A Cost-Effective Design for ATM Switching Fabrics*

Nian-Feng Tzeng, Kiran Ponnuru, and Kemathat Vibhatavanij

Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, Louisiana 70504

Abstract — In this paper, we present a cost-effective design for ATM switching fabrics based on multistage structures, which involve no internal buffers at the constituent switching elements (SEs). The design consists of repeated copies of multiple stages of SEs, that are interconnected according to the indirect n-cube connection style between stages and that provide outlets for cells to terminate at their respective output queues when they reach their destined SEs, referred to as the *I-Cubeout*. This I-Cubeout makes use of far simpler SEs and requires fewer stages to achieve a given cell drop rate than the earlier design known as the Shuffleout. It routes cells in a distributed manner and exhibits low hardware complexity, making it suitable for the ATM switch resided in wireless base stations.

1 Introduction

Now that asynchronous transfer mode (ATM) has been standardized as the switching and transport technology for Broadband Integrated Services Digital Networks, it is essential to arrive at a cost-effective ATM switching fabric, which is scalable and can support high data transfer rates to meet the performance requirements. In particular, the wireless base station has been considered to incorporate ATM switching functionality (so that cell traffic is reduced and the handoff latency time is cut considerably) [18], and such a wireless ATM switch preferably should possess low hardware complexity and good scalability. ATM switching fabrics can be implemented by different techniques, all of which in use today belong to the bus, crossbar, shared memory, or multistage structure. A bus can be employed to link all the input and output ports, constituting the simplest switching fabric. It is naturally limited in scalability and speed because the ports share one single critical resource and the arbitration is needed among all active ports. A crossbar provides multiple simultaneous data paths through the fabric, and an input port is connected to an output port by setting a corresponding crosspoint to create a dedicated path. It relies on a scheduler to turn on and off crosspoints to establish simultaneous paths for transmission in parallel over the crossbar. The cost and the crosspoint control complexity of a crossbar grow rapidly as its size increases, limiting its scalability and speed. As a result, the crossbar switch in reality is limited to a small size, serving as a building block for constructing a larger switching fabric according to a certain interconnection style.

A shared-memory ATM switch employs buffer storage for keeping incoming cells (i.e., packets), which are to be routed over their destined output ports. From the storage utilization standpoint, it is efficient to have buffer storage *shared* by all the output (or input) ports. Several such switch designs have been reported [1-5], and they are based on either linked lists or content addressable memory (CAM). Designed by Foundation for Research and Technology, Greece, ATLAS I is a recent ATM switch implementation based on linked lists [3, 5]. In contrast, the ATM switch prototyped at the University of Toronto [2], replaces the linked list part with CAM, in which tags and the sequence number are employed as keys to search for cells in shared CAM. Since ATLAS I is optimized in switch architecture for implementation on a single chip and is a commercial product, we shall provide its design details in the next section.

A multistage ATM structure is commonly known as the space-division architecture, and it can be either non-blocking or blocking internally. An non-blocking structure normally requires complex, centralized control for path setup, potentially exhibiting limited scalability and a high cost. On the other hand, a blocking structure may fail to deliver two or more cells heading for distinct output ports due to contention within the structure, but it supports self-routing with distributed control and has lower hardware complexity, thus presenting good scalability. Every constituent switching element (SE) of a multistage ATM structure may include no buffer. Without internal buffers at each SE, an unbuffered blocking multistage structure enjoys the full advantage of hardware simplicity and self-routing capability. In this paper, our focus is limited to the blocking multistage ATM structures without internal buffers. When contention happens, all but one competing cell are deflection-routed (i.e, routed to the outputs other than their destined common one). This type of unbuffered blocking structures requires many stages of SEs to achieve an acceptably low cell drop rate (say, 10^{-6}) through deflection routing, with more stages leading to smaller drop rates. The tandem banyan switching fabric, formed by placing banyan networks in tandem, is an earlier unbuffered multistage blocking structure [6]. Rerouting networks [7], dual shuffle-exchange networks [8], bridged shuffle-exchange networks [9], and pipeline banyan networks [10] are all obtained by stacking identical copies of unbuffered multistage structures. Separately, a more efficient ATM switching architecture of this type, called the Shuffleout, is proposed [11], where each SE is equipped with outlets terminating at output queues so that cells are sent to their respective output queues through the outlets as soon as they reach destined SEs. Note that it may be able to reduce the number of stages required for a given drop rate by *recirculating* primary outputs back to primary inputs [12, 13]. This recirculating design, however, leads to longer mean cell delay and requires an input buffer associated with every primary input.

