A High-Performance Scalable ATM Switch Design by Integrating Time-Division and Space-Division Switch Architectures

Young-Keun Park

Abstract

Advances in VLSI technology have brought us completely new design principles for the high-performance switching fabrics including ATM switches. From a practical point of view, port scalability of ATM switches emerges as an important issue while complexity and performance of the switches have been major issues in the switch design. In this paper, we propose a cost-effective approach to modular ATM switch design which provides the good scalability. Taking advantages of both time-division and space-division switch architectures, we propose a practically implementable large scale ATM switch architecture. We present a scalable shared buffer type switch for a building block and its expansion method. In our design, a large scale ATM switch is realized by interconnecting the proposed shared buffer switches in three stages. We also present an efficient control mechanism of the shared buffers, synchronization method for the switches in each stage, and a flow control between stages. It is believed that the proposed approach will have a significant impact on both improving the ATM switch performance and enhancing the scalability of the switch with a new cost-effective scheme for handling the traffic congestion. We show that the proposed ATM switch provides an excellent performance and that its cell delay characteristic is comparable to output queueing which provides the best performance in cell delay among known approaches.

I. Introduction

Within the field of Broadband Integrated Services Digital Networks(B-ISDN's), Asynchronous Transfer Mode(ATM) technique has recently become an extremely popular topic around the world. The ATM systems are expected to provide a wide range of services such as data, voice, and video communications, and to satisfy the requirements of various media under various traffic conditions. The ATM is essentially a packet-oriented transfer mode, in which transmitted data are segmented into a number of short fixed-size packets called *cells*. Each cell consists of 53 octets comprising a 5-octet header and a 48-octet payload carrying the user information[1]. The main function of an ATM switch is to route ATM cells from input ports to designated output ports by using the address information carried in the header of each ATM cell.

An ideal switch is one that can route all packets from their inputs to their outputs without loss, with a short delay, and with a good output line efficiency (i.e., switch throughput), while preserving the order of the transmitted packets. Various types of ATM switch architectures have been proposed, and a survey of several proposed high-performance packet switching fabrics can be found in[2] and[3]. Most of the proposals on the internal architecture of the ATM switch are still at a research-and-evaluation stage, and they are based on the switching architectures of the 1970s and 1980s, which have been proposed for telecommunication switching system or as the interconnect fabric of multiprocessor systems. Only a few ATM switches that have entered the commercial market embody attributes to meet current demands. Even though complexity and performance of a switch have been major issues in ATM switch design, the port scalability, efficient use of switch bandwidth, and reliability factors are becoming increasingly important in the commercial market[4].

In this paper, we propose a cost-effective approach to ATM switch design which provides the good scalability as well as the good performance. Scalability refers to a system's capability to evolve or expand in response to increased demands. This requirement is becoming crucial as the network size grows rapidly. Port scalability is also closely related to the complexity measure of switch architectures. In a matrix or crossbar switch, the addition of k new ports requires k^2 junctions. The addition of k new ports to a Banyan network requires $k\log_2 k$ switching elements. For a shared medium or a shared buffer memory architecture[5, 6], this scalability characteristic is linear in k. Since each switch architecture

Manuscript received July 15, 1997; accepted December 8, 1997.

Y. K. Park is with the Dept. of Electrical Engineering, Yonsei University, Seoul, Korea (e-mail : ypark@bubble.yonsei.ac.kr).

has its own advantages, integrating a couple of different architectures in a cascaded or hierarchical fashion may be the logical way to address the scalability, the efficient use of bandwidth, and the reliability concerns of the market[4]. In our approach to scalable ATM switch design, we integrate two different architectures: *time-division* and *space-division*. We consider a three-stage Clos interconnection (space-division) of shared buffer type switches (time-division) without employing the rearrangeability of Clos network[8], and then we propose a shared buffer switch which can be used as a building block of a three-stage ATM switch. In order to maximize the switch throughput and to minimize congestion, we propose a synchronized control for all shared buffer switches.

The major advantages of the proposed approach are three folds. First, our switch is strictly nonblocking and we can obtain 100% of potential switch throughput since we employ shared buffer architecture which does not suffer from blocking. Therefore, no extra hardware for cell scheduling or congestion control is required. Second, our switch guarantees in-sequence transmissions of ATM cells since we do not employ the rearrangeability of Clos network in which multiple paths exit between a pair of source and destination. Therefore, we don't have to worry about the out-of-sequence cell transmission which is always a concern when a multiple path switching network is employed. Third, the scalability characteristic of the proposed switch is linear in the number of ports. This is because the shared buffer switch has a linear scalability and the number of stages in the proposed ATM switch is fixed.

