, are arranged in a two-dimensional array and is similar to an abacus. That's why the switch is called the Abacus switch. The challenging issue of designing an input–output buffered switch is to design a fast and scalable arbitration scheme.

The arbitration algorithm proposed in the Abacus switch takes advantage of the capability that the switch element can resolve contention for the routing links based on their priority levels. As a result, with some extra feedback lines and logic circuits at the input ports, the arbitration scheme can be implemented without adding much complexity and cost. The switch uses a new arbitration scheme to resolve the contention among the HOL cells.

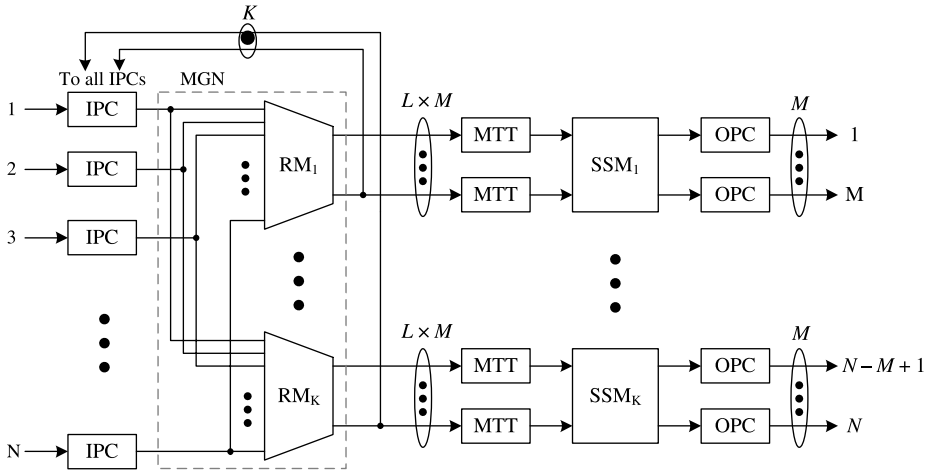
*High Performance Switches and Routers*, by H. Jonathan Chao and Bin Liu
Copyright © 2007 John Wiley & Sons, Inc.

The arbitration is done in a distributed manner and thus enables the switch to grow to a large size.

Section 10.1 describes the basic architecture of the Abacus switch. Section 10.2 presents the new arbitration scheme that is implemented in a distributed manner. Section 10.3 depicts the implementation of an input controller and how it resolves contention resolution. Section 10.4 discusses the performance of the Abacus in throughput, delay, and loss. Section 10.5 shows a key component, the ATM routing and concentration (ARC) chip used to implement the Abacus switch. Section 10.6 describes three approaches to scale the Abacus switch to 1-Tbit/s capacity. Section 10.7 shows how the Abacus switch can also route switch packets through the switch fabric.

## 10.1 BASIC ARCHITECTURE

The Abacus switch [1] is a scalable multicast architecture with input and output buffering. It uses input buffers to temporarily store cells that lost contention to other inputs and thus eliminates the possibility of discarding cells due to the loss of contention in the switch fabric, as is the case with MOBAS. Figure 10.1 shows the architecture of the Abacus switch. It consists of input port controllers (IPCs), a MGN, multicast translation tables (MTTs), small switch modules (SSMs), and output port controllers (OPCs). The architecture is very similar to that of the MOBAS with the exception that the MGN2 in the MOBAS is replaced with the SSM and the Abacus switch has feedback lines from the RMs to the IPCs to facilitate output port contention resolution (see Section 10.2 for details). The RM (routing module) in the Abacus switch is exactly the same as the SM in the MOBAS and the term 'RM' will be used from now on. If the group size, $M$, is carefully chosen in a way that the second stage switch network's capacity is within 20 Gbit/s, it will be more cost-effective to implement the MGN2 in the MOBAS with a shared-memory switch module. For instance, for $M = 32$,




IPC: Input port controller
RM: Routing module
MGN: Multicast grouping network
MTT: Multicast translation table
SSM: Small switch module
OPC: Output port controller

**Figure 10.1** Architecture of the Abacus switch (©1997 IEEE).

$L = 2$, and line rate = 155.52 Mbit/s, the SSM's capacity is 10 Gbit/s. The IPC performs similar functions as those in the MOBAS, except that it also assists in resolving contention among cells that are destined for the same output group and buffering those cells losing contention.

The switch performs cell replication and cell routing simultaneously. Cell replication is achieved by broadcasting incoming cells to all RMs, which then selectively route cells to their output links. Cell routing is performed distributedly by an array of switch elements (SWEs). The concept of channel grouping described in Section 9.2 is applied to construct the MGN in order to reduce hardware complexity, where all $M$ 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 providing $L \times M$ routing links to each output group. $L$ is defined as the 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 an arbitration mechanism to be described in Section 10.2, up to $L \times M$ cells from $N$ IPCs 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 RMs will be further routed to proper output port(s) through the SSMs.

The group expansion ratio $L$ is engineered in such a way that the required maximum throughput in a switch fabric can be achieved. Performance study shows that the larger $M$ is, the smaller $L$ is required to be 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. Since cell loss does not occur within the Abacus switch (unlike the MOBAS), $L$ is chosen to achieve sufficiently large maximum throughput and low delay in the input buffers, but not for cell loss rate as in the MOBAS. Its value can be slightly smaller than the one in the MOBAS (e.g., for $M = 32$, $L$ is 2 for a cell loss rate of $10^{-10}$).

Each RM in the MGN contains a two-dimensional array of switch elements and an address broadcaster (AB), as shown in Figure 10.2. It is similar to Figure 9.15 except that each RM provides a feedback line to all IPCs. The multicast pattern maskers (MPMs) are not shown here for simplicity.

Figure 10.3 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 MP is a bit map of all 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 $i$th bit in the MP is set to '1', the cell is to be sent to the $i$th output group. The 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 a 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

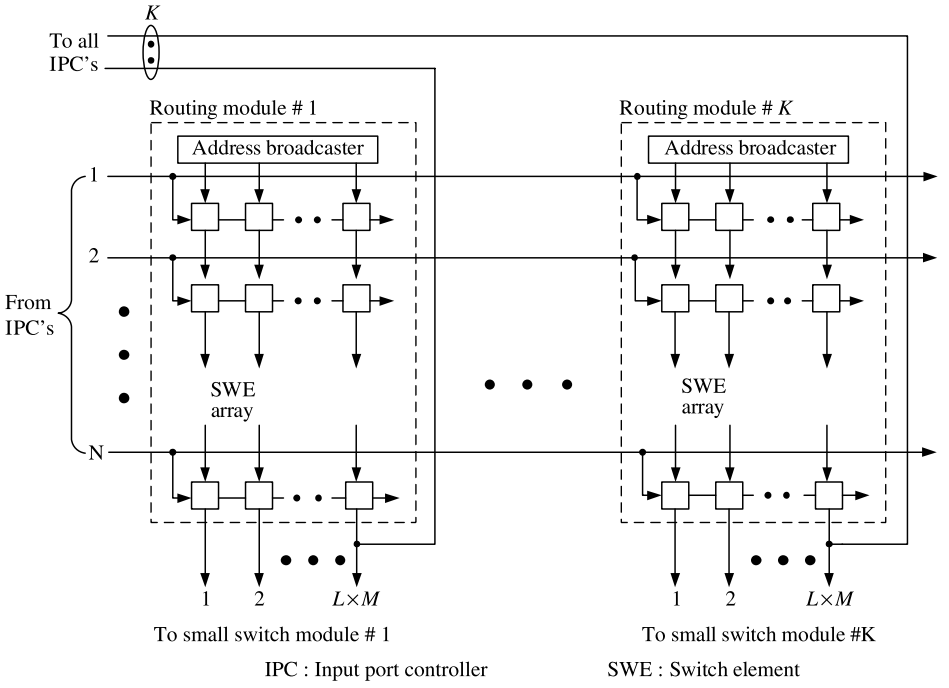
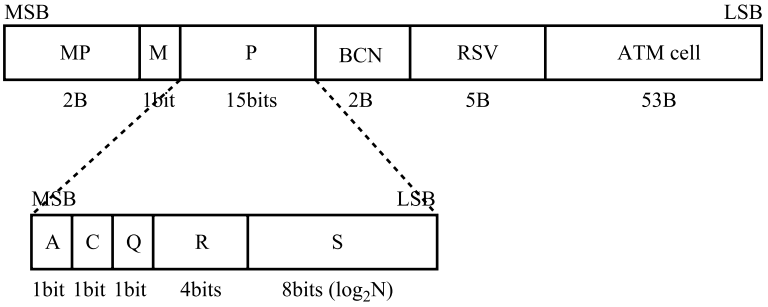



**Figure 10.2** Multicast grouping network (MGN) (©1997 IEEE).

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 a global first-come-first-served (FCFS) discipline, allowing




MP: Multicast pattern
P: Priority for contention resolution
A: Activity bit
C: Connection priority
Q: Buffer state priority
R: Retry priority
S: Input port priority
M: Multicast bit
BCN: Broadcast channel number
RSV: Reserved field for future growth
MSB: Most significent bit
LSB: Least significent bit

**Figure 10.3** Routing information used by Abacus switch with $N = 256$, $M = 16$.

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 HOL 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 BCN in Figure 10.3 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 virtual path identifier/virtual channel identifier (VPI/VCI) for each copy of the replicated cell.

## 10.2 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 perform call splitting for multicasting and thus improves the system throughput. 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 [2], ring reservation [3], and centralized contention resolution device [4].

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, 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 in the order of $O(N \times K)$. The arbiter may become the system's bottleneck when $N$ or $K$ is large.

The arbitration scheme described here performs $N$-to-$L \times M$ selection in a distributed manner using the switch fabric and all IPCs, thus eliminating the speed constraint. Another difference between this arbitration scheme and others is that here the HOL cell is repeatedly sent to the switch fabric to compete with others until it has successfully transmitted to all necessary output groups that the cell is destined for. Unlike other arbitration schemes, the scheme described here 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 the IPC knows whether or not its HOL cell has been successfully transmitted to all necessary output groups. In the Abacus switch, the RMs are responsible for returning the routing results to the IPC. One possible way is to let each RM inform the IPCs of 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 $K$ acknowledgments from $K$ RMs. The complexity of returning the identification of every successfully routed copy to all IPCs is too high to be practical for a large-scale switch. A scheme that significantly simplifies the complexity of the acknowledgment operation is described in the following.

The RM cannot only route cells to proper output groups, but also, based on cells' priority levels, 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. 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. 10.2). 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 IPCs. 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, $FP_1, \ldots, FP_K$. 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 $K$ RMs in total, there will be $K$ lines broadcast from $K$ RMs to all IPCs, each carrying the lowest priority information in its output group.