* This work was supported by the ARO under Grant/Cooperative Agreement No. DAAG55-98-1-0240.

0-7803-5284-X/99/$10.00 © 1999 IEEE.

This paper introduces a cost-effective design for ATM switching fabrics based on unbuffered blocking structures. Our ATM switch structure, referred to as the *I-Cubeout*, consists of multiple stages of SEs interconnected between consecutive stages according to the connection style of the indirect n-cube switching network [14]. Like the Shuffleout, the I-Cubeout provides each SE with outlets for terminating cells at their output queues. However, our I-Cubeout presents considerably lower hardware complexity than the Shuffleout, because it requires not only fewer stages of SEs (to achieve a given cell drop rate) but also far simpler SEs as the tags of cells are not recomputed inside the switching fabric. A deflection-routed cell in our I-Cubeout is not "corrected" immediately in the subsequent stages after deflection takes place (which is possible only through recomputing the tag), following a less greedy mechanism (than that employed in the Shuffleout). This mechanism is demonstrated to be beneficial, especially when the load is high. The I-Cubeout supports distributed self-routing and is readily scalable to a large size cost-effectively, suitable for ATM switch implementation on the wireless base station.

2 Pertinent Work

This section reviews briefly the ATLAS I ATM switch [3, 5] and the Shuffleout [11, 15], which are implemented on the basis of two different techniques, i.e., the shared memory and the multistage structure, respectively. They represent recent implementations of the two techniques. A hardware complexity comparison between these two designs is also provided.

2.1 ATLAS I Switch

ATLAS I is intended for use in high-throughput and low-latency communication systems, with point-to-point serial links running at 622 Mbits/s each. It involves advanced architectural features, such as the backpressure flow control protocol and the support of three service classes, multicasting, and shared memory for 256 cells. Shared memory in the ATLAS I (of size 16) is realized using multiple memory banks, addressed in a pipelined fashion [3]: the first unit of a cell is transferred to/from the first bank, followed by a "wave" of similar operations for the remaining units in the remaining banks in sequence. At the same time, the first unit of another incoming cell (from the next input port) is transferred to/from the second bank, followed by a sequence of similar operations for the remaining units in the remaining banks. Shared memory is the only storage for all the input ports to keep (i.e., write) incoming cells and for all the output ports to get (i.e., read) the cells for delivery. If the memory bank width is of, say, 16 bits, the accumulation of an incoming string (at the rate of 622 Mbits/s) at a read/write register takes about 26 ns, which is longer than the read/write cycle of a typical SRAM. Since ATLAS I has 16 input and 16 output ports, it requires 16 (or 32) memory banks or more in total to constitute the shared storage, if the memory access cycle time is no longer than 13 ns (or 26 ns). The management of shared storage requires considerable logics, whose complexity will be examined later in this section.

There are three service classes supported by ATLAS I, so every output port is associated with three sets of pointers, one for each service class. Each set contains the head and the tail pointers (to the shared storage where the cell bodies, which are to be transferred through the output port, are kept). When an incoming cell arrives and is determined (by looking up the routing table) to be delivered through an output port, say port p, the tail pointer of the class to which the cell belongs, for port p is modified. All cells of the same service class at port p are "linked" together, with the "next" field of one cell pointing to the next cell. Multicast cells are handled by separate sets of pointers, which are shared by all output ports. Shared memory thus keeps the cell body as well as the "next" field necessary to form link lists of cells.

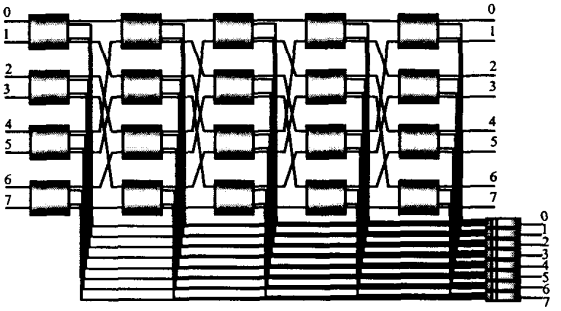


Fig. 1. Shuffleout switch with $N = 8$, $k = 5$, and $b = 2$.