In Section Ⅱ, general description of the shared buffer switch is provided. Section Ⅲ is the main body of this paper. In this section, we describe how we interconnect the shared buffer switches and the details of the proposed shared buffer switch along with the control algorithm and hardware concerns. In Section Ⅳ, we evaluate the performance of our proposed switch under uniform random and bursty traffic conditions. Finally, concluding remarks are given in Section V.

Ⅱ. Shared Buffer Switch : Its Pros and Cons

An ATM switch consists of switching elements and buffers in which ATM cells are stored for queueing. Depending upon the buffer position, the ATM switch architecture can be classified as input buffer switch, output buffer switch, buffered Banyan switch, and shared buffer switch[4]. An input buffer switch, in spite of its simple queueing structure, provides a limited throughput due to *head-of-line* (HOL) blocking. An output buffer switch can provide better performance in terms of delay or throughput than an input queueing approach. With output queueing, however, the switch fabric of size N and its output buffers must operate N times as fast as the input/output line speed. Output queueing at the reduced speed can be achieved by providing an N to L ($< N$) concentrator at each output and a fully interconnected switch fabric[9],

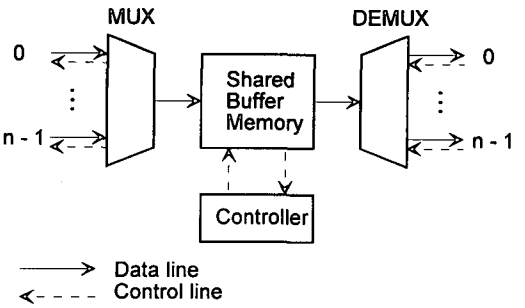


Fig. 1. An $n \times n$ shared buffer switch.

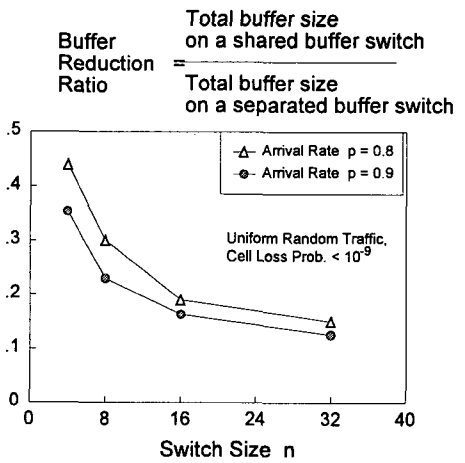


Fig. 2. Buffer reduction ratio under uniform random traffic : adapted from[10].

increasing the hardware complexity. Buffered Banyan switch is seldom used for ATM because it requires a quite large space for internal buffers. A typical ATM switch would need to be extended to hundreds or more ports. In terms of both port scalability and switch performance, the shared buffer type switch is shown to be the best choice[4]. An $n \times n$ shared buffer switch usually consists of an $n \times 1$ multiplexer, an $1 \times n$ demultiplexer, a shared buffer memory, and a controller as shown in Fig. 1. ATM cells arriving at the input ports are time-multiplexed on a round-robin basis and stored in the shared buffer one by one. The buffer controller selects and transmits the cells designated to each output port through the demultiplexer. A major difficulty in realizing this type of switch lies in implementing the high speed buffer, since the buffer access speed has to be faster than the input/output speed. To overcome this problem, some shared buffer switch architectures employ a bit-parallel scheme which makes the implementation less difficult with the current technology[7].

With a shared buffer switch, the buffer size required to meet a given cell loss probability is less than that with a switch having non-shared buffers. In terms of buffer sharing, its effectiveness depends on the number of input / output ports that share a buffer memory. The effectiveness of buffer sharing can be represented as

a *buffer reduction ratio*, given a required cell loss probability and the input load. Fig. 2 shows the buffer reduction ratio provided in [10]. From this figure, we see that, considering cost-effectiveness, a desirable switch size would be less than 16. In a shared buffer switch, the required buffer size depends on the cell loss probability requirement, the traffic load, and the traffic pattern. In the following section, based on the basics of the shared buffer switch described in this section, we propose our shared buffer switch design and its control schemes designed for three-stage interconnections

III. Proposed ATM Switch Architecture

One of the major area under study of ATM switching systems is switch architecture. Each architecture has its own advantages and disadvantages. A time-division switch which employs a shared medium architecture or a shared memory architecture exhibits linear scalability relative to port expansion. However, it has a limit in the number of ports due to the limit of bus speed or memory access speed. A space-division switch fabric based on simple switching elements is suitable for VLSI implementation. However, an additional hardware is required for controlling internal and / or output port blocking, and this type of switch does not scale linearly in switch complexity. Hence, we consider a hybrid architecture, integrating time and space division switch architectures. In this section, we first give an idea of how we interconnect the shared buffer switches to realize a large ATM switch, and then we present the details of our shared buffer switch architecture with a synchronized control mechanism.