The priority assigned to the HOL cells will be dynamically changed according to some arbitration policies, such as random, round-robin, state-dependent, or delay-dependent [5]. The random scheme randomly chooses the HOL cells of input ports for transmission; the drawback being 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. 10.3). 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. 10.3). 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. 10.3). Since the arbitration is performed in a distributed manner by $K$ RMs and in parallel by IPCs, any of the above policies, or a combination of them, can be implemented 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 sized buffer during its transmission. After cells have traversed through the RMs, priority information, $FP_1$ to $FP_K$ (the priority of the right most link of each RM), is fed back to every IPC. Each IPC will then compare the feedback priority level $FP_j, j = 1, 2, \ldots, K$, with its local priority level, $LP$. Three situations can happen: (a) $MP_j = 1$ and $LP \leq FP_j$ (recall that the smaller the priority value, the higher the priority level), which means the HOL cell is destined for the $j$th output group and has been successfully routed through the $j$th RM. The $MP_j$ bit is then set to '0'; (b) $MP_j = 1$ and $LP > FP_j$, which means the HOL cell is destined for the $j$th output group but discarded in the $j$th RM. The $MP_j$ bit remains '1'; (c) $MP_j = 0$, the $j$th bit of the HOL cell's multicast pattern can be equal to '0', which means the HOL cell is not destined for the $j$th output group. Then, the $MP_j$ bit remains '0'.

After all $MP_j$ bits ($j = 1, 2, \ldots, K$) 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 bits in the multicast pattern remains '1'. The resend

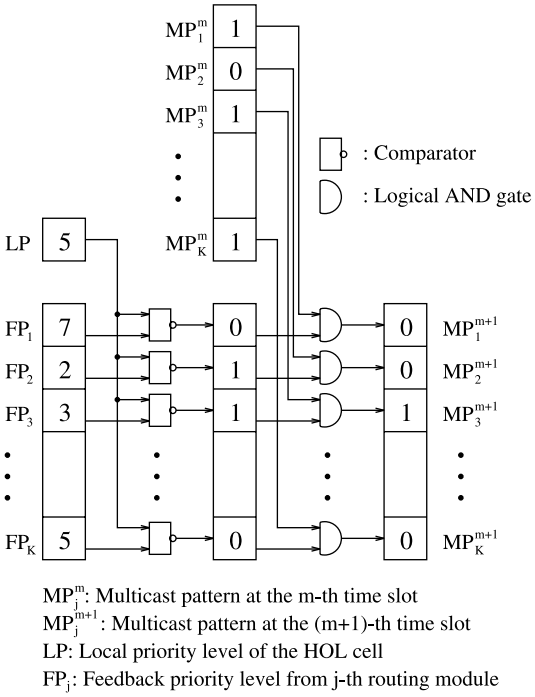



**Figure 10.4** Example of modifying a multicast pattern (MP) (©1997 IEEE).

signal is initially set to '0'. If multicast pattern bits are all '0', meaning the HOL cell has been successfully transmitted to all the 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).

Figure 10.4 gives an example of how a multicast pattern is modified. Let us assume at the beginning of the $m$th time slot, the HOL cell is destined for three output groups: #1, #3, #$K$. Therefore, the multicast pattern at the $m$th time slot, $MP^m$, has three bits set to '1'. Let us also assume the local priority value ($LP$) of the HOL cell is 5 and the feedback priority values from #1, #2, #3, and #$K$ are 7, 2, 3, and 5, respectively, as shown in Figure 10.4. The result of comparing $LP$ with $FP$s is '0110...00', which is then logically ANDed with the $MP^m$ and produces a new multicast pattern, '0010...00', 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 RMs #1 and #$K$ but discarded in RM #3 and will retransmit in the next time slot.

## 10.3 IMPLEMENTATION OF INPUT PORT CONTROLLER

Figure 10.5 shows a block diagram of the IPC. For easy explanation, let us assume 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

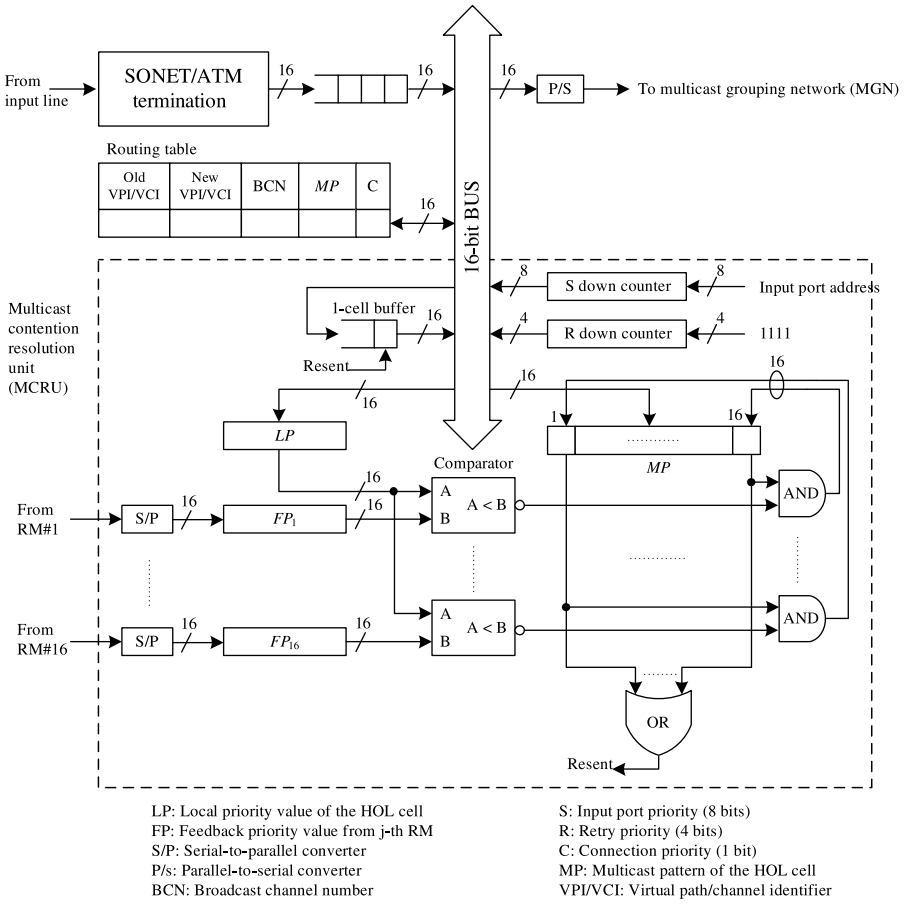



**Figure 10.5** Implementation of the IPC with $N = 256, M = 16$ (© 1997 IEEE).

$K$ 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 with 16 bits wide are written into an input buffer. The HOL cell's VPI/VCI is used to extract the necessary information from a routing table. This information includes a new VPI/VCI for unicast connections, a BCN for multicast connections, which uniquely identifies each multicast call in the entire switch, 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 Figure 10.3.

As the cell is transmitted to the MGN through a parallel-to-serial (P/S) converter, the cell is also stored temporarily in a one-cell buffer. If the cell fails to successfully route through the RMs, 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 (S/P) converters 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, 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 $j$th 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 Figure 10.5, there are $K$ sets of S/P, $FP$ register, and comparator. As a switch size increases, the number of output groups, $K$, also increases. However, if the time permits, only one set of this hardware is required by time-division multiplexing the operation of comparing the local priority value, $LP$, with $K$ feedback priority values.

## 10.4 PERFORMANCE

This section discusses the performance analysis of the Abacus switch. Both simulation and analytical results are shown to compare with each other. Simulation results are obtained with a 95 percent confidence interval, not greater than 10 percent for the cell loss probability or 5 percent for the maximum throughput and average cell delay.

### 10.4.1 Maximum Throughput

The maximum throughput of an ATM switch employing input queuing is defined by the maximum utilization at the output port. An input-buffered switch's HOL blocking problem can be alleviated 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 ($N$) or burstiness ($\beta$), the smaller the maximum throughput ($\rho_{\max}$) will be. However, the larger the group expansion ratio ($L$) or group size ($M$) is, the larger the maximum throughput will be.

Karol et al. [6] have shown that the maximum throughput of an input-buffered ATM switch is 58.6 percent for $M = 1$, $L = 1$, $N \to \infty$, with random traffic. Oie et al. [7] have obtained the maximum throughput of an input–output-buffered ATM switch for $M = 1$, an arbitrary group expansion ratio or speed-up factor $L$, and an infinite $N$ with random traffic. Pattavina [8] has shown, through computer simulations, the maximum throughput of an input–output-buffered ATM switch using the channel grouping concept for an arbitrary group size $M$, $L = 1$, and an infinite $N$ with random traffic. Liew and Lu [9] have shown the maximum throughput of an asymmetric input–output-buffered switch module for arbitrary $M$ and $L$, and $N \to \infty$ with bursty traffic.

Figure 10.6 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 that proposed by Kozaki et al. [10]. Although it can achieve 100 percent

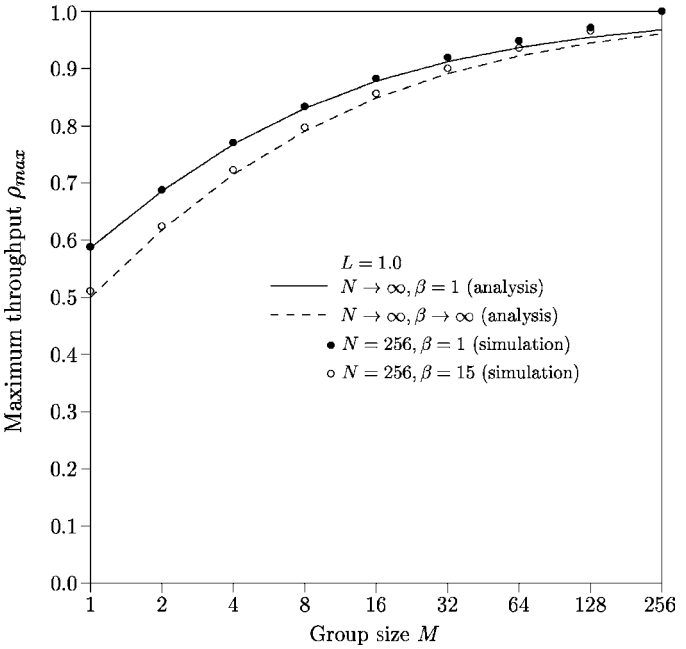



**Figure 10.6** Maximum throughput versus group size, $L = 1.0$.

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.

Figures 10.7 and 10.8 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 and Lu's analysis [9].