2.2 Shuffleout Switch

The Shuffleout switch consists of multiple stages of unbuffered SEs, which are basically crossbar switches, with adjacent stages interconnected by a shuffle connection pattern. An SE, say, of size b x $2b$, has b switch outlets terminating at b output queues, which are fed by cells received on links of stages where they reach their destined rows. If a cell fails to reach its destined row after traversing all the stages, the cell is dropped at the output side of the last stage. Fig. 1 depicts the Shuffleout switch of size (N =) 8 composed of (k =) 5 stages of 2 x 4 SEs, where each SE has two switch outlets terminating at two output queues and each queue is fed by cells received at the five constituent stages. There are four SEs in each stage of the Shuffleout switch.

In order to perform routing over a shortest path for a cell in the Shuffleout switch, each cell is provided with an additional field, called *tag*, which specifies the address of the required switch outlet for the cell. On receiving a cell, the SE always attempts to route the cell along the shortest path to the destined switch outlet, where the shortest path involves the fewest interstage hops needed to reach its destined outlet. If two cells at an SE require the same switch outlet to an SE in the next stage (or to an output queue), called a *remote* switch outlet (or a *local* switch outlet), deflection routing is applied so that the cell closest to its destined switch outlet is routed along the shortest path, whereas the other cell is deflection-routed over another remote switch outlet. When a cell arrives at an SE in any stage of the Shuffleout switch, its tag field is updated by the distance computing block of the SE through recomputing the distance of the cell from the current SE to its destined SE for identifying the SE remote switch outlet which is on the shortest path [15]. This recomputation of the tag field is necessary at every SE, whether the cell is routed over a shortest path or by deflection routing.

2.3 Comparison of Switch Designs

As described above, ATLAS I represents an advanced switching structure with shared memory, whereas the Shuffleout comprises multiple stages of unbuffered SEs with output queues, which are reachable from every stage. The following compares these two different switch designs quantitatively. An unbuffered multistage switch design is shown to exhibit lower hardware complexity, possibly a preferred choice of ATM switch structures.

Including shared memory capable of holding 256 cells, ATLAS I is of size 16 and implements a pipelined memory organization [3] to support the rate of 622 Mbits/s on every port. Its shared memory is realized by static RAM and requires access control logics plus link list management logics for keeping cells in it. Access control logics for shared memory involve sense amplifiers, bitline drivers, wordline register/driver, decoders, predecoders, incoming and outgoing link drivers, incoming and outgoing link control pipeline latches, which take up roughly 152K transistors for implementation [16, 17]. Since link lists management utilizes 54 sets of pointers, one set for a service class at an output port, the logics for managing these pointers as well as the next field of cells in shared memory to form linked lists take thousands of transistors to implement. Every input port requires a direct connection to these 54 sets of pointers, and different ports take different connections to these sets of pointers; a 1 x 54 demultiplexor and a 16 x 1 multiplexor are thus associated with each input port and each pointer set, respectively. A Shuffleout switch of size 16 requires 15 stages of 2 x 4 SEs to yield no cell drop under load = 1.0, according to our simulation results. Each SE consists of four key parts [15]: distance computing blocks (2 of them), a commutation block, outlet identifiers (2), an output (de)multiplexor. The distance computing block (for one input port) of a 2 x 4 SE requires slightly more than 200 transistors. The two distance computing blocks provide inputs to a logic, known as the *commutation block*, which sets the output (de)multiplexer according to the routing strategy, including conflict resolution and the use of local switch outlets. It takes about 800 transistors to implement one SE. The Shuffleout switch of interest comprises 120 (i.e., 8*15) such 2 x 4 SEs. These SEs have local switch outlets terminating at 16 output queues. Each output queue is preceded with a 15-to-1 multiplexor (from the 15 stages of SEs) and is of capacity 16 (found by our simulation to yield a smaller cell drop rate than shared memory of capacity 256 cells under load equal to 1.0). As a result, the Shuffleout switch requires a total of 96K transistors for the implementation of switching stages and 16 output queues capable of holding 256 (i.e., 16*16) cells in total. When compared with the ATLAS I (a buffered ATM switch based on shared memory), where access control logics for shared memory (of capacity 256 cells) alone take about 152K transistors, with link lists management utilizing 54 sets of pointers and demultiplexors plus multiplexors in existence from inputs to all sets of pointers, the Shuffleout switch is clearly favorable in terms of hardware complexity. This suggests that an unbuffered multistage ATM structure is advantageous (in terms of hardware) over the shared-memory ATM structure, besides the benefits that the multistage ATM structure enjoys distributed routing and provides a dedicated path from an input to an output port.