1. Three-Stage Clos Interconnection with Time-Division Multiplexing

One way to build a large ATM switch is to interconnect many identical switches in a cascaded form. In this paper, we employ three-stage Clos interconnection scheme. Clos network[8] is a three-stage rearrangeable switching network. Since it provides multiple paths between any source / destination pair, it is non-blocking by way of rearrangement (i.e., internal link blocking can be resolved by rearranging existing connections) while unique path multistage interconnection networks(MIN's) suffer from internal link blocking which results in low throughput. Clos network, however, requires a central controller for routing while MIN's have a self-routing capability.

In our approach, to interconnect our shared buffer switches we employ the interconnection topology of Clos network as shown in Fig. 3. Here, we emphasize that our approach provides neither multiple paths nor rearrangeability. Hence, we do not need a central routing controller to determine the connection paths. Instead, the individual shared buffer switch (we call this switch as *unit-switch* in this paper) determines the path. Since the unit-switch employs a time-division multiplexing scheme, we can

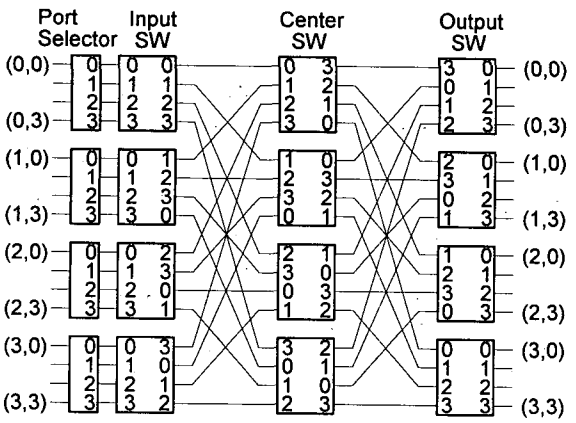


Fig. 3. Three-stage Clos interconnection of proposed shared buffer switches(N = 16, n = 4).

interconnect the whole switching system in a *time slot*(i.e., a time duration for a 53byte ATM cell to be transmitted) by operating all unit-switches in a synchronized and time-multiplexed manner.

For an $N\times N$ ATM switch, we assume that each stage consists of n identical shared buffer switches of size n, where $n=\sqrt{N}$, and that during a time slot each switch accepts up to n ATM cells one by one from n inputs. The choice of $n=\sqrt{N}$ came from the fact that it provides the minimum hardware complexity for an $N\times N$ Clos network and this choice enables us to design our switch without any difficulty in synchronization. Let T_S denote a time slot, and we divide T_S into n cycles, making a timing sequence $t_0, t_1, \cdots, t_{n-1}$. The timing sequence of input or output ports of a switch can be set in many ways, so that there should be no timing conflict among input (or output) ports of every switch. Fig. 3 shows an example of the proposed ATM switch when N = 16 (n = 4). In this figure, we let the numbers inside each unit-switch indicate both the timing sequences and the port labels. The numbers in parentheses represent the upper half and the lower half port addresses. That is, (i, j) represents j-th input (or output) port of i-th input (or output) switch. Consider a following example. At time t_0, a cell arriving at source input port (0, 0) is sent to input switch 0 through its port 0. At the same time, a cell having been stored in input switch 0 is chosen and transmitted to center switch 0 through its port 0. During this time, a cell having been stored in center switch 0 is chosen and transmitted to output switch 3 through its port 0. Simultaneously, a cell in output switch 3 is selected and transmitted through its port 0.

For every unit-switch at time t_i, a cell arriving at its port i is written into the buffer of the switch and a cell, if any, leaves from the port i. Note that in the center switches the sequences of output ports are in the reverse order of those of input ports. This is to eliminate the possible unfairness among the input / output ports which may otherwise results from the use of cyclic sequence. Fig. 4 shows snap shots of connection patterns between stages at each

cycle during a time slot, and these are taken based on the example given in Fig. 3.

For routing in a unit-switch, we provide a routing decision rule as follows.

(1) For input switches : At time t_i, every switch selects an ATM cell destined to output switch i and it transmits the cell to a connected center switch.
(2) For center switches : At time t_i, every switch selects an ATM cell destined to the output switch which is connected to port i of the center switch and it transmits the cell to the output switch.
(3) For output switches : At time t_i, every switch selects an ATM cell destined to its output port i and it transmits the cell through its output link i.

All unit-switches in three stages are operated in this synchronized manner. Since the connection pattern between stages is uniquely defined at every cycle t_i and since the path for an ATM cell is uniquely determined by the cell's destination, there is no chance to have *out-of-sequence* errors in cell transfer between any pair of source and destination of the proposed ATM switch. This routing decision rule not only guarantees the sequence integrity of ATM cells, but it also makes the traffic evenly distributed to the center switches, making an efficient utilization of all switches.