A HOL virtual queue is defined as the queue that consists of cells at the head of line of input buffers destined for a tagged output group. For uniform random traffic, the average number of cells $E[C]$ in the HOL virtual queue becomes

$$E[C] = \frac{\rho_o[2(L \times M) - \rho_o] - (L \times M)(L \times M - 1)}{2(L \times M - \rho_o)} + \sum_{k=1}^{L \times M - 1} \frac{1}{1 - z_k} \tag{10.1}$$

where $z_0 = 1$ and $z_k$ $(k = 1, \ldots, L \times M - 1)$ are the roots of $z^{L \times M}/A(z) = [p + (1 - p)z]^{L \times M}$ and $A(z) = \mathrm{e}^{-p\rho_o(1-z)}$. For completely bursty traffic, $E[C]$ becomes

$$E[C] = \frac{\sum_{k=0}^{L \times M - 1} k(L \times M - k)c_k + \rho_o}{L \times M - \rho_o} \tag{10.2}$$

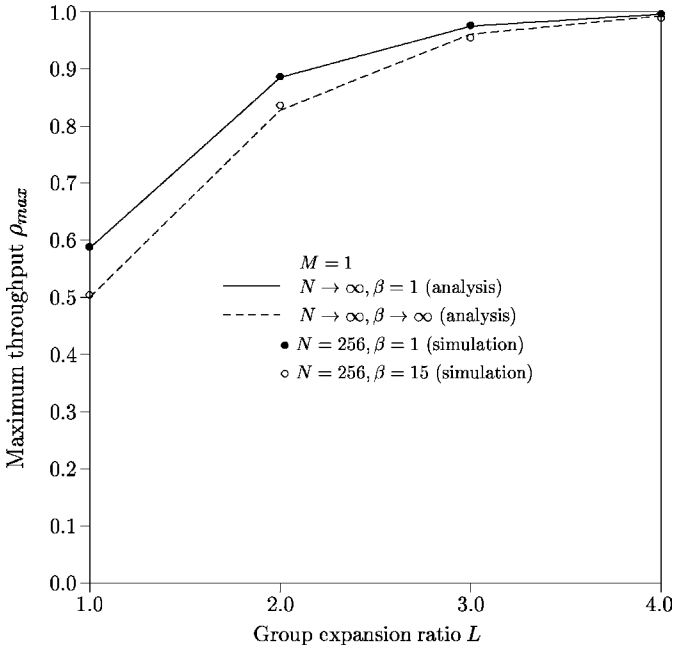



**Figure 10.7** Maximum throughput versus group expansion ratio with $M = 1$.

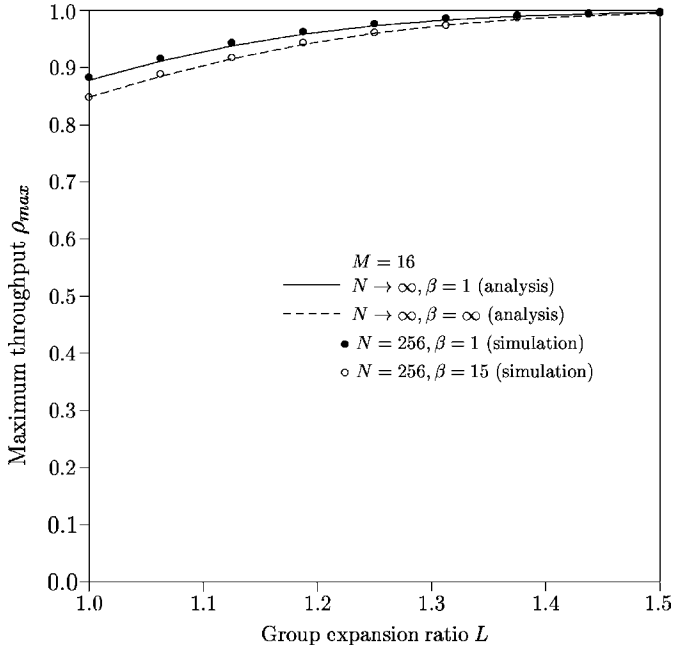



**Figure 10.8** Maximum throughput versus group expansion ratio with $M = 16$.

where

$$c_k = \begin{cases} \rho_o^k c_0/k! & \text{if } k < L \times M \\ \rho_o^k c_0/[(L \times M)!(L \times M)^{k-L\times M}] & \text{if } k \geq L \times M \end{cases}$$

$$c_0 = \frac{L \times M - \rho_o}{\sum_{k=0}^{L\times M-1}(L \times M - k)\rho_o^k/k!}.$$

The maximum throughput of the proposed ATM switch can be obtained by considering the total number of backlogged cells in all $K$ HOL virtual queues to be $N$. This means under a saturation condition, there is always a HOL cell at each input queue. Since it is assumed that cells are to be uniformly distributed over all output groups, we obtain

$$E[C] = N/K = M. \tag{10.3}$$

The maximum throughput can be obtained by equating (10.1) and (10.3) for random traffic, and equating (10.2) and (10.3) for bursty traffic, respectively. If $\rho_o^*$ satisfies both equations, the maximum throughput at each input, $\rho_{\max}$, is related to $\rho_o^*$ by $\rho_{\max} = \rho_o^*/M$.

For a given $M$, the maximum throughput increases with $L$ because there are more routing links available between input ports and output groups. However, since a larger $L$ means higher hardware complexity, the value of $L$ should be selected prudently such that both hardware complexity and the maximum throughput are acceptable. For instance, for a group size $M$ of 16 (for input traffic with an average burst length of 15 cells), $L$ has to be at least 1.25 to achieve a maximum throughput of 0.95. But for a group size $M$ of 32 and the same input traffic characteristic, $L$ will be at least 1.125 to achieve the same throughput. Both analytical results and simulation results show that the impact of input traffic's burstiness on the maximum throughput is very small. For example, the maximum throughput for uniform random traffic with $M = 16$ and $L = 1.25$ is 97.35 percent, and for completely bursty traffic is 96.03 percent, only 1.32 percent difference. The discrepancy between theoretical and simulation results comes from the assumption of switch size $N$ and $\beta$. In theoretical analysis, it is assumed that $N \to \infty$, $\beta \to \infty$, but in simulation it is assumed that $N = 256$, $\beta = 15$. As these two numbers increase, the discrepancy reduces.

### 10.4.2 Average Delay

A cell may experience two kinds of delay while traversing through the Abacus switch: input-buffer delay and output-buffer delay. In theoretical analysis, the buffer size is assumed to be infinite. But in simulations, it is assumed that the maximum possible size for the input buffers and output buffers that can be sustained in computer simulations: $B_i = 1024$ and $B_o = 256$, respectively. Here, $B_i$ is the input buffer size and $B_o$ is the normalized output buffer size (i.e., the size of a physical buffer 4096 divided by $M$, 16). Although finite buffers may cause cell loss, it is small enough to be neglected when evaluating average delay.

Let us assume each input buffer receives a cell per time slot with a probability $\lambda$, and transmits a cell with a probability $\mu$. The probability $\mu$ is equal to 1.0 if there is no HOL blocking. But, as the probability of HOL blocking increases, $\mu$ will decrease. If $\lambda > \mu$, then the input buffer will rapidly saturate, and the delay at the input buffer will be infinite. The analytical

results for uniform random traffic can be obtained from the analysis of Chang et al. [11],

$$E[T] = \frac{1-\lambda}{\mu-\lambda}, \tag{10.4}$$

where $\lambda = \rho_i$, $\mu = \rho_i/E[C]$, and $E[C]$ is a function of $M$, $L$, $\rho_i$, and $\beta$ as $N \to \infty$, which can be obtained from (10.1) or (10.2).

Note that the input buffer's average delay is very small when the input offered load is less than the maximum saturation throughput. This results from small HOL blocking probability before the saturated throughput. It also shows that the impact of the burstiness of input traffic on the input buffer's average delay is very small when the traffic load is below the maximum throughput.

Figure 10.9 compares the average delay at the input and output buffers. 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 $\rho_i$ of 0.8 and an average burst length $\beta$ 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.

Figure 10.10 shows simulation results of the input buffer's average delay versus the expanded throughput $\rho_j$ for both unicast and multicast traffic. It is assumed that the number of replicated cells is distributed geometrically with an average of $c$. The expanded throughput $\rho_j$ is measured at the inputs of the SSM and normalized to 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 10 unicast cells and the other input port $j$ has a multicast cell with a fanout of 10. Input port $i$ will take at least 10 time

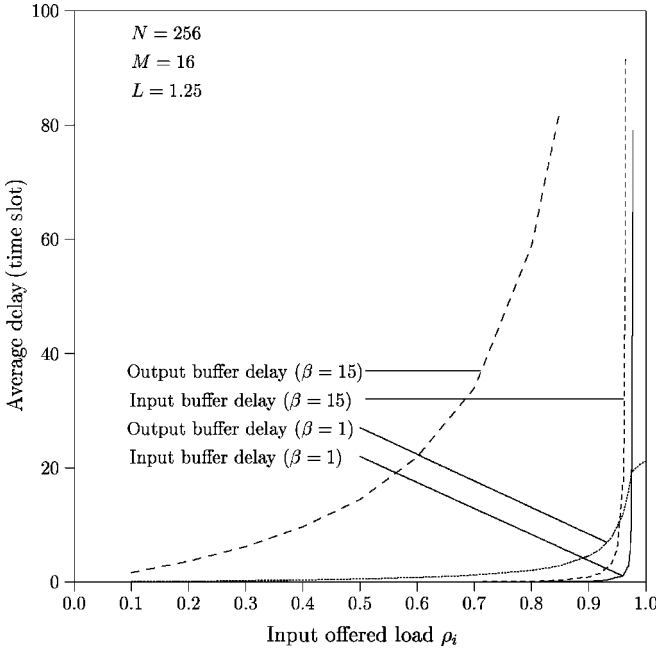



**Figure 10.9** Comparison of average input buffer delay and average output buffer delay.

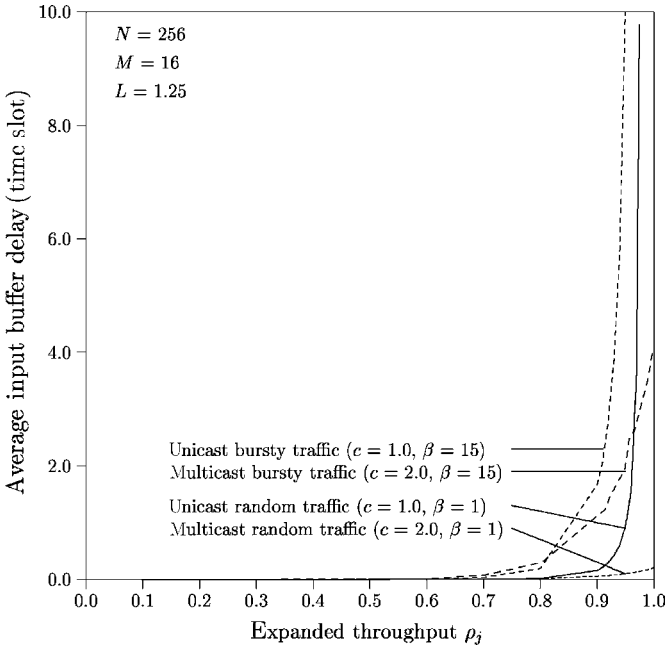



**Figure 10.10** Average input buffer delay versus expanded throughput for unicast and multicast traffic (simulation).

slots to transmit the 10 unicast cells while input port $j$ can possibly transmit the multicast cell in one time slot.