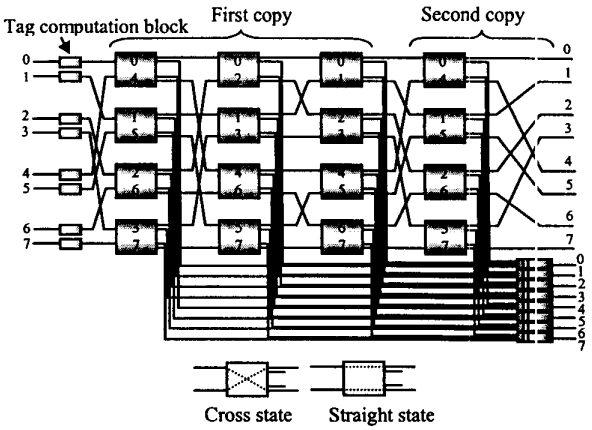


Fig. 2. I-Cubeout switch with $N = 8$, $k = 4$, and $b = 2$.

We shall introduce below an ATM switch based on multiple stages of unbuffered SEs, like the Shuffleout switch. However, the SE used is much simpler and the number of stages needed is slightly smaller, resulting in far lower hardware complexity.

3 Proposed I-Cubeout Switch

Our introduced ATM switch comprises stages of b x $2b$ SEs, with b outlets of each SE being remote (for connecting SEs in the next stage) and the other b outlets being local (for terminating cells at output queues). For simplicity, we explain in the following, the case of b equal to 2 only. A switch of size $N = 8$ is depicted in Fig. 2, where the interconnection patterns between stages follow those of the indirect n-cube network [14]. Specifically, the two remote outlets of any SE in the same stage differ in their addresses by a constant, namely, those in the first (i.e., leftmost) stage differ by $N/2$, those in the second stage differ by $N/4$, and so on, for a switching fabric of size N ($= 2^n$). At the n^{th} stage, the two remote outlets of any SE differ by 1. A *copy* of the indirect n-cube connection ranges from the first stage to stage n, as depicted in Fig. 2. In stage $(n+1)$, the two remote outlets of an SE differ by $N/2$, identical to the situation of the first stage. This stage starts another copy of the indirect n-cube connection, which spans the next n stages, including stage $(n+1)$. This introduced ATM switch, called *I-Cubeout*, may involve any number of stages, not necessarily an exact multiple of n. All remote outlets with the same address, say d, are connected across the stages, as shown in Fig. 2, and every such outlet has an associated local outlet which terminates at output queue d.

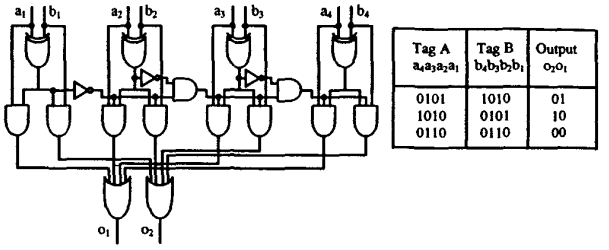
This I-Cubeout employs distributed self-routing, with the routing tag computed at the primary input (as demonstrated in Fig. 2). Like routing in the hypercube, the I-Cubeout computes the tag of a cell according to its source and destination addresses through an XOR operation at a primary input. The tag of a cell so computed is carried along with the cell when traveling through the I-Cubeout to guide the traversal of the cell over the switch.

