

# CAM-Based Single-Chip Shared Buffer ATM Switch

Kenneth J. Schultz and P. Glenn Gulak

Dept. of Electrical and Computer Engineering

University of Toronto

Toronto, Ontario M5S 1A4 CANADA

**Abstract:** A novel single-chip shared buffer ATM switch architecture is presented, in which a Content Addressable Memory (CAM) is used to control access to the shared buffer RAM, in place of a linked list mechanism. Switch operation is explained in detail, and performance is compared to that of a Linked List switch. Memory capacity requirements are decreased, and cell storage and retrieval is simplified. Additional features are easily added, including priority handling and the first reported architectural support for multicasting in a single-chip shared buffer ATM switch. A novel Parallel-to-Serial memory is presented as an interface between the shared buffer and the output ports. Speed, area, power dissipation, and other physical performance parameters, are estimated.

## I. INTRODUCTION

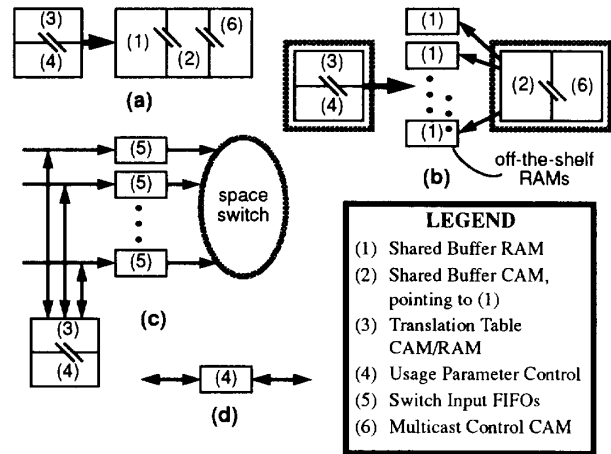
ATM (Asynchronous Transfer Mode) will provide the transport and switching environment for emerging B-ISDN networks, and has thus recently been the subject of intense research interest. In particular, ATM switch hardware design has been studied extensively, and this study "has reached a significant degree of understanding" [1]. Despite the maturity of this characterization effort, architectural innovation can still be achieved. This paper introduces a novel switch design that takes advantage of application-specific memories, which are available in a full-custom design environment.

Although researchers have shown significant interest in Batcherman and other space switches, shared buffer (or common memory) topologies offer a superior implementation alternative for switching nodes, in terms of testability, scalability, cell loss rate, and switch throughput [2]. Limitations due to internal, external, and head-of-line blocking are avoided, and buffer capacity requirements are decreased drastically due to sharing [3]. Unfortunately, read and write access to the shared buffer becomes the switch throughput bottleneck. A means of increasing speed at this bottleneck is the integration of the entire switching node on a single IC [4,5,6]. Although some functionality may be sacrificed in exchange for such aggressive integration, it is in keeping with a general industry-wide trend toward systems-on-a-chip [7], leading toward the implementation of entire thousand-channel switching systems on one or two chips.

This paper proposes the use of a Content Addressable Memory (CAM) in a single-chip shared buffer ATM switch, providing architectural novelty and benefits, as discussed in Section II. Basic performance is described in Section III, and additional features — efficiently implemented in this architecture — are the subject of Sections IV and V. Physical implementation issues are addressed in Section VI, and a summary is presented in Section VII.

## II. CAMS AND ATM SWITCHING

Most commercial ATM switch implementations use off-the-shelf components and gate arrays. This implementation environment necessarily limits the architectural design. Unfortunately, the resulting architectures have migrated to single-chip full-custom silicon switches, placing unnecessary constraints on their design space. Custom silicon, in these cases, is treated as an assemblage of off-the-shelf components, and custom approaches are conservatively avoided. But true technologi-



**Figure 1:** Drop-in components in four ATM switch configurations: (a) single-chip shared buffer switch; (b) large shared buffer switch; (c) input-buffered switch; (d) line card or workstation/LAN interface.

cal leverage and design differentiation require both novel circuits and novel architectures. ASIC memories [8] can provide this differentiation and leverage in memory-intensive circuits such as switches. Specifically, CAMs [9] can be used as the basis for a number of new ATM circuits and architectures.