### 10.4.3 Cell Loss Probability

As suggested in the work of Pattavina and Bruzzi [12], there can be two buffer control schemes for an input–output-buffered switch: the queue loss (QL) scheme and the back-pressure (BP) scheme. In the QL scheme, cell loss can occur at both input and output buffers. 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 AB in Figure 10.2 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 changed 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. Figure 10.11 shows input buffer overflow probabilities with different average burst lengths, $\beta$. 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 $\beta$ of 15, the required

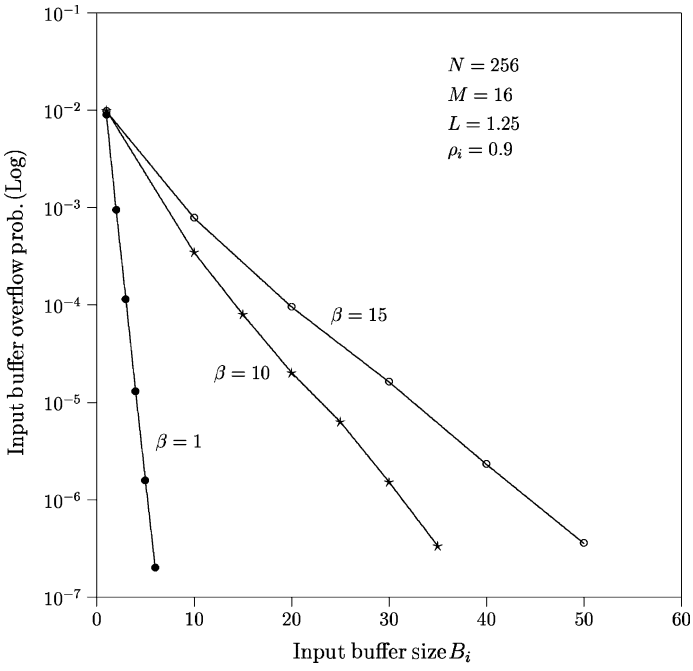



**Figure 10.11** Input buffer overflow probability versus input buffer size (simulation).

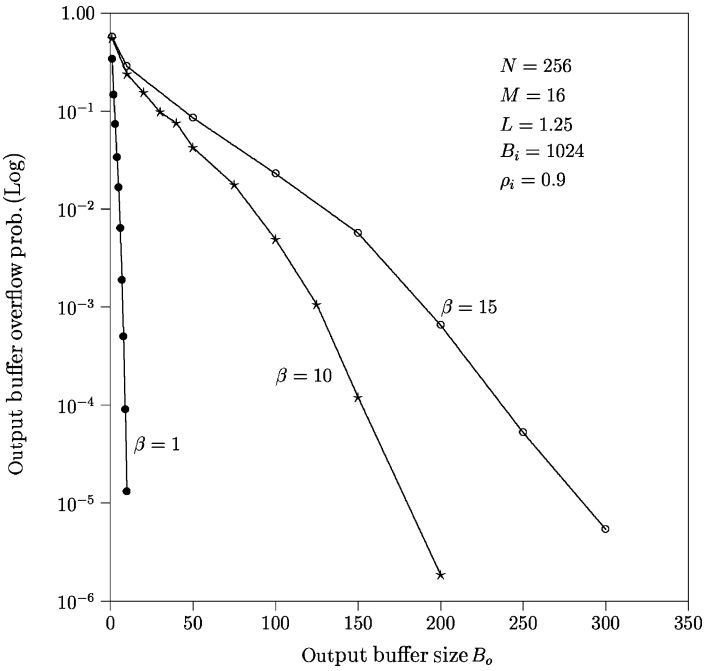



**Figure 10.12** Output buffer overflow probability versus output buffer size (simulation).

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.

Figure 10.12 shows the output buffer overflow probabilities with different average burst lengths. Here, $B_o$ is the normalized buffer size for each output. It is shown that the required output buffer size is much larger than the input buffer size for the same cell loss probability.

## 10.5 ATM ROUTING AND CONCENTRATION (ARC) CHIP

An ASIC (Application Specific Integrated Circuit) has been implemented based on the Abacus switch architecture. Figure 10.13 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 the work of Chao and Uzun [13]. The ARC chip contains $32 \times 32$ SWEs, which are partioned into eight SWE arrays, each with $32 \times 4$ SWEs. A set of input data signals, $w[0:31]$, comes from the IPCs. 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 go to the north input of the chips one row below or go to the output buffer.

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

$m[0:1]$ signals are used to configure the chip into four different group sizes as shown in Table 10.1: (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 0, while for the multicast case, $m[2]$ is set to 1.

As shown in Figure 10.14, the SWEs are arranged in a cross-bar structure, where signals only communicate between adjacent SWEs, 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

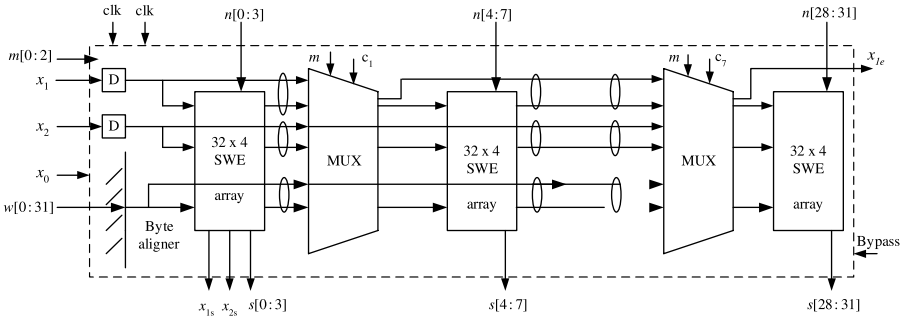



**Figure 10.13** Block diagram of the ARC chip.

**TABLE 10.1 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.

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. $x_1$ and $x_2$ signals are passed to the neighbor SWEs (east and south) after one clock cycle delay, as are data signals ($w$ and $n$). The $x_0$ signal is broadcast to all SWEs (not shown in Fig. 10.14) 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 Figure 10.15. Two bit-aligned cells, one from the west and one from the north, are applied to the 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, that is, 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 SWEs are 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.

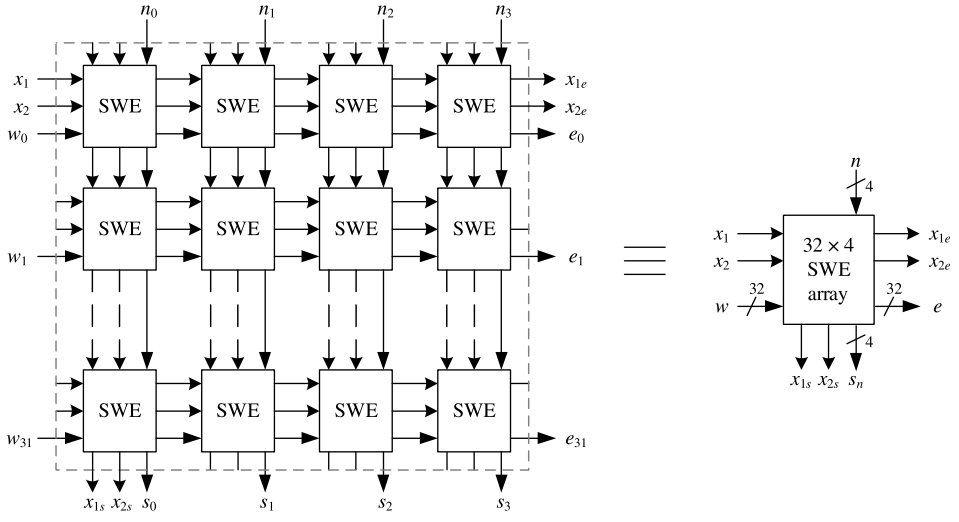



**Figure 10.14** 32 × 4 SWE array.

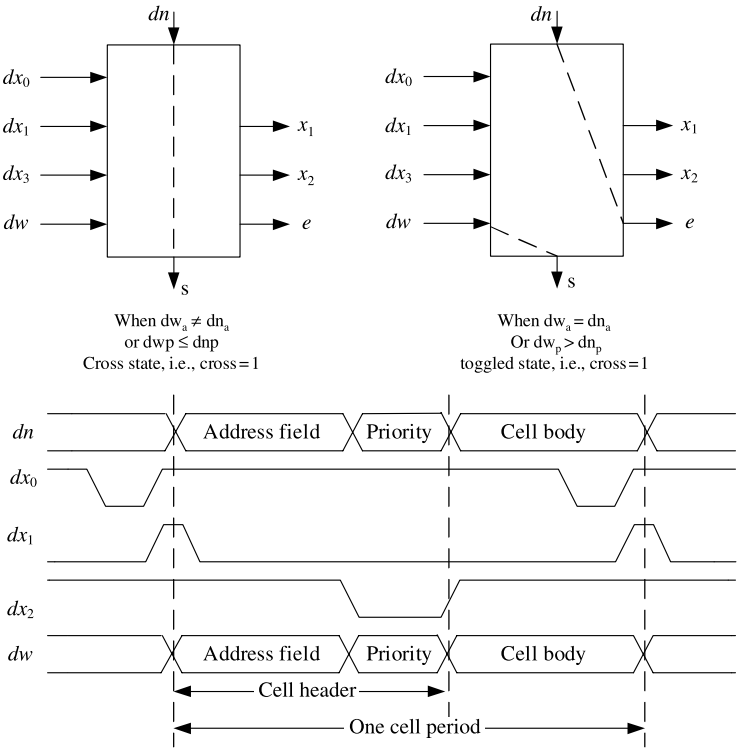



**Figure 10.15** Two states of the switch element.

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 and its characteristics are summarized in Table 10.2. Its photograph is shown in Figure 10.16.