3.1 Routing Algorithm

The tag of a cell, obtained by performing an XOR operation on the source and the destination addresses of the cell at the source (i.e., primary input), represents the positions needed to be "corrected" before getting to its destination. A tag bit position corresponds to certain stages of the I-Cubeout switch, which is composed of repeated copies of n stages of SEs (interconnected according to the indirect n-cube patterns [14]) in sequence, with the last copy containing an arbitrary number of stages of SEs (not necessarily n stages). The first (i.e., leftmost) tag bit position is used by SEs in the first stage of any copy, the second tag bit position is by SEs in the second stage of any copy, and so on. If a tag bit, say position p, is nonzero, the cell takes a "cross" state at the visited SE in stage p of any copy, where a "cross" state indicates that the upper (lower) inlet of an SE is connected to the lower (upper) remote outlet (see Fig. 2). This reflects the situation that the cell is "corrected" at position p, and the correction can be done in stage p of any copy. Once a correction is done at the SE by setting it to a cross state for the cell, the tag bit p of the cell is reset to zero. If the tag bit is zero, on the other hand, the cell takes a "straight" state at the visited SE, since no correction is needed at that position. When the tag bits all become zero, the cell has reached its destination and may take a local outlet to arrive at its destination queue.

To simplify the SE design, the tag of a cell is rotated leftward by one (1) bit position after it traverses a stage, so that the tag bit position to be examined at any stage is always the leftmost one. The number of tag bits equals the number of stages in a copy. When two cells at an SE in stage p have distinct values in tag bit p, i.e., one equal to zero and the other equal to nonzero, a conflict occurs, because the SE can be set to either the cross or the straight state at a time, but not both. Only one of the two conflicting cells is forwarded to its destined outlet, with the other being deflection-routed. If the SE with a conflict is set to the cross state, both cells have their tag bit position p complemented, including the deflection-routed one, which then has to be corrected in stage p of a later copy. If the SE is set to the straight stage, both cells keep their tags unchanged after traveling through the SE. In either case, the deflection-routed cell cannot have its tag bit position p corrected until stage p of the next copy (which is n stages away from the place where deflection routing happens). This is different from the routing technique employed in the Shuffleout, where a deflection-routed cell is to have the tag bit position corrected immediately in a subsequent stage (by recomputing the tag in every stage). Our routing algorithm is less greedy, and therefore advantageous when the load is high, as will be demonstrated in Sec. 4.

If a conflict happens at an SE, the cell with a smaller distance to its destination is given priority, where the distance is defined as the minimum number of stages (i.e., hops) a cell has to travel before getting to its destination. The distance of a cell can be told directly from its current tag (which has been rotated leftward by r bit positions if the cell is at stage $(r+1)$ of any copy). Specifically, the distance of a cell is the rightmost nonzero bit position of its tag.



Tag A $a_4a_3a_2a_1$	Tag B $b_4b_3b_2b_1$	Output o_2o_1
0101	1010	01
1010	0101	10
0110	0110	00

Fig. 3. 4-bit priority resolution logic.

The rationale behind giving priority to the cell with a smaller distance is to let that cell reach its destination as soon as possible, and this is more likely to achieve because it has fewer stages left to travel, freeing resources for other cells. If two conflicting cells have an identical distance to their respective destinations, a random one is chosen to give priority, with the other one deflection-routed. (Note that when two tags have the same distance, our logic given in Fig. 3 compares their second non-zero bit positions.) A deflection-routed cell always has a larger distance than, and thus yields to, a non-deflection-routed one, with which it conflicts at an SE in a later stage. This naturally keeps deflection-routed cells as few as possible, making best utilization of resources.

3.2 Design Consideration

Unlike the Shuffleout, our I-Cubeout never recomputes the tag of a cell, even after the cell is deflection-routed at an SE. This makes the constituent SEs of our I-Cubeout much simpler and more scalable than those of the Shuffleout. Every SE in the Shuffleout switch is equipped with the distance computing blocks, one for an input port. A diagram of such a block is given in [15]. The complexity of the block grows in the order of approximately $O(m^3)$, where m equals $log_2 N$ for a switch of size N comprising 2 x 4 SEs. This rapid growth in hardware complexity makes it prohibitly expensive to construct SEs for a large Shuffleout switch, limiting scalability. In addition, the commutation block [15] in every SE of the Shuffleout becomes more complicated when the size of SEs increases, since it is fed with outputs of all distance computing blocks in an SE. The priority resolution logic in our SE is realized by comparing the distances of two tags in an SE, as illustrated in Fig. 3, where output $o_2\, o_1 = 01$ (or 10) indicates that tag $A = a_4\, a_3\, a_2\, a_1$ has a larger (or smaller) distance than tag $B = b_4\, b_3\, b_2\, b_1$, and output 00 indicates identical tags.