We recognize six memory-intensive ATM circuits that may be implemented using CAMs; these are shown in Figure 1, and enumerated in the legend. Four example switch configurations are shown, demonstrating the versatility of these CAM-based circuits (some of the memories may be used in a number of different switch types). As illustrated, CAMs are useful in ATM time switches, space switches, and terminal interfaces. This paper focuses on the shared buffer portion of the complete single-chip switch shown in Figure 1(a), comprising components (1), (2), and (6); an alternative implementation, shown in Figure 1(b), allows for increased cell storage through the use of commodity RAMs in combination with custom CAM-based memory and multicast control. Elsewhere, we describe architectures for (3) and (5) [10,11].

These CAM-based circuits, when designed to enable expandability and modularity, can be used to implement (for example) a scalable translation table with 8 entries or 8K entries, or a shared buffer switch with 4 ports or 64 ports. The combination of versatility and modularity enables a "drop-in component" approach, which provides considerable benefit due to design re-use.

## III. BASIC OPERATION

### A. Review of Linked List Operations

Shared buffer switches typically use a Linked List to control memory access [3]; the cells destined for each output port are chained through this mechanism. Each cell  $C[o][j]$  is stored in the shared buffer (at address  $A[o][j]$ ) together with the address of the next cell destined for the same output,  $A[o][j+1]$ . This "next address" is also stored in the register  $WA[o]$ , for the most recently written cell destined for each output.

On a write:

1. The address  $A[o][j+1]$  is read from  $WA[o]$ .
2. An address  $A[o][j+2]$ , corresponding to an empty memory location, is read from  $IAF[h]$  (the head of the Idle Address FIFO).
3. The cell is written into the buffer  $B[A[o][j+1]]=C[o][j+1]$ , along with  $A[o][j+2]$ .
4.  $WA[o]$  is over-written with  $A[o][j+2]$ .

On a read:

1. The buffer address  $A[o][k]$  is read from the register  $RA[o]$ .
2. The cell is read from  $B[A[o][k]]$ .
3. The buffer address has now become idle (free) so the tail of the FIFO  $IAF[t]=A[o][k]$ .
4. The address  $A[o][k+1]$  (stored along with the cell just read) is written into  $RA[o]$ .

FIFO pointers must be initialized and maintained, along with registers for write and read addresses. Though this technique is quite complex, it is well-understood and widely used. A more efficient and elegant technique, if available, could simplify not only the basic switch design, but also the addition of features such as priority handling and multicasting.

### B. CAM-Based Operations

The bulk of the processing in the above algorithm is concerned with the manipulation of addresses. Yet, the design goal is to efficiently switch ATM cells; address manipulation is merely a means of achieving that goal. By using CAMs, we greatly simplify storage and retrieval of the cells, and are thus able to focus resources on the primary task.

The shared buffer RAM is replaced by a CAM/RAM structure, where the RAM stores the cell and the CAM stores a tag used to reference the cell. A cell can be uniquely identified by its output port number and a sequence number, and these together constitute the tag. Cells are accessed for reads by searching for the desired tag. Incoming cells are written into the first free location, the address of which is irrelevant. We refer to this architecture as the "CAM-Access" approach.

Figure 2 shows switch architecture. Write circuits service input ports on a round-robin basis, and read circuits service output ports on a round-robin basis. Operation at each step in the round-robin is as follows:

For a write:

1. Read the write sequence number (corresponding to the destination port)  $WS[o]$ , and use this value in the cell's tag  $\{o, WS[o]\}$ .
2. Search the tag CAM for the first empty location  $emp$ .
3. Write the cell into the buffer  $B[emp]=C[o][j]$ , and  $\{o, WS[o]\}$  into the associated tag.
4. Increment the sequence number  $WS[o]++$  and re-write it.

For a read:

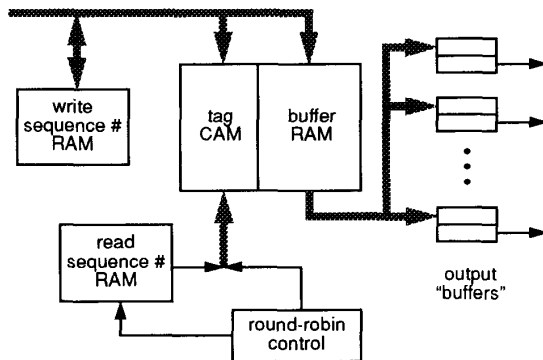


Figure 2: CAM-based shared buffer switch basic architecture

1. Read the read sequence number (corresponding to the destination port)  $RS[o]$ .
2. Search for the tag with the value  $\{o, RS[o]\}$ .
3. Read the cell in the buffer associated with the tag  $\{o, RS[o]\}$ ; this is  $C[o][k]$ .
4. Increment the sequence number  $RS[o]++$  and re-write it.