**TABLE 10.2 Chip Summary**

| | |
|---|---|
| Process technology | 0.8-μm CMOS, triple metal |
| Number of switching elements | 32 × 32 |
| 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 × 6.6 $mm^2$ |
| Clock signals | Pseudo ECL |
| Interface signals | TTL/CMOS inputs, CMOS outputs |
| Maximum clock speed | 240 MHz |
| Worst-case power dissipation | 2.8 W at 240 MHz |

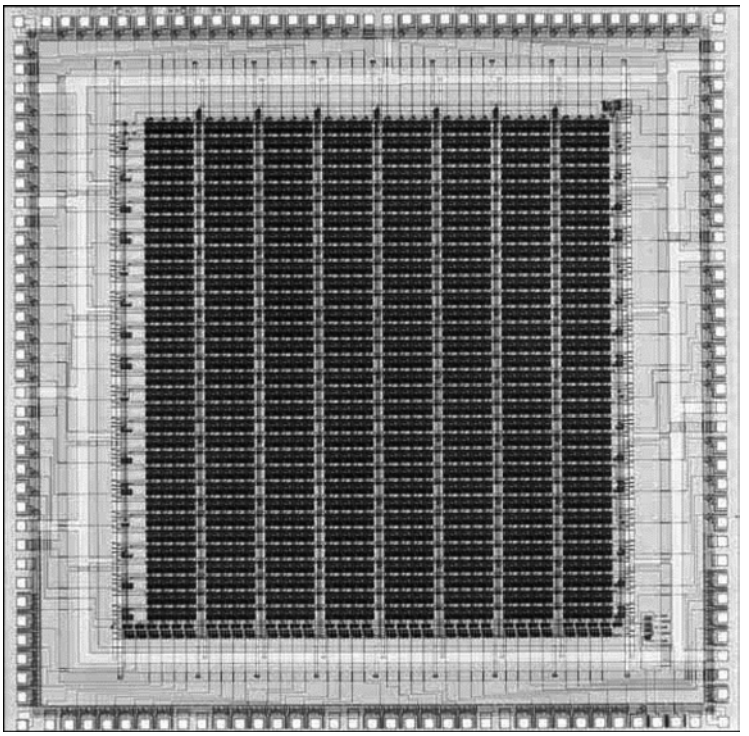


**Figure 10.16** Photograph of the ARC chip (©1997 IEEE).

## 10.6 ENHANCED ABACUS SWITCH

This section discusses three approaches of implementing the MGN in Figure 10.1 to scale-up the Abacus switch to a larger-size. As described in Section 10.2, the time for routing cells through an RM and feeding back the lowest priority information from the RM to all IPCs must be less than one cell slot time. The feedback information is used to determine whether or not the cell has been successfully routed to the destined output group(s). If not, the cell will continue retrying until it has reached all the desired output groups. Since each SWE in an RM introduces a 1-bit delay as the signal passes it in either direction, the number of SWEs between the uppermost link and the rightmost link of an RM should be less than the number of bits in a cell. In other words, $N + L \times M - 1 < 424$ (see Section 10.2). For example, if we choose $M = 16$, $L = 1.25$, the equation becomes $N + 1.25 \times 16 - 1 < 424$. The maximum value of $N$ is 405, which is not large enough for a large-capacity switch.

### 10.6.1 Memoryless Multi-Stage Concentration Network

One way to scale up the Abacus switch is to reduce the time spent on traversing cells from the uppermost link to the rightmost link in a RM (see Fig. 10.2). Let us call this time the 'routing delay'. In a single-stage Abacus switch, the routing delay is $N + L \times M - 1$, which limits the switch size because it grows with $N$.

To reduce the routing delay, the number of SWEs that a cell traverses in a RM must be minimized. If we divide a MGN into many small MGNs, the routing delay can be reduced. Figure 10.17 shows a two-stage memoryless multi-stage concentration network (MMCN) architecture that can implement a large-capacity Abacus switch. It consists of $N$ IPCs, $J$ $(=N/n)$ MGNs, and $K$ $(=N/M)$ concentration modules (CMs). Each MGN has $K$ RMs, and each RM has $n$ input links and $L \times M$ output links. Each CM has $J \times (L \times M)$ input links and $L \times M$ output links.

After cells are routed through the RMs, they need to be further concentrated at the CMs. Since cells that are routed to the CM always have correct output group addresses, we do not need to perform a routing function in the CM. In the CM, only the concentration function is performed by using the priority field in the routing information. The structure and implementation of the RM and the CM are identical, except the functions performed are slightly different.

Recall that each group of $M$ output ports requires $L \times M$ routing links to achieve a high delay/throughput performance. The output expansion ratio of the RM must be equal to or greater than that of the CM. If not, the multicast contention resolution algorithm does not work properly. For example, let us assume that $N = 1024$, $M = 16$, and $n = 128$. Consider the case that there are 16 links between a RM and a CM, while there are 20 links between

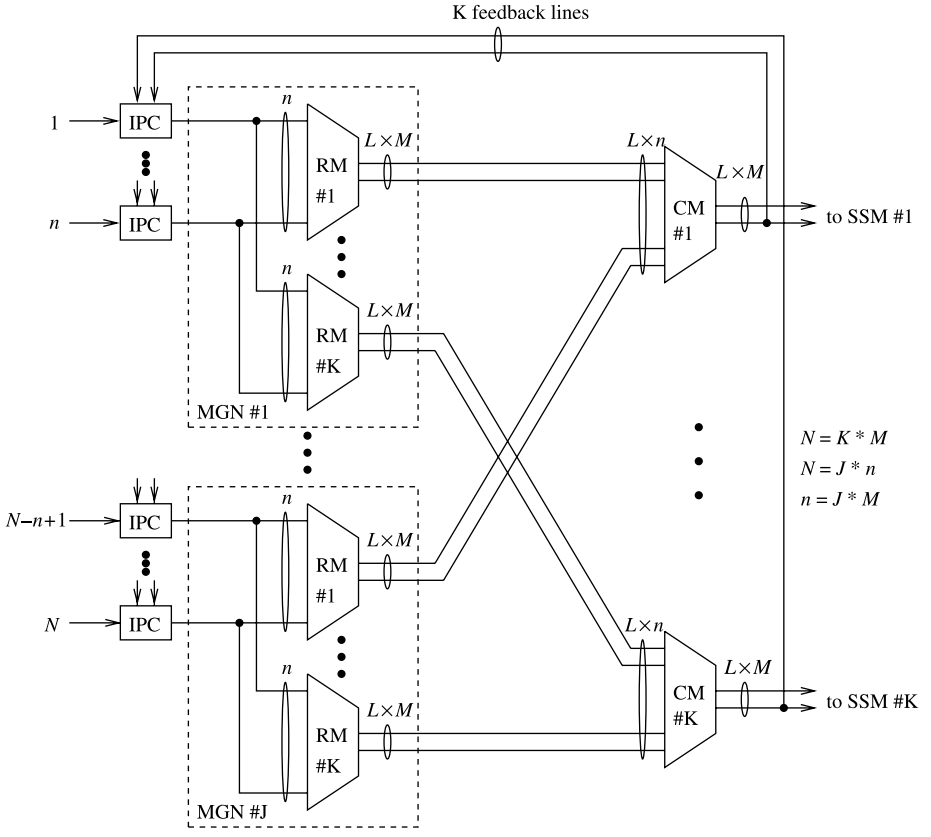



**Figure 10.17** Two-stage MMCN.

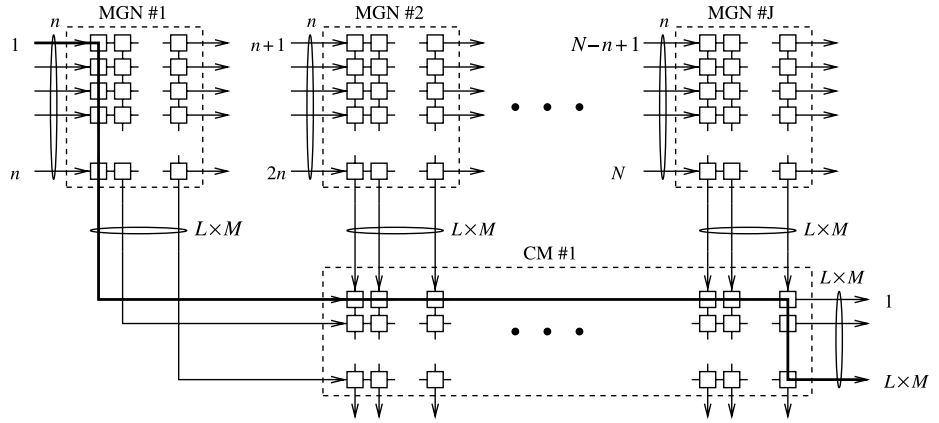



**Figure 10.18** Routing delay in a two-stage MMCN.

a CM and a SSM. If all 128 cells of MGN #1 are destined for output group #1 and no cells from other MGNs are destined for output group #1, the feedback priority of CM #1 will be the priority of the address broadcaster, which has the lowest priority level. Then, all 128 cells destined for output group #1 are cleared from the IPCs of MGN #1, even though only 20 cells can be accepted in the SSM. The other 108 cells will be lost. Therefore, the output expansion ratio of the RM must be equal to or greater than that of the CM.

Let us define $n$ as the module size. The number of input links of a RM is $n$, and the number of input links of the CM is $J \times (L \times M)$. By letting $n = J \times M$, the number of input links of the CM is on the same order with the number of input links of the RM because we can engineer $M$ such that $L$ is close to one.

In the MMCN, the feedback priorities (FPs) are extracted from the CMs and broadcast to all IPCs. To maintain the cell sequence integrity from the same connection, the cell behind the HOL cell at each IPC cannot be sent to the switch fabric until the HOL cell has been successfully transmitted to the desired output port(s). In other words, the routing delay must be less than one cell slot. This requirement limits the MMCN to a certain size.

Cells that have arrived at a CM must carry the address of the associated output group (either valid cells or dummy cells from the RM's address broadcaster). As a result, there is no need for using the AB in the CM to generate dummy cells to carry the address of the output group. Rather, the inputs that are reserved for the AB are substituted by the routing links of MGN #1. Thus, the routing delay of the two-stage MMCN is $n + (J - 1) \times (L \times M) + L \times M - 1$, as shown in Figure 10.18, which should be less than 424. Therefore, we have the following equations by replacing $J$ with $n/M$: $n + [(n/M) - 1] \times (L \times M) + L \times M - 1 < 424$. It can be simplified to $n < [425/(1 + L)]$. Thus, $N = J \times n = n^2/M < \{425^2/[M \times (1 + L)^2]\}$. Table 10.3 shows the minimum value of $L$ for a given $M$ to get a maximum throughput of 99 percent with random uniform traffic.

**TABLE 10.3 Minimum Value of $L$ for a Given $M$**

| $M$ | 1 | 2 | 4 | 8 | 16 | 32 |
|---|---|---|---|---|---|---|
| $L$ | 4 | 3 | 2.25 | 1.75 | 1.25 | 1.125 |

Clearly, the smaller the group size $M$, the larger the switch size $N$. The largest Abacus switch can be obtained by letting $M = 1$. But in this case ($M = 1$), the group expansion ratio ($L$) must be equal to or greater than four to have a satisfactory delay/throughput performance. Increasing the group size $M$ reduces the maximum switch size $N$, but also reduces the number of feedback links ($N^2/M$) and the number of SWEs ($LN^2 + L^2Nn$). Therefore, by engineering the group size properly, we can build a practical large-capacity Abacus switch. For example, if we choose $M = 16$ and $L = 1.25$, then the maximum module size $n$ is 188, and the maximum switch size $N$ is 2209. With the advanced CMOS technology (e.g., 0.25 μm), it is feasible to operate at the OC-12 rate (i.e., 622 Mbit/s). Thus, the MMCN is capable of providing more than 1 Tbit/s of capacity.