In addition to the routing mechanism, we provide a flow control between stages. To prevent center switches and output switches from buffer overflow, *buffer-full* control signals are fed back to the previous stages. If the buffer in a receiving switch is full, the sending switch does not send a cell at the current cycle. Therefore, a cell loss due to a buffer overflow occurs only at the input switches.

One may argue that the overall synchronization for all unit-switches will not be an easy task in actual implementation, however, this can be done easily. Since each shared buffer switch has internal counters for controlling its multiplexer and demultiplexer, we can externally reset or load the counters with predefined initial values before operating the ATM switch. The details of this will be given in the following subsection.

2. *Proposed Shared Buffer Switch*

In designing our shared buffer unit-switch, we focused on the efficient way of controlling multiplexers, demultiplexers, and buffers with low hardware overhead. The proposed shared buffer switch is sketched in Fig. 5. Notice that, in our switch, an OR gate is used instead of the input multiplexer, since only one input port is active at a time. In front of the input switches, we place port selectors to multiplex incoming cells in time (see Fig. 3).

Incoming cells are converted from serial to parallel format in the S / P converter, and outgoing cells are converted from parallel to serial format in the P/S converter. For shared buffers, we

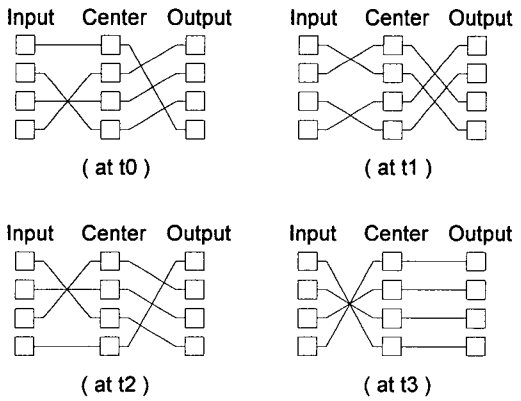


Fig. 4. Snap shots of connection pattern at every cycle(N = 16, n = 4).

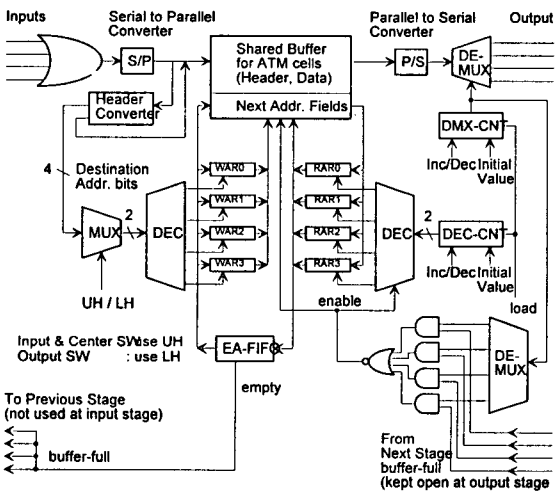


Fig. 5. Proposed shared buffer switch with flow control(n = 4).

employed a buffer memory architecture similar to those used in [5, 10]. The buffer memory is structured as linked lists of memory as shown in Fig. 6. For an $n \times n$ shared buffer switch, n linked lists are maintained by n read address registers (RAR's) and n write address registers (WAR's). Every list consists of an ATM cell and a pointer to next ATM cell. RAR indicates the head of lists and WAR indicates the end of list which is an empty memory location. Therefore, each pair of RAR and WAR maintains an output queue. In our approach, however, an output queue represented by RAR_i and WAR_i is not necessarily dedicated to output port i ($0 \le i \le n$-1) of the $n \times n$ shared buffer switch, rather it is related to destination port j($0 \le j \le N$-1) of the $N \times N$ ATM switch. In order to obey our previously discussed routing decision rule, the unit-switch decodes the incoming cell's destination address and writes the cell into the buffer in a predefined manner. For the switches at input and center stages, the decoder for WAR's is set to select a WAR based on the upper half address bits of cell's destination. So, a pair of RAR_i and WAR_i holds the cells that are destined to output switch i. For the output switches, the decoder is set to select a WAR based on

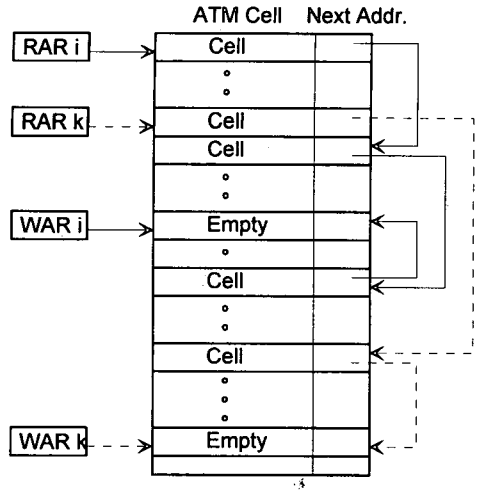


Fig. 6. Shared buffer memory structure.