This technique replaces link storage (indirectly) with content-addressable tag storage, and replaces read and write pointers and registers with sequence numbers. A single extra "valid" bit, added to each tag, identifies empty buffer words, effectively replacing the IAF (and its pointers). Since no address decoding is used by the CAM/RAM buffer, no decoder is required. As well, since the CAM need not output the address of tag matches ("hits"), it requires no address encoder (usually a source of CAM area overhead). Both Linked List address registers and CAM-Access sequence number registers must be initialized to known values on power-up. If queue lengths need to be monitored, additional counters are required for the Linked List case, but sequence numbers can simply be subtracted in the CAM-Access case. Table 1 summarizes this comparison, and provides bit counts for an example single-chip configuration.

Table 1: Comparison of Linked List and CAM-Access. (sample switch size is  $16 \times 16$  with  $2^8 = 256$  cell buffer capacity; 7-bit sequence numbers used)

	Linked List	CAM-Access
cell storage (decode/encode) bits	RAM (decode) $256 \times 424 = 108,544$	CAM/RAM (neither) $256 \times 424 = 108,544$
look-up bits	link: RAM $256 \times 8 = 2,048$	tag: CAM $256 \times (4+7) = 2,816$
write and read reference (queue length checking) bits	address registers (additional counters) $2 \times 16 \times 8 = 256$	sequence number registers (compare W and R numbers) $2 \times 16 \times 7 = 224$
idle address storage (additional overhead) bits	IAF (pointer maintenance, extra memory block) $256 \times 8 = 2,048$	CAM valid bit (none) $256 \times 1 = 256$
TOTAL BITS	112,896	111,840

While CAM core cells are inherently more complex than RAM core cells, area overhead is cancelled out since the CAM storage replaces both RAM link storage and address decoders. Likewise, speed is not compromised — CAM/RAM access rates in excess of 150 MHz have been demonstrated using CMOS technology [12]. Hence, we pay no area or time penalty for a more elegant implementation.

Switch operation is pipelined, as illustrated in Figure 3. The figure shows resources required by each operation. For maximum switch throughput, write and read operations must proceed simultaneously. However, whenever both pipelines are full ( $t=3$  and  $t=4$  in Figure 3), this results in a resource conflict for the CAM, RAM, and sequence number registers. The switch designer must choose to either (1) interleave write and read accesses, cutting switch throughput in half, or (2) use dual port CAM and RAM cells, introducing an area overhead of approximately 60%. This problem is not specific to this architecture; the Linked List architecture requires a dual port shared buffer for full-speed operation, as well as dual port write and read address registers and IAF.

### C. Priority Handling

Cell loss priority is defined by a single CLP bit in the header, according to CCITT standards.  $CLP=1$  cells are discarded preferentially over  $CLP=0$  cells as the shared buffer fills near capacity. Using CAM-Access, one may append an extra bit to the tag, and thus divide the CAM/RAM into two portions: one reserved for  $CLP=0$  cells and a smaller one reserved for  $CLP=1$  cells. Furthermore, this extra tag bit is