### 10.6.2 Buffered Multi-Stage Concentration Network

Figure 10.19 shows a two-stage buffered multi-stage concentration network (BMCN). As discussed in the previous section, the MMCN needs to have the feedback priority lines connected to all IPCs, which increases the interconnection complexity. This can be resolved by keeping RMs and CMs autonomous, where the feedback priorities (FPs) are extracted

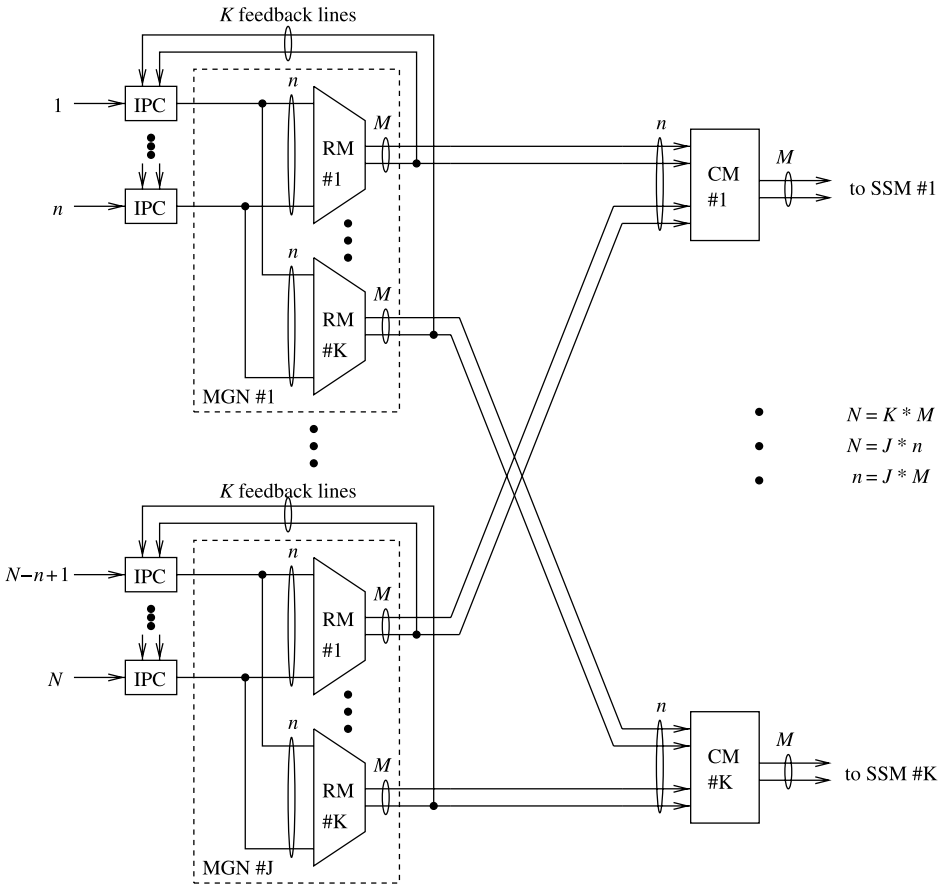



**Figure 10.19** Two-stage BMCN.

from the RMs rather than from the CMs. However, buffers are required in the CMs since cells that successfully pass through the RMs and are cleared from input buffers may not pass through the CMs.

Figure 10.20 shows three ways of building the CM. Figure 10.20*a* uses a shared-memory structure similar to the MainStreet*Xpress* 36190 core services switch [14] and the concentrator-based growable switch architecture [15]. Its size is limited due to the memory speed constraint.

Figure 10.20*b* shows another way to implement the CM by using a two-dimensional SWE array and input buffers. One potential problem of having buffers at the input links of the CM is cell out-of-sequence. This is because after cells that belong to the same virtual connection are routed through a RM, they may be queued at different input buffers. Since the queue lengths in the buffers can be different, cells that arrive at the buffer with shorter queue lengths will be served earlier by the CM, resulting in cell out-of-sequence.

This out-of-sequence problem can be eliminated by time-division multiplexing cells from a RM ($M$ cells), storing them in an intermediate stage controller (ISC), and sending them sequentially to $M$ one-cell buffers, as shown in Figure 10.20*c*. The ISC has an internal FIFO buffer and logic circuits that handle feedback priorities as in the Abacus switch. This achieves a global FIFO effect and thus maintains the cells' sequence. Each ISC can receive up to $M$ cells and transmit up to $M$ cells during each cell time slot.

The key to maintaining cell sequence is to assign priority properly. At each ISC, cells are dispatched to the one-cell buffers whenever the one-cell buffers become empty and there are cells in the ISC. When a cell is dispatched to the one-cell buffer, the ISC assigns a priority value to the cell. The priority field is divided into two parts, port priority and sequence priority. The port priority field is more significant than the sequence priority. The port priority field has $\lceil \log_2 J \rceil$ bits for the $J$ ISCs in a CM, where $\lceil x \rceil$ denotes the smallest integer that is equal to or greater than $x$. The sequence priority field must have at least $\lceil \log_2 JM \rceil$ bits to ensure the cell sequence integrity to accommodate $L \times M$ priority levels, which will be explained in an example later. The port priority is updated in every arbitration cycle (i.e., in each cell slot time) in a round-robin fashion. The sequence priority is increased by one whenever a cell is dispatched from the ISC to a one-cell buffer. When the port priority has the highest priority level, the sequence priority is reset to zero at the beginning of the next arbitration cycle (assuming the smaller the priority value, the higher the priority level). This is because all cells in the one-cell buffers will be cleared at the current cycle. Using this dispatch algorithm, cells in the ISC will be delivered to the output port in sequence. The reason that the sequence priority needs to have $L \times M$ levels is to

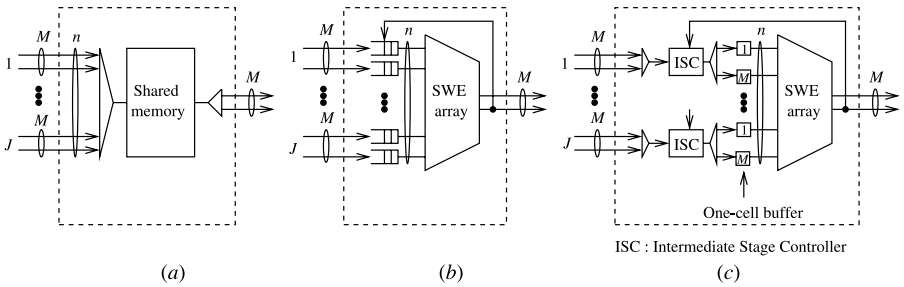



**Figure 10.20** Three ways of building the CM.

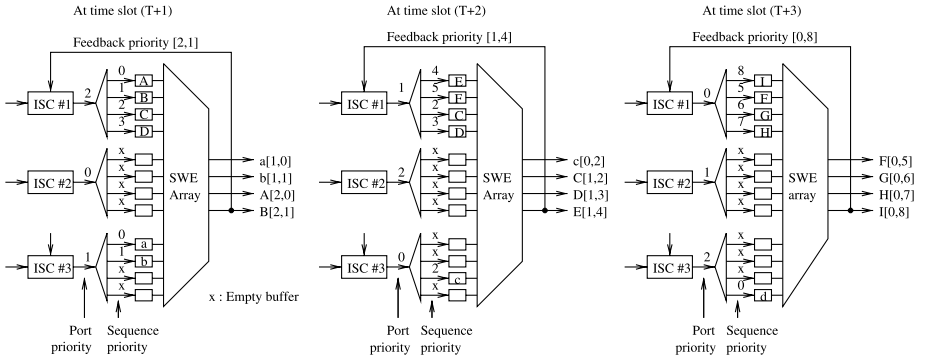



**Figure 10.21** Example of priority assignment in the CM.

accommodate the maximum number of cells that can be transmitted from an ISC between two reset operations for the sequence priority.

Figure 10.21 shows an example of a cell's priority assignment scheme for $J = 3$ and $M = 4$. Port priority ($p$) and sequence priority ($q$) are represented by two numbers in $[p, q]$, where $p = 0, 1, 2$, and $q = 0, 1, 2, \ldots, 11$. During each time slot, each ISC can receive up to four cells and transmit up to four cells. Let us consider the following case. ISC #1 receives four cells in every time slot, ISC #3 receives one cell in every time slot, and ISC #2 receives no cells.

The port priority is changed in every time slot in a round-robin fashion. In time slot $T$, ISC #1 has the highest port priority level, and ISC #3 has the lowest port priority level. In time slot $(T + 1)$, ISC #2 has the highest port priority level, and ISC #1 has the lowest port priority level, and so on. ISC #3 has a higher port priority level than ISC #1 in time slots $(T + 1)$ and $(T + 2)$.

In time slot $T$, all four cells in the one-cell buffers of ISC #1 pass through the CM because they have the highest priority levels. In time slot $(T + 1)$, ISC #3 transmits two cells (a and b) and ISC #1 transmits two cells (A and B). In time slot $(T + 2)$, ISC #3 transmits one cell (c), while ISC #1 transmits three cells (C, D, and E). In time slot $(T + 3)$, ISC #1 has the highest port priority ('0') and is able to transmit all its cells (F, G, H, and I). Once they are cleared from the one-cell buffers, ISC #1 resets its sequence priority to zero.

### 10.6.3 Resequencing Cells

As discussed before, the routing delay in the Abacus switch must be less than 424 bit times. If the routing delay is greater than 424, there are two possibilities. First, if a cell is held up in the input buffer longer than a cell slot time, the throughput of the switch fabric will be degraded. Second, if a cell next to the HOL cell is sent to the MGN before knowing if the HOL cell has been successfully transmitted to the desired output switch module(s), it may be ahead of the HOL cell that did not pass through the MGN. This cell out-of-sequence problem can be resolved with a resequencing buffer (RSQB) at the output port of the MGN.

For a switch size $N$ of 1024, output group size $M$ of 16, and group expansion ratio $L$ of 1.25, the maximum routing delay of the single stage Abacus switch is 1043 (i.e., $N + L \times M - 1$ as described previously). Therefore, an arbitration cycle is at least three cell time slots. If up to three cells are allowed to transmit during each arbitration cycle, the IPC must have three one-cell buffers arranged in parallel.