DEC-CNT

SW#	Input	Center	Output
0	0	3	0
1	0	0	0
2	0	1	0
3	0	2	0

DMX-CNT

SW#	Input	Center	Output
0	0	3	0
1	3	0	0
2	2	1	0
3	1	2	0

(n - 1 + SW#) modulo n

(n - SW#) modulo n

Counters at input/output stages : Incremented at every cycle
Counters at center stage : Decremented at every cycle

Fig. 7. Initial values of internal counters(N = 16, n = 4).

the lower half destination address bits of incoming cells. In this case, a pair of RAR_i and WAR_i holds the cells that are destined to its output port i.

The output demultiplexer and the decoder for RAR's are controlled by counters DMX-CNT and DEC-CNT, respectively. The decoding sequence for RAR's is different between stages, and it is determined by the initial value of the DEC-CNT. In the input and the output switches, these two counters are incremented at every cycle, while those in the center switches are decremented at every cycle. The initial values for these two counters depends on the stage and the switch index i at each stage. They are shown in Fig. 7. For input switch i, the initial value of DEC-CNT is zero and the initial value of DMX-CNT is $(n - i)$ modulo n, where n is the size of the shared buffer switch. For center switch i, DEC-CNT and DMX-CNT have the identical initial value of $(n - 1 + i)$ modulo n. For output switches, the initial values of both counters are zeros. In our $N \times N$ ATM switch, the interplay of decoders and demultiplexers controlled by these two counters exactly follows the routing decision algorithm provided in the previous subsection. The initialization of internal counters can be easily done by a *load* signal, and the initial values can be loaded externally.

At every cycle, an incoming cell is decoded and written into the memory location pointed by a WAR which is selected based on the cell's destination address. At the same time, an empty memory address is read from empty address FIFO queue (EA-FIFO) and written into the next address field of the buffer memory along with the cell, and it is also overwritten into the WAR. A cell is read from the memory location pointed by a RAR which is selected based on the current cycle, and the cell is transmitted to an output link which is selected based on the current cycle through the demultiplexer and the P/S converter. Simultaneously, the content of the next address field of the transmitted cell replaces the content of the RAR. The memory location where the transmitted cell has been stored becomes empty and its address which has been stored in the RAR is written into the EA-FIFO. During a time slot, this whole cycle performs n times, allowing routing of up to n ATM cells. Since the counters inside all unit-switches are synchronized by a main clock signal, we can easily interconnect the unit-switches as discussed in the previous subsection.

Now, we consider the flow control between stages. Besides the input / output links for data transfer between switches, we provide the switch with a buffer-full control signal. When the empty address FIFO queue (EA-FIFO) is empty, which means the shared buffer is full, the buffer-full control line becomes active high, otherwise it is kept low. The input switches or the center switches transmit their cells to the switches in the next stage only when the buffer-full signal from the receiving side is low. This is controlled by the *enable* signal shown in Fig. 5. Therefore, a cell loss only occurs at the input stage when the buffer of an input switch is full.

3. *Hardware Consideration*

The proposed shared buffer switch can be implemented in one or multiple LSI chip(s). The hardware complexity of a unit-switch is linear in the number of ports, since the most of the space is occupied by the shared buffer memory and the required size of buffer memory grows with the number of ports. The buffer size per port is the main factor that determines the cell loss rate of the switch.

At 155 Mbps port speed, a time slot T_S is about 2.74 μsec (time required to send a 53 byte cell at this speed). Since the buffer is accessed twice (i.e., read and write) at every cycle, the required buffer memory access speed can be determined by

$$T_{ACC} = \frac{T_S}{2 \times n} \cdot \frac{b}{424\ bits}$$

where n is the number of ports in a unit-switch and b is the number of parallel bits employed in the S / P or P / S converters and the buffer memory structure.

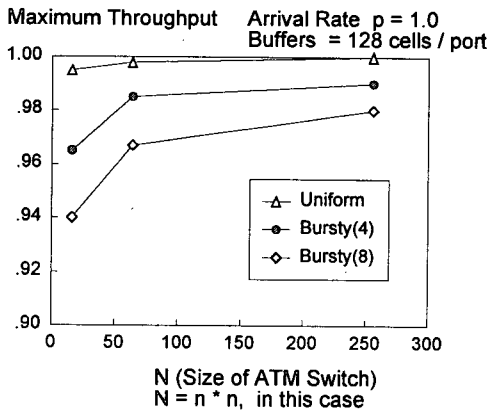


Fig. 8. Maximum throughput of proposed ATM switch.

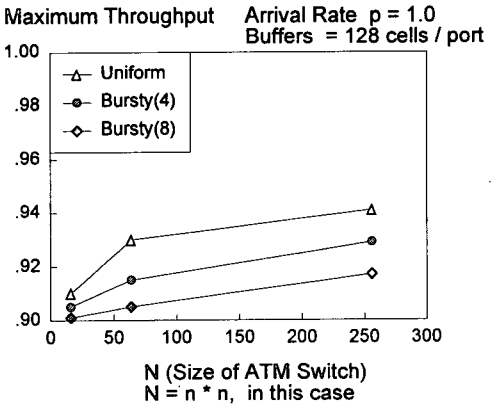


Fig. 9. Maximum throughput of Clos network with time multiplexing.

Ⅳ. Performance Evaluations by Software Simulations

In this section, we show the performance of the proposed ATM switch consisting of shared buffer switches in three stages. We also compare the performance of proposed approach with other queueing approaches (Clos network with time-multiplexed input queueing, and nonblocking switch with output queueing). The performance measure includes switch throughput, cell loss rate, and mean cell waiting time or delay. The results are obtained by simulations with the following traffic models: uniform random traffic, bursty traffic with average burst length of 4 or 8 cells. We assume that the cell arrival rate p is identical at every input port, and we tested for N = 16, 64, and 256(i.e., n = 4, 8, and 16). Each simulation run comprised 10^7 time slots at a chosen arrival rate p. We found that, in less than 200 time slots, the buffers reached a stable state. Therefore, in order to measure the steady-state performance, initial 1000 time slots were discarded. In our simulations, there was no out-of-sequence error.

Fig. 8 shows the maximum throughput of the proposed ATM switch of various sizes under various traffic conditions. Under the uniform random traffic, the proposed switch achieved 100% of throughput. Under the bursty traffic, the degradation in switch throughput depends on the burst length, and it is less sensitive with a larger switch. For comparisons, we provide Fig. 9 that shows the maximum throughput of Clos network with input queueing. We let the crossbar switches in Clos network operate in a time-multiplexed manner at the same speed as our shared buffer unit switches. We see that the proposed ATM switch outperforms Clos network even if the throughput of Clos network is improved by time-multiplexing (Maximum throughput of nonblocking switches without time multiplexing is 0.58 under uniform traffic[11]). Fig. 10 shows the average queue length in each of three stages as a function of arrival rate under the uniform traffic. The average queue lengths of input switches and output switches are almost the same, while center switches keep their buffers almost empty.

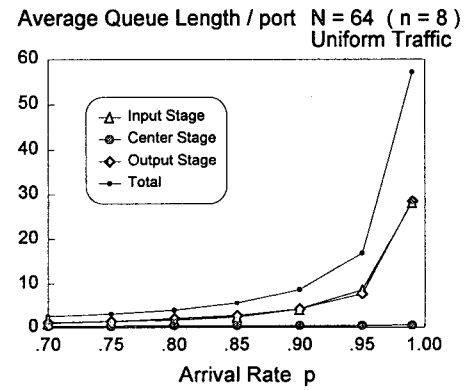


Fig. 10. Average queue length per port as a functions of arrival rate(N = 16, n = 8).

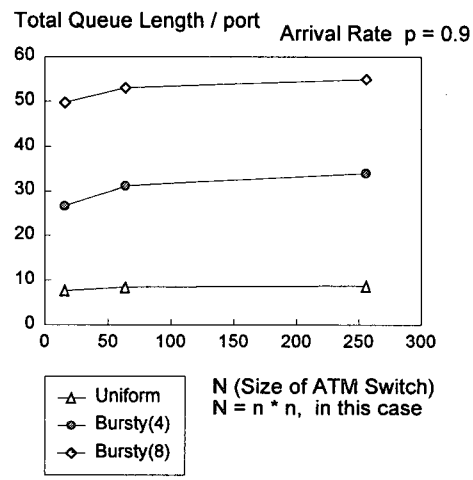


Fig. 11. Average end-to-end queue length(p = 0.9).