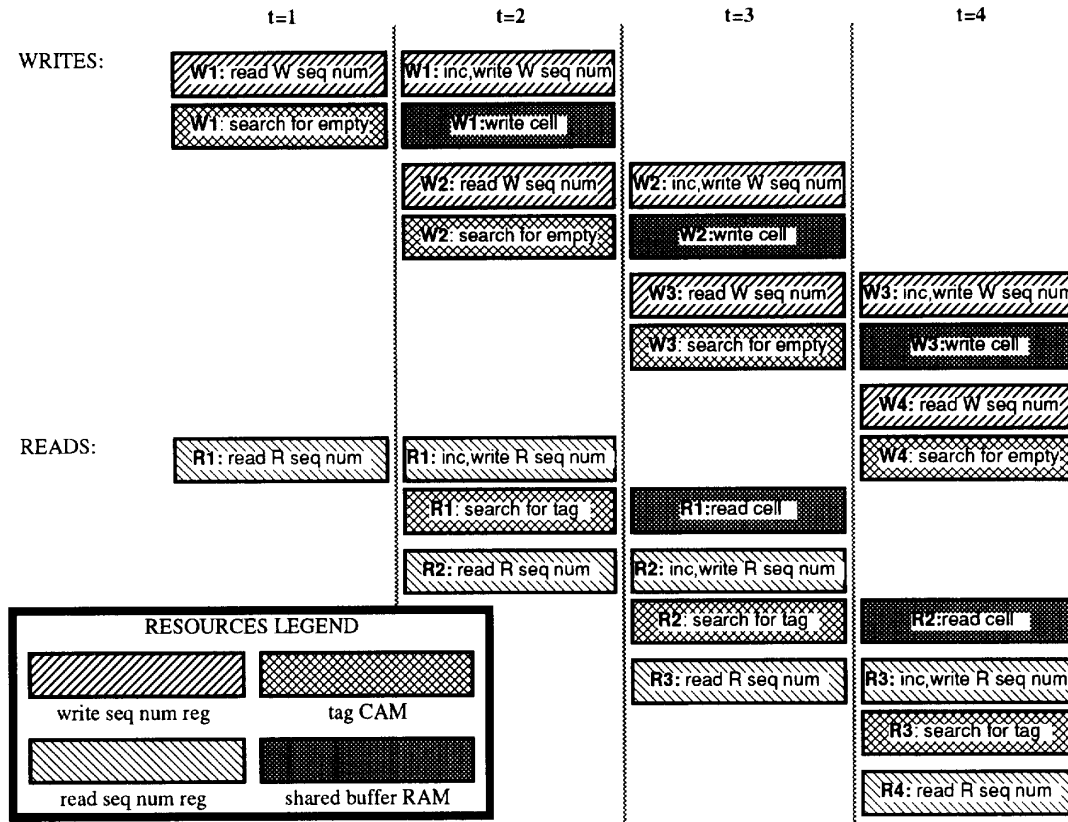


Figure 3: Pipelined operation of the CAM-based switch.

alterable in real-time, hence allowing network supervisory commands to adjust buffer allotment based on admitted connections and traffic fluctuations. Taking this approach a step further, multiple tag bits could be used to differentiate between three or more classes of CAM/RAM storage, as required by network control. Perhaps one of these new classes ("X") could be shared between CLP=0 cells, and CLP=1 cells admitted beyond a "danger" threshold. Once classes "0" and "X" fill up, CLP=0 cells are permitted to over-write (or "push out") CLP=1 cells in class "X". This requires special attention regarding sequence integrity of CLP=1 cells in class "X"; if the class is small enough, sequence integrity could be guaranteed by dedicated hardware.

Delay priority is negotiated on call set-up and maximum latencies must be ensured by the switch. This is implemented by replicating write and read sequence number registers for each delay priority level desired. For a 4-level system, each output port requires 4 separate sequences of cells, one per level. When an output port is serviced for reading, higher priority sequences are read preferentially. This is accomplished by setting an "empty" flag for each read sequence register, once it catches up to its corresponding write sequence number (the write sequence number is forwarded for comparison along with each cell written). If the "priority 00" queue is empty, the "priority 01" queue is serviced, and so on.

#### IV. MULTICASTING

No existing commercial ATM switch directly supports multicasting, and its implementation on single-chip shared buffer switches has not been described in the literature. Yet, multicasting is a vital function for the support of both desired ISDN services (such as video conference calls) and profit-generating services of the future (such as video-on-demand/cable"). For most efficient use of switch input bandwidth and

buffer capacity, a multicast cell should be written only once and stored in only one location.

Using the CAM-based switch architecture we have described, unicast cells are identified based on their output port. In the case of a multicast cell, there are multiple output ports, and a different means of identification is required. Each multicast connection is assigned a Multicast Connection Identifier (MCI), which is unique and significant only within a single switching node. Sequence numbers are assigned for each MCI. The number of bits required to uniquely identify a multicast cell (MCI plus sequence number) must be the same as that required to identify a unicast cell (output port plus sequence number), because both identifiers must fit in a single word of the tag CAM. Optimum partitioning of bits between MCI and sequence number would depend on traffic patterns, and — most significantly — on the maximum number of simultaneously active multicast connections. For example, in the case of the 16x16 switch of Table 1, we may choose to identify multicast cells by a 6-bit MCI and a 5-bit sequence number.

The switch hardware must keep track of the output ports associated with each multicast connection. This is accomplished using a bit-mapped multicast CAM (McCAM), as shown in Figure 4. Each word of the CAM represents a multicast connection, and each column represents an output port. A word is written into the McCAM on call set-up, such that a 1 appears in bit positions corresponding to destination ports, and 0's appear in all other bit positions. This memory is similar to the Scheduling CAM described in [13].