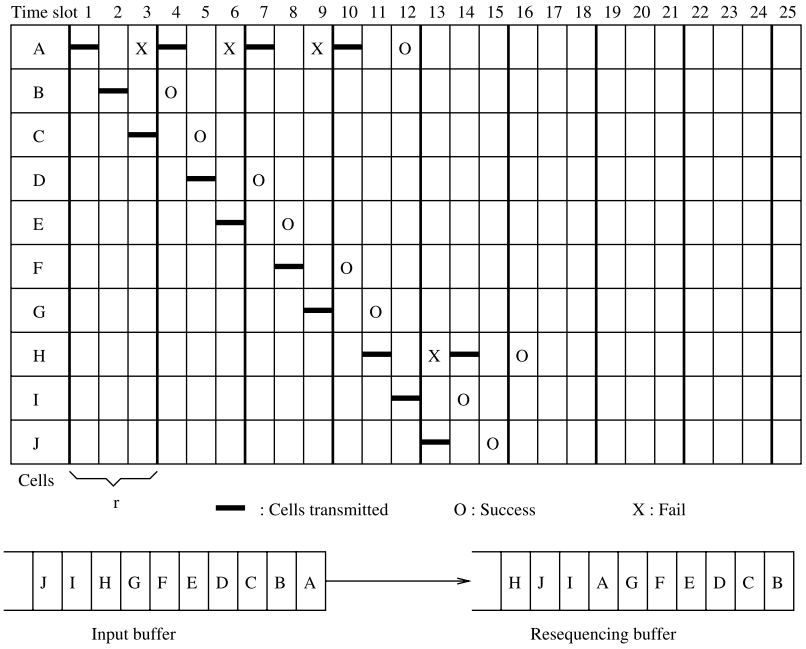



**Figure 10.22** Example of cell out-of-sequence with arbitration cycle of three cell slots.

Figure 10.22 illustrates an example of the maximum degree of out-of-sequence. Assume cells A to J are stored in the same input buffer in sequence. In time slot 1, cell A is sent to the switch fabric. It takes three time slots to know if cell A has passed through the switch fabric successfully. In time slot 2, cell B is sent to the switch fabric, and in time slot 3, cell C is sent to the switch fabric. Before the end of time slot 3, the IPC knows that cell A has failed to pass the switch fabric, so cell A will be transmitted again in time slot 4. If cell A passes through the switch fabric successfully on the fourth try, up to six cells (cells B to G) can be ahead of the cell A. The cell arriving sequence in the RSQB is shown in Figure 10.22.

For this scheme to be practical, the size of the maximum degree of cell out-of-sequence and the size of the RSQB should be bounded. Let us consider the worst case. If all HOL cells of $N$ input ports are destined for the same output group and the tagged cell has the lowest port priority in time slot $T$, $L \times M$ highest priority cells will be routed in time slot $T$. In time slot $(T + 1)$, the priority level of the tagged cell will be incremented by one so that there can be at most $(N - L \times M - 1)$ cells whose priority levels are greater than that of the tagged cell. In time slot $(T + 2)$, there can be at most $(N - 2 \times L \times M - 1)$ cells whose priority levels are greater than that of the tagged cell, and so on.

The maximum degree of out-of-sequence can be obtained from the following. An arbitration cycle is equal to or greater than $r = \lceil N + L \times M - 1/424 \rceil$ cell time slots. A tagged cell succeeds in at most $s = \lceil N/L \times M \rceil$ tries. Therefore, in the worst case, the HOL cell passes through the switch fabric successfully in $(r \times s)$ time slots. Denote the maximum degree of out-of-sequence to be $t$; $t = r \times s$.

One way to implement the resequencing is to time stamp each cell with a value equal to the real time plus the maximum degree of cell out-of-sequence ($t$), when it is sent to the switch fabric. Let us call the time stamp the 'due time of the cell'. Cells at the RSQBs are

**TABLE 10.4 Maximum Degree of Cell Out-of-Sequence ($t$)**

| $N$ | $M$ | $L$ | $r$ | $s$ | $t$ |
|---|---|---|---|---|---|
| 1024 | 16 | 1.25 | 3 | 52 | 156 |
| 8192 | 16 | 1.25 | 20 | 410 | 8200 |

moved to the SSM when their due times are equal to the real time. This maintains the cell transmission sequence.

The drawbacks of this scheme are high implementation scomplexity of the RSQB and large cell transfer delay. Since a cell must wait in the RSQB until the real time reaches the due time of the cell, every cell experiences at least $t$ cell slot delay even if there is no contention. Table 10.4 shows the maximum degree of cell out-of-sequence for switch sizes of 1024 and 8192. For switch size 1024, the maximum degree of cell out-of-sequence is 156 cell slots, which corresponds to 441 μsec for the OC-3 line rate (i.e., $156 \times 2.83$ μsec).

### 10.6.4 Complexity Comparison

This section compares the complexity of the above three approaches for building a large-capacity Abacus switch and summarizes the results in Table 10.5. Here, the switch element in the RMs and CMs is a $2 \times 2$ crosspoint device. The number of inter-stage links is the number of links between the RMs and the CMs. The number of buffers is the number of ISCs. For the BMCN, the CMs have ISCs and one-cell buffers (Fig. 10.20*c*). The second and third parts of Table 10.5 give some numerical values of a 160 Gbit/s Abacus switch.

**TABLE 10.5 Complexity Comparison of Three Approaches for a 160 Gbit/s Abacus Switch**

| | MMCN | BMCN | Resequencing |
|---|---|---|---|
| Number of switch elements | $LN^2 + L^2Nn$ | $N^2 + Nn$ | $LN^2$ |
| Number of inter-stage links | $JLN$ | $JN$ | 0 |
| Number of internal buffers | 0 | $JK$ | 0 |
| Number of MP bits | $K$ | $K$ | $K$ |
| Out-of-sequence delay | 0 | 0 | $\lceil (N + LM - 1)/424 \rceil \times \lceil N/(LM) \rceil$ |
| Routing delay in bits | $n + Ln - 1$ | $n + LM - 1$ | $N + LM - 1$ |
| Switch size $N$ | 1024 | 1024 | 1024 |
| Group size $M$ | 16 | 16 | 16 |
| Output exp. ratio $L$ | 1.25 | 1.25 | 1.25 |
| Module size $n$ | 128 | 128 | 128 |
| Number of switch elements | 1,515,520 | 1,179,648 | 1,310,720 |
| Number of inter-stage links | 10,240 | 8192 | 0 |
| Number of internal buffers | 0 | 512 | 0 |
| Number of MP bits | 64 | 64 | 64 |
| Out-of-sequence delay | 0 | 0 | 156 |
| Routing delay in bits | 287 | 147 | 1043 |

Here, it is assumed that the switch size $N$ is 1024, the input line speed is 155.52 Mbit/s, the group size $M$ is 16, the group expansion ratio $L$ is 1.25, and the module size $n$ is 128.

With respect to the MMCN, there are no internal buffers and no out-of-sequence cells. Its routing delay is less than 424 bit times. But the number of SWEs and inter-stage links is the highest among the three approaches.

For the BMCN, the routing delay is no longer a concern for a large-capacity Abacus switch. Its number of SWEs and inter-stage links is smaller than that of the MMCN. However, it may not be cost-effective when it is required to implement buffer management and cell scheduling in the intermediate-stage buffers to meet QoS requirements.

The last approach of resequencing out-of-sequence cells has no inter-stage links nor internal buffers. It requires a resequencing buffer at each output port. For a switch capacity of 160 Gbit/s, the delay caused by resequencing cells is at least 156 cell slot times.

## 10.7 ABACUS SWITCH FOR PACKET SWITCHING

The Abacus switch can also handle variable-length packets. To reserve cell[1] sequence in a packet, two cell scheduling schemes are used at the input buffers. The packet interleaving scheme transfers all cells in a packet consecutively, while the cell interleaving scheme transfers cells from different inputs and reassembles them at the output.

### 10.7.1 Packet Interleaving

A packet switch using the packet interleaving technique is shown in Figure 10.23. The switch consists of a memoryless nonblocking MGN, IPCs, and OPCs. Arriving cells are stored in an input buffer until the last cell of the packet arrives. When the last cell arrives, the packet is then eligible for transmission to output port(s).

In the packet interleaving scheme, all cells belonging to the same packet are transferred consecutively. That is, if the first cell in the packet wins the output port contention for a destination among the contending input ports, all the following cells of the packet will be transferred consecutively to the destination.

A packet interleaving switch can be easily implemented by the Abacus switch. Only the first cells of head-of-line (HOL) packets can contend for output ports. The contention among the first cells of HOL packets can be resolved by properly assigning priority fields to them. The priority field of a cell has $(1 + \log_2 N)$ bits. Among them, the $\log_2 N$-bit field is used to achieve fair contention by dynamically changing its value as in the Abacus switch. Prior to the contention resolution, the most-significant-bit (MSB) of the priority field is set to 1 (low priority). As soon as the first cell of an HOL packet wins the contention (known from the feedback priorities), the MSB of the priority field of all the following cells in the same packet is asserted to 0 (high priority). As soon as the last cell of the packet is successfully sent to the output, the MSB is set to 1 for the next packet. As a result, it is ensured that cells belonging to the same packet are transferred consecutively.

Figure 10.24 shows the average packet delay versus offered load for a packet switch with packet interleaving. In the simulations, it is assumed that the traffic source is an ON–OFF model. The packet size is assumed to have a truncated geometric distribution with an

[1]The cell discussed in this section is just a fixed-length segment of a packet and does not have to be 53 bytes like an ATM cell.