This is because the input switches selectively send their cells to the center switches, and the most of the center switches always have the cells to send to the destination output switches (see the routing decision algorithm in Section III). Fig. 11 shows the total queue length (i.e., sum of average queue lengths of each stage)

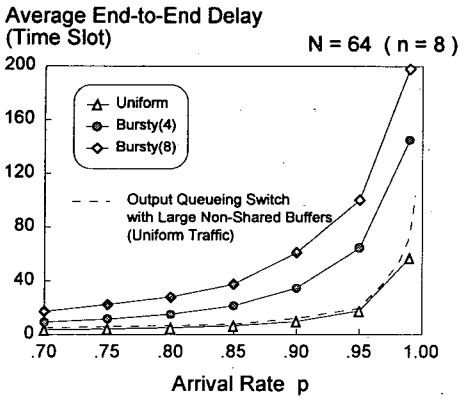


Fig. 12. Mean cell waiting time (average end-to-end delay).

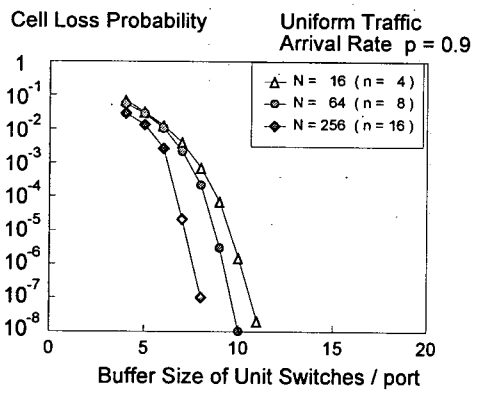


Fig. 13. Cell loss probability under uniform random traffic.

under various traffic patterns. We see that the queue length also depends on the burstiness of the traffic.

Fig. 12 depicts the average end-to-end delay (i.e., mean cell waiting time) of a 64×64 ATM switch, and it also gives a comparison with an output queueing switch[11]. Although input buffer switch with bypass queueing and intelligent congestion control[12, 13] can outperform an output queueing switch, it is generally known that the best performance in cell delay can be obtained with output queueing among various queueing approaches. The proposed ATM switch achieves a good performance in cell delay comparable to that of output queueing, even though our switch has three stages of queueing.

We obtained a cell loss probability for a buffer size k by measuring the probability that k cells are occupied in a buffer of an input switch, given a sufficiently large buffer (say 128 cells / port). With this method, we could significantly reduce the simulation time. We verified that the results obtained by this method exactly match the cell loss probabilities obtained directly for chosen buffer sizes. From Fig. 13, we can see that a buffer size of less than 10 cells/port is sufficient to provide a cell loss probability less than 10^{-6} at an arrival rate of 0.9 under the uniform random traffic. We also see that the larger switch provides the lower cell loss rate with the same buffer size per

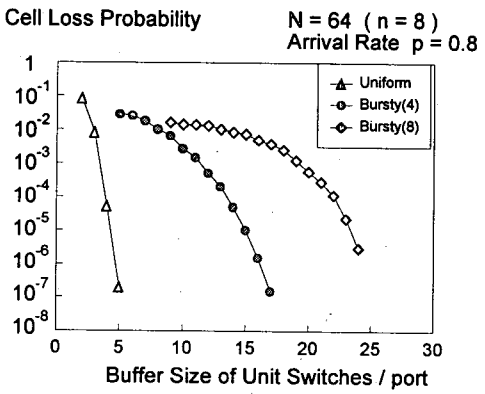


Fig. 14. Cell loss probability under bursty traffic(N = 64, n = 8).

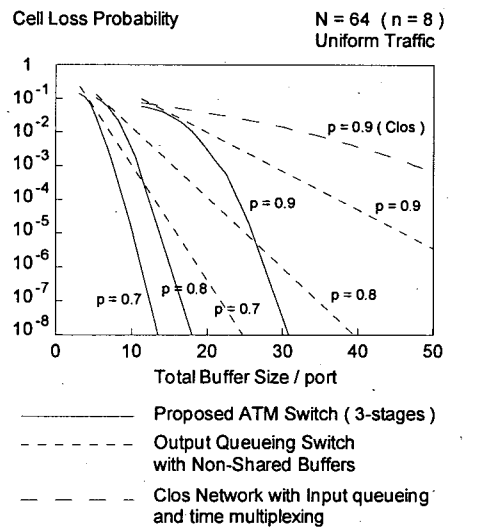


Fig. 15. Cell loss probability as a function of arrival rate (N = 64, n = 8).

port. This implies that the larger switch has a greater buffer sharing effect, resulting in lower cell loss rate. Fig. 14 shows the cell loss probability for a 64×64 switch under he various traffic conditions. It is shown that the cell loss characteristic also depends on the burstiness of the traffic. Comparisons in terms of cell loss rate between the proposed ATM switch and other queueing approaches are shown in Fig. 15. The dotted lines represent the cell loss probabilities with output queueing. The switch operating speed with output queueing has to be eight times faster than ours, that is, the output buffer switch operates 64 times per time slot. The results were obtained by our simulations and they were the same as the analytic results provided in[11]. We also performed a simulation for a 64×64 Clos network with input queueing and time multiplexing. We let all 8×8 crossbar switches in Clos network operate eight times per time slot which is the same operating speed as our shared buffer unit-switches.