The I-Cubeout has a separate output queue for each destination. When the queue is full, a feedback signal is sent back to the switch stages so that those local outlets of involved SEs will be disabled, preventing cells from leaving the switch stages (as otherwise they will be dropped) and forcing affected cells which have already reached their destined SEs to be deflection-routed. The signal goes away as soon as a slot in the queue becomes available. This effectively makes SEs in later switch stages more utilized by cells as the load grows. It leads to a lower drop rate than the shared memory design (with the same storage capacity as I-Cubeout's all output queues combined) for load = 1.0.

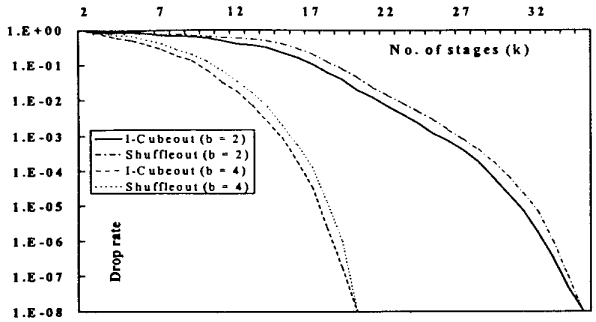


Fig. 4. Results for the switch size of 256 with load = 1.0.

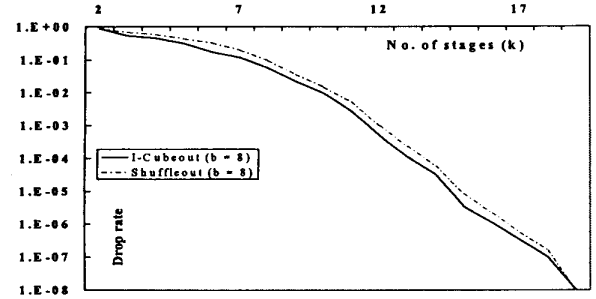


Fig. 5. Results for the switch size of 512 with load = 1.0.

4 Performance Evaluation

We evaluate the performance of our I-Cubeout for various sizes composed of different types of SEs (including 2 x 4 SEs, 4 x 8 SEs, and 8 x 16 SEs) using simulation and compare the results with those of the Shuffleout. Each source is assumed to generate cells following an independent Bernoulli process under load = 1.0. Cell drop rate versus number of stages (k) is shown in Fig. 4 and Fig. 5, where each data value is the result after 200,000 clock cycles in our simulation (note that this number of cycles has been observed to give rise to stable results).

For the switch size of 256, our I-Cubeout requires slightly fewer stages to achieve any given drop rate, as can be seen in Fig. 4. This is in part due to our use of a feedback signal from the output queue which is full, back to the associated SEs in switch stages so that no cells are then allowed to leave the switching fabric and those cells which have already reached their destined SEs are deflection-routed, and in part due to our "less greedy" routing mechanism. The I-Cubeout has a similar saving in the number of stages with 8 x 16 SEs for the switch size of 512, as found in Fig. 5. Given a drop rate of 10^{-6}, for example, the I-Cubeout of size 512 takes only 15 stages of 8 x 16 SEs, as opposed to 16 stages for the Shuffleout. This saving is on top of the fact that the 8 x 16 SE is far simpler in our I-Cubeout than in the Shuffleout. Our described ATM structure apparently enjoys significantly reduced hardware complexity and better scalability.

5 Conclusions

A cost-effective ATM switch design has been presented and evaluated. The design is based on an unbuffered multistage structure, with consecutive stages interconnected according to the indirect n-cube style, and it provides outlets from each stage for terminating cells at output queues as soon as the cells reach their destined SEs (and the corresponding output queues are not full), referred to as the I-Cubeout. Our I-Cubeout follows distributed routing and exhibits lower hardware complexity than earlier designs, including the Shuffleout [11], which compares favorably with the Tandem Banyan [6] and the rerouting network [7], among others. This results directly from two facts: (1) cell tags are never recomputed inside the switching fabric (unlike tags in the Shuffleout), making the SE much simpler, and (2) the number of stages needed to yield a given drop rate is slightly fewer than that required by the Shuffleout. The I-Cubeout is therefore better scalable and more cost-effective, readily suitable for large ATM switch construction.