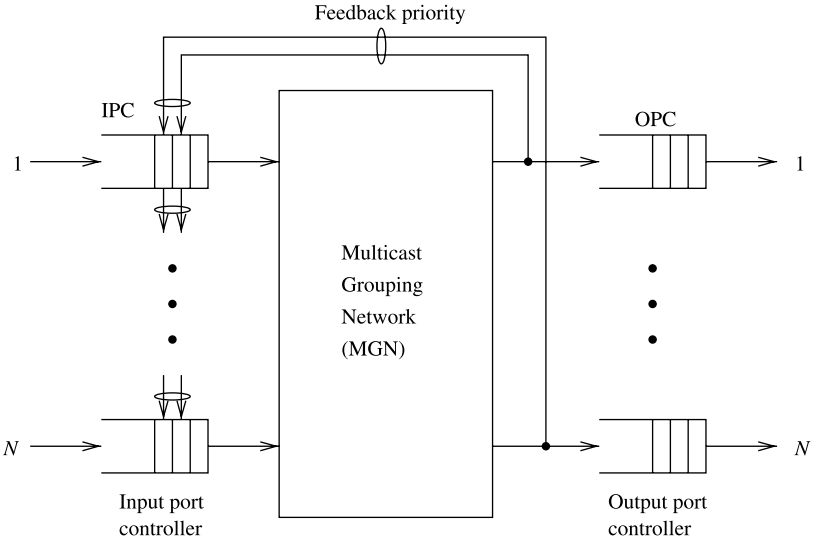



**Figure 10.23** Packet switch with packet interleaving.

average packet size of 10 cells and the maximum packet size of 32 cells (to accommodate the maximum Ethernet frame size). The packet delay through the switch is defined as follows. When the last cell of a packet arrives at an input buffer, the packet is time-stamped with an arrival time. When the last cell of a packet leaves the output buffer, the packet is time-stamped

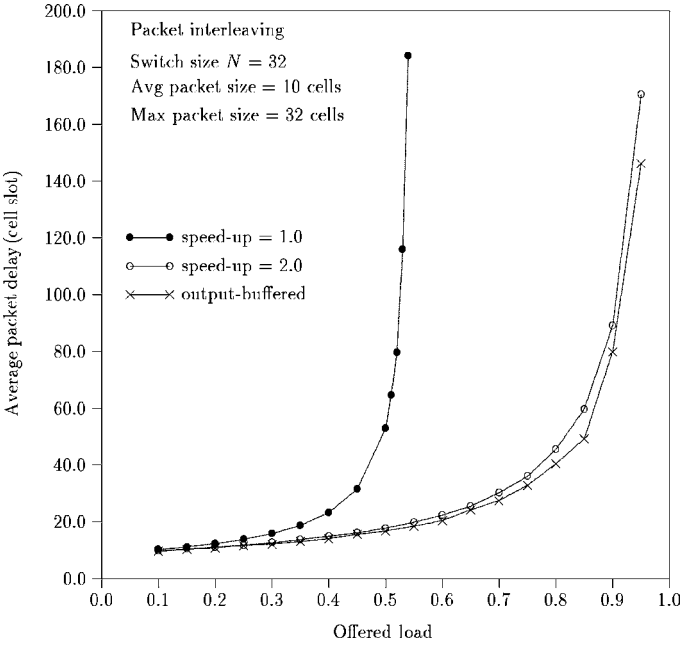



**Figure 10.24** Delay performance of a packet switch with packet interleaving.

with a departure time. The difference between the arrival time and the departure time is defined as the packet delay. When there is no internal speedup ($S = 1$) in the switch fabric, the delay performance of the packet interleaving switch is very poor, mainly due to the HOL blocking. The delay/throughput performance is improved by increasing the speedup factor $S$ (e.g., $S = 2$). Note that the input buffer's average delay is much smaller than the output buffer's average delay. With an internal speedup of two, its output buffer's average delay dominates the total average delay and is very close to that of the output-buffered switch.

### 10.7.2 Cell Interleaving

A packet switch using a cell interleaving technique is shown in Figure 10.25. Arriving cells are stored in the input buffer until the last cell of a packet arrives. Once the last cell of a packet arrives, cells are transferred in the same way as in an ATM switch. That is, cells from different input ports can be interleaved with each other as they arrive at the output port. Cells have to carry input port numbers so that output ports can distinguish them from different packets. Therefore, each output port has $N$ reassembly buffers, each corresponding to each input port. When the last cell of a packet arrives at the reassembly buffer, all cells belonging to the packet are moved to the output buffer for transmission to the output link. In real implementation, only pointers are moved, not the cells. This architecture is similar to the one in [16] in the sense that they both use reassembly buffers at the outputs, but it is more scalable than the one in [16].

The operation speed of the Abacus switch fabric is limited to several hundred Mbit/s with state-of-the-art CMOS technology. To accommodate the line rate of a few Gbit/s (e.g., Gigabit Ethernet and OC-48), we can either use the bit-slice technique or the one shown in Figure 10.25, where the high-speed cell stream is distributed up to $m$ one-cell buffers at each input port. The way of dispatching cells from the IPC to the $m$ one-cell buffers is identical

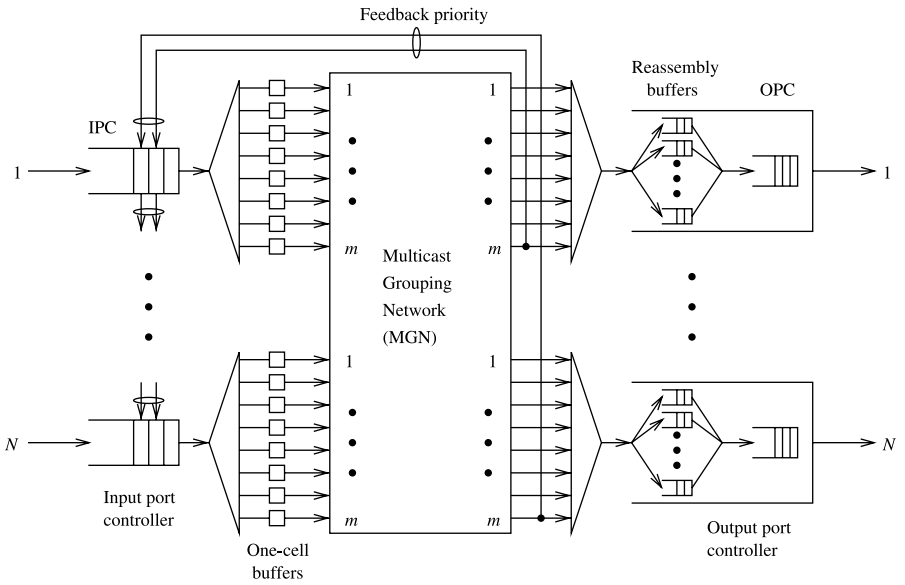



**Figure 10.25** Packet switch with cell interleaving.

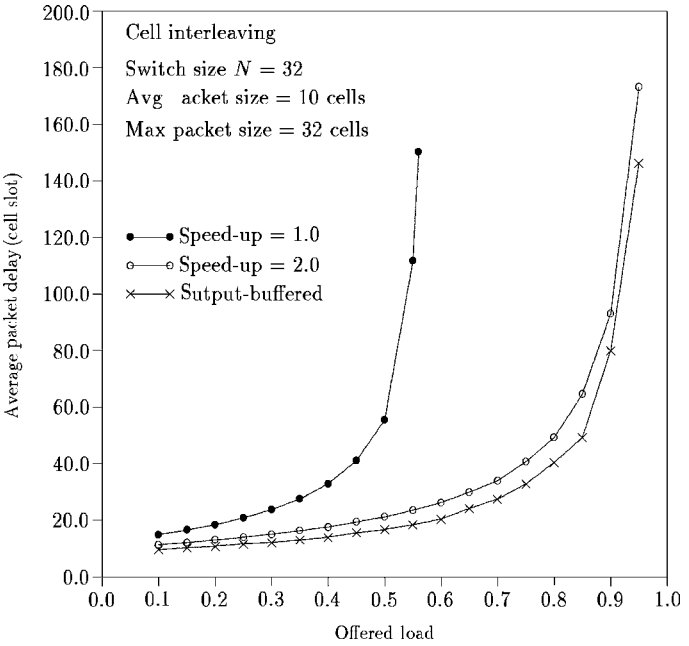



**Figure 10.26** Delay performance of a packet switch with cell interleaving.

to dispatching cells from the ISC to the $M$ one-cell buffers in Figure 10.20*c*. The advantage of using the technique in Figure 10.25 over the bit-slice technique is its smaller overhead bandwidth, since the latter shrinks the cell duration while keeping the same overhead for each cell.

Figure 10.26 shows the average packet delay versus offered load for a packet switch with cell interleaving. When there is no internal speedup ($S = 1$), the delay performance is poor. With an internal speedup $S$ of 2, the delay performance is close to that of an output-buffered packet switch. By comparing the delay performance between Figure 10.24 and Figure 10.26, we can see that the average delay performance is comparable. However, we believe that the delay variation of cell interleaving will be smaller than that of packet interleaving because of its finer granularity in switching.

## REFERENCES

[1] H. J. Chao, B. S. Choe, J. S. Park, and N. Uzun, "Design and implementation of Abacus switch: A scalable multicast ATM switch," *IEEE Journal on Selected Areas in Communications*, vol. 15, no. 5, pp. 830–843 (June 1997).

[2] J. Hui and E. Arthurs, "A broadband packet switch for integrated transport," *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 8, pp. 1264–1273 (Oct. 1987).

[3] B. Bingham and H. Bussey, "Reservation-based contention resolution mechanism for Batcher-banyan packet switches," *Electronics Letters*, vol. 24, no. 13, pp. 772–773 (June 1988).

[4] A. Cisneros and C. A. Bracket, "A large ATM switch based on memory switches and optical star couplers," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1348–1360 (Oct. 1991).

[5] R. Handel, M. N. Huber, and S. Schroder, *ATM networks: Concepts, Protocols, Applications*. Addison-Wesley Publishing Company, Reading, MA, 1994, Chapter 7.

[6] M. J. Karol, M. G. Hluchyj, and S. P. Morgan, "Input versus output queuing on a space-division packet switch," *IEEE Transactions on Communications*, vol. 35, pp. 1347–1356 (Dec. 1987).

[7] Y. Oie, M. Murata, K. Kubota, and H. Miyahara, "Effect of speedup in nonblocking packet switch," in *Proc. ICC'89*, Boston, Massachusetts, pp. 410–415 (June 1989).

[8] A. Pattavina, "Multichannel bandwidth allocation in a broadband packet switch," *IEEE Journal on Selected Areas in Communications*, vol. 6, no. 9, pp. 1489–1499 (Dec. 1988).

[9] S. C. Liew and K. W. Lu, "Comparison of buffering strategies for asymmetric packet switch modules," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 3, pp. 428–438 (Apr. 1991).

[10] T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano, "$32 \times 32$ shared buffer type ATM switch VLSI's for B-ISDN's," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1239–1247 (Oct. 1991).

[11] C. Y. Chang, A. J. Paulraj, and T. Kailath, "A broadband packet switch architecture with input and output queuing," in *Proc. IEEE GLOBECOM'94*, San Francisco, California, pp. 448–452 (Nov. 1994).

[12] A. Pattavina and G. Bruzzi, "Analysis of input and output queuing for nonblocking ATM switches," *IEEE/ACM Transactions on Networking*, vol. 1, no. 3, pp. 314–328 (June 1993).

[13] H. J. Chao and N. Uzun, "An ATM routing and concentration chip for a scalable multicast ATM switch," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 816–828 (June 1997).

[14] E. P. Rathgeb, W. Fischer, C. Hinterberger, E. Wallmeier, and R. Wille-Fier, "The MainStreet*xpress* core services node – a versatile ATM switch architecture for the full service network," *IEEE J. on Selected Areas in Communications*, vol. 15, no. 5, pp. 830–843 (June 1997).

[15] K. Y. Eng and M. J. Karol, "State-of-the-art in gigabit ATM switching," in *Proc. IEEE BSS'95*, Poznan, Poland, pp. 3–20 (Apr. 1995).

[16] I. Widjaja and A. I. Elwalid, "Performance issues in VC-merge capable switches for IP over ATM networks," in *Proc. INFOCOM'98*, San Francisco, California, pp. 372–380 (Mar. 1998).