A dashed line represents the cell loss probability at p = 0.9 of this Clos network. From this figure, we see that the proposed ATM switch outperforms other queueing approaches.

V. Conclusions

In this paper, we proposed a shared buffer switch that can be easily expanded to a large ATM switch by the three-stage interconnection. By integrating time-division and space-division switch architectures, we could achieve a linear scalability of the ATM switch with a high performance comparable to any other high performance switches. We used a distributed routing algorithm for our shared buffer switches. Therefore, the proposed approach eliminates the need for the central control of interconnected switches. With a synchronized operation of each shared buffer switch, we could guarantee the sequence integrity of ATM cell transmission without having extra hardware overhead. We could achieve a high performance by efficiently utilizing the unit-switches, and it is due to the evenly distributed load of traffic that results from our distributed routing algorithm.

The proposed ATM switch can be easily implemented with the current technology, and it is scalable to any size since we use identical unit-switches in three stages whose hardware complexity is linear in the number of ports. The global synchronization of all unit-switches can be easily done by the synchronized operations of internal counters of each switch unit with a global clock signal.

From the simulation results, it is shown that the proposed ATM switch achieves 100% switch throughput under uniform random traffic, and that it has a quite little performance degradation under bursty traffic conditions. It is also shown that the proposed switch, even with the three stages of queueing, provides a good performance in cell delay which is comparable to that of the best known queueing approach.

References

[1] M. Kawarasaki and B. Jabbari, "B-ISDN architecture and protocol," *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 9, pp. 1405-1415, Dec. 1991.

[2] H. Ahmadi, "A survey of modern high-performance switching techniques," *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 7, pp. 1091-1103, Sept. 1989.

[3] Newman, "ATM technology for corporate networks," *IEEE Communications Magazine*, pp. 90-101, April 1992.

[4] R. Rooholamini, V. Cherkassky, and M. Garver, "Finding the right ATM switch for the market," *IEEE Computer Magazine*, pp. 16-28, April 1994.

[5] T. Kozaki, et al., "32´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.

[6] J. Garcia-Haro and A. Jajszczyk, "ATM shared-memory switching architecture," *IEEE Network*, pp. 18-26, July/Aug. 1994.

[7] Y. Shobatake, et al., "A one-chip scalable 8´8 ATM switch LSI employing shared buffer architecture," *IEEE Journal on Selected Areas in Communications*, Vol. 9, No. 8, pp. 1248-1254, Oct. 1991.

[8] C. Clos, "A study of non-blocking switching networks," *Bell Syst. Tech. J.*, Vol. 32, pp. 406-424, March 1953.

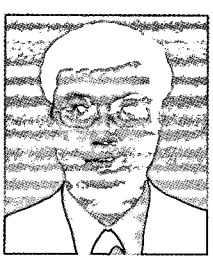
[9] Y. Yeh, M. Hluchyj, and A. Acampora, "The knockout switch: A simple, modular architecture for high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, Vol. SAC-5, pp. 1274-1283, Oct. 1987.

[10] H. Kuwahara, et al., "A shared buffer memory switch for an ATM exchange," in *Proc. Int. Conference on Communications (ICC'89)*, pp. 118-122, June 1989.

[11] M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE Journal on Selected Areas in Communications*, Vol. 6, No. 9, pp. 1587-1597, Dec. 1988.

[12] Y. K. Park and G. Lee, "NN-based ATM cell scheduling with queue length-based priority scheme," *IEEE Journal on Selected Areas in Communications - Computational and Artificial Intelligence in High Speed Networks*, Vol. 15, No. 2, pp. 261-270, Feb. 1997.

[13] Y. K. Park, V. Cherkassky, and G. Lee, "Omega network-based ATM switch with neural network-controlled bypass queueing and multiplexing," *IEEE Journal on Selected Areas in Communications-Intelligent Signal Processing in Communications*, Vol. 12, No. 9, pp. 1471-1480, Dec. 1994.



Young-Keun Park received the B.S. degree from Yonsei University, Seoul, Korea in 1986, and the M.S. and Ph.D. degrees from the University of Minnesota, Minneapolis, in 1990 and 1993, respectively, all in electrical engineering. From 1994 to 1995 he was with the Department of Electrical Engineering at the University of Minnesota. He is currently an Assistant Professor with the Department of Electrical Engineering, Yonsei University, Seoul, Korea. His research interests are in design of modular and scalable ATM switches, congestion controls in ATM networks, and neural network applications to optimization problems in high speed networks.