With only a single copy of the multicast cell residing in memory, we could either (1) read it once and send it to all destination ports simultaneously, or (2) perform separate reads for each destination port, and

clear the buffer entry only after all destination ports have been serviced. Most efficient use of shared buffer read bandwidth is obtained if the cell is read only once. This requires that all destination ports be available to receive the cell from the output bus during the same memory clock cycle (see Figure 2).

All multicast connections, as a group, are assigned one or more slots in the output round-robin. The relationship between buffer and output port bandwidth is such that an output port can only receive a single cell in each round-robin cycle. When it is the multicasts' "turn", the McCAM is searched against an input word with X's (don't cares) in bit positions corresponding to free output ports, and 0's corresponding to ports already occupied. A 1 stored in the CAM in the same bit as a 0 in the input word results in a miss (shown as an arrow in Figure 4), since the cell belonging to that multicast cannot be released to all destination ports simultaneously. If there is at least one hit, the MCI with the highest priority<sup>1</sup> is read from the shared buffer, and latched at all destination ports. These destination ports cannot receive another cell during the current round-robin cycle.

If multicasts always have the first slot in the round-robin, they may continually inhibit the reading of unicast cells (including, potentially, some with strict latency requirements). If they always occupy the last slot, no multicast cells could ever be read. Thus, this approach requires the scheduling of the multicast slot to vary from cycle to cycle (from first down to, but not including, last). In this case, the order of port access for unicasts must also be rotated from cycle to cycle to avoid shutting out the port serviced last. The following pseudo-code (with mod-8 arithmetic) describes this scheduling mechanism for an 8x8 switch, and Table 2 shows a resulting schedule.

```

mcslot = 0
for i from 0 to 7 : unislot[i] = i-1
for cycle from 0 to forever
  if (cycle mod 8 == 0) then
    for i from 0 to 7 : unislot[i]++
  else continue
  k = 0
  for slot from 0 to 8
    if slot == mcslot then
      service multicast
      k = 1
    else service unicast unislot[slot-k]
  end for
  mcslot++
  for i from 0 to 7 : unislot[i]++
end for

```

More flexibility and increased multicast bandwidth are attainable if multicasts are allotted multiple slots per cycle. The hardware may be modified to allow selectable multicast to unicast service ratios, in response to varying traffic mixes.

## V. EFFICIENT OUTPUT CIRCUITRY

The main bandwidth bottleneck in a shared buffer switch is the buffer, and its bandwidth can be expressed as the product of clock frequency (discussed in Section VI) and word width. In a full-custom environment, the switch designer has the luxury of choosing the memory word width. In an ATM switch, maximum bandwidth is attainable if the entire 53-byte (424-bit) cell constitutes a single memory word.

Figure 2 shows a bus (apparently 424 bits wide) running from the memory to the output ports. The dynamic power dissipated by driving this bus, as well as the silicon area required to implement it, can be eliminated. By abutting a novel Parallel-to-Serial RAM (PSRAM) to the buffer output, as shown in Figure 5, we eliminate the bus area and ca-

1. Here, "priority" refers only to the encoding priority of the McCAM, and not to any QOS-related parameter; however, this priority can be varied (by rotation of the location of the highest-priority MCI) to achieve better fairness.

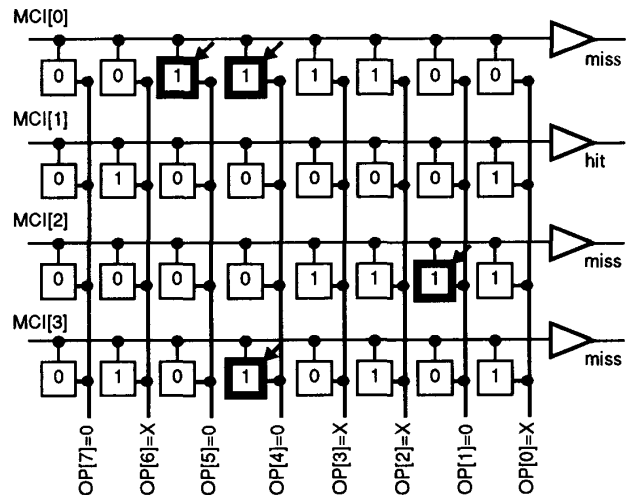


Figure 4: Multicast CAM. One of the four MCIs shown is eligible for reading. OP = Output Port.

Table 2: Sample schedule for output servicing of an 8x8 switch with a single multicast queue (mc=multicast, uni=unicast)

cycle →	0	1	2	3	4	5	6	7	8	9
slot 0	mc	uni1	uni2	uni3	uni4	uni5	uni6	uni7	mc	uni2
slot 1	uni0	mc	uni3	uni4	uni5	uni6	uni7	uni0	uni1	mc
slot 2	uni1	uni2	mc	uni5	uni6	uni7	uni0	uni1	uni2	uni3
slot 3	uni2	uni3	uni4	mc	uni7	uni0	uni1	uni2	uni3	uni4
slot 4	uni3	uni4	uni5	uni6	mc	uni1	uni2	uni3	uni4	uni5
slot 5	uni4	uni5	uni6	uni7	uni0	mc	uni3	uni4	uni5	uni6
slot 6	uni5	uni6	uni7	uni0	uni1	uni2	mc	uni5	uni6	uni7
slot 7	uni6	uni7	uni0	uni1	uni2	uni3	uni4	mc	uni7	uni0
slot 8	uni7	uni0	uni1	uni2	uni3	uni4	uni5	uni6	uni0	uni1

pacitance, and hence decrease the power, area and noise penalty associated with an extremely-wide-word memory. Although the PSRAM has an orthogonal structure, it is not an Orthogonal RAM [14], because more than one bit per word is read in each clock cycle.

Each output port actually has two 424-bit words. The first is connected to the bit lines that are driven by the buffer sense amplifiers, and it stores the buffer output when selected by the round-robin circuitry (more than one word may be selected in the case of a multicast). The second latches the cell from the first and has, as its output, a bus (of appropriate width) leading to that port's output pins. Figure 5 shows a 4-bit output. Because of the "double buffering", no commutation is required; Figure 6 demonstrates that bits 0-3 of all output ports are driven to the pads simultaneously.

Analogously, a serial-to-parallel function is required at the switch input. This may be implemented with a similar memory structure, abutted to the input of the shared buffer memory.

## VI. PHYSICAL IMPLEMENTATION ISSUES

The largest physical component of the switch chip is the buffer storage RAM. To achieve maximum density, it may be implemented as DRAM, rather than SRAM. At refresh rates approaching 10 ms, cells exceed QOS latency limits before their stored charge has leaked appreciably. Thus, no refresh mechanism is required, just a means of invalidating cells once they reach latency limits.

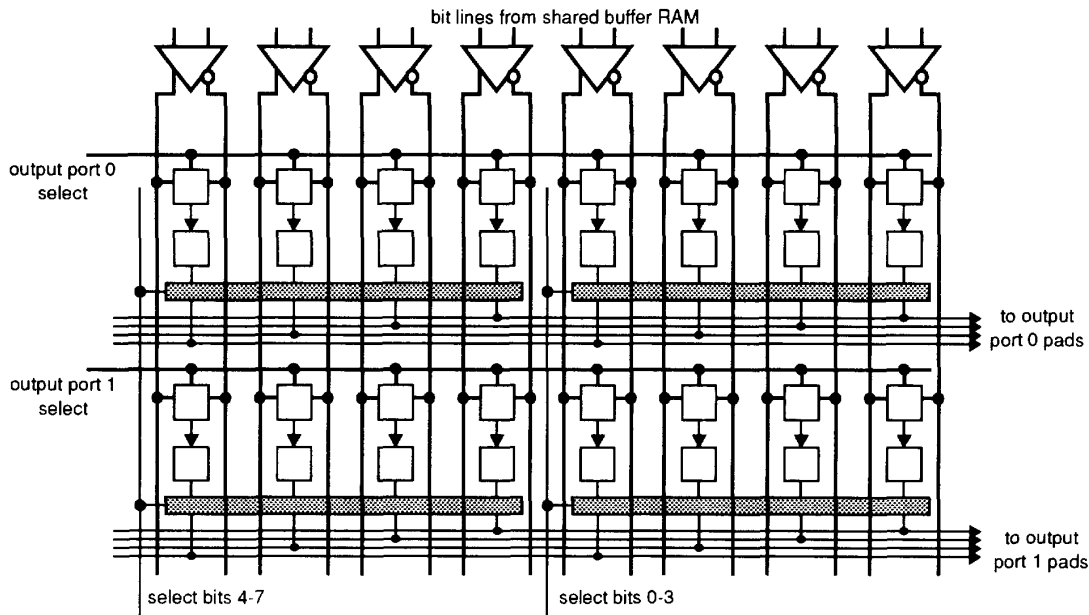


Figure 5: Parallel-to-serial RAM (2 port x 8-bit portion shown)