References

[1] N. Endo *et al.*, "Shared Buffer Memory Switch for an ATM Exchange," *IEEE Trans. on Commun.*, vol. 41, pp. 237-245, Jan. 1993.

[2] K. J. Schultz and P. G. Gulak, "CAM-Based Single-Chip Shared Buffer ATM Switch," *Proc. 1994 IEEE ICC*, June 1994, pp. 1190-1195.

[3] M. Katevenis, P. Vatsolaki, and A. Efthymiou, "Pipelined Memory Shared Buffer for VLSI Switches," *Proc. ACM SIGCOMM'95*, Aug. 1995, pp. 39-48.

[4] H. Yamanaka *et al.*, "Scalable Shared-Buffering ATM Switch with a Versatile Searchable Queue" *IEEE J. on Sel. Areas in Commun.*, vol. 15, pp. 773-784, June 1997.

[5] M. Katevenis, D. Serpanos, and E. Spyridakis, "Switching Fabrics with Internal Backpressure Using the ATLAS I Single-Chip ATM Switch," *Proc. GLOBECOM'97*, Nov. 1997, pp. 242-246.

[6] F. A. Tobagi, T. Kwok, and F. M. Chiussi, "Architecture, Performance, and Implementation of the Tandem Banyan Fast Packet Switch," *IEEE J. on Sel. Areas in Commun.*, vol. 9, pp. 1173-1193, Oct. 1991.

[7] S. Urushidani, "Rerouting Network: A High-Performance Self-Routing Switch for B-ISDN, *IEEE J. on Sel. Areas in Commun.*, vol. 9, pp. 1194-1204, Oct. 1991.

[8] T. Lee and S. Liew, "NlogN Dual Shuffle-Exchange Network with Error-Correcting Routing, *Proc. 1992 IEEE ICC*, June 1992, pp. 311.3.1-311.3.7.

[9] R. Zarour and H. T. Mouftah, "Bridged Shuffle-Exchange Network: A High Performance Self-Routing ATM Switch," *Proc. 1993 IEEE ICC*, June 1992, pp. 696-700.

[10] P. C. Wong and M. S. Yeung, "Design and Analysis of a Novel Fast Packet Switch - Pipeline Banyan," *IEEE/ACM Trans. on Network.*, vol. 3, no. 1, pp. 63-69, Feb. 1995.

[11] S. Bassi *et al.*, "Multistage Shuffle Networks with Shortest Path and Deflection Routing for High Performance ATM Switching: The Open-Loop Shuffleout, *IEEE Trans. on Commun.*, vol. 42, pp. 2881-2889, Oct. 1994.

[12] S. Bassi *et al.*, "Multistage Shuffle Networks with Shortest Path and Deflection Routing for High Performance ATM Switching: The Closed-Loop Shuffleout," *IEEE Trans. on Commun.*, vol. 42, pp. 3034-3044, Nov. 1994.

[13] H.-I. Lee *et al.*, "A High Performance ATM Switch Based on the Augmented Composite Banyan Network," *Proc. 1998 IEEE ICC*, June 1998, pp. 309-313.

[14] M. C. Pease, III, "The Indirect Binary n-Cube Microprocessor Array," *IEEE Trans. on Comp.*, vol. C-26, pp. 250-265, May 1977.

[15] M. Decina, P. Giacomazzi, and A. Pattavina, "Shuffle Interconnection Networks with Deflection Routing for ATM Switching: the Open-Loop Shuffleout," *Proc. 13th Int'l Teletraffic Cong.*, June 1991, pp. 27-34.

[16] A. Efthymiou, "Design, Implementation, and Testing of a 25Gb/s Pipelined Memory Switch buffer in Full-Custom CMOS," FORTH-ICS Technical Report - 143, Nov. 1995.

[17] A. Efthymiou, private communications, Apr. 1998.

[18] M. Veeraraghavan, M. J. Karol, and K. Y. Eng, "Mobility and Connection Management in a Wireless ATM LAN," *IEEE J. on Sel Areas in Commun.*,vol. 15, pp. 50-68, Jan. 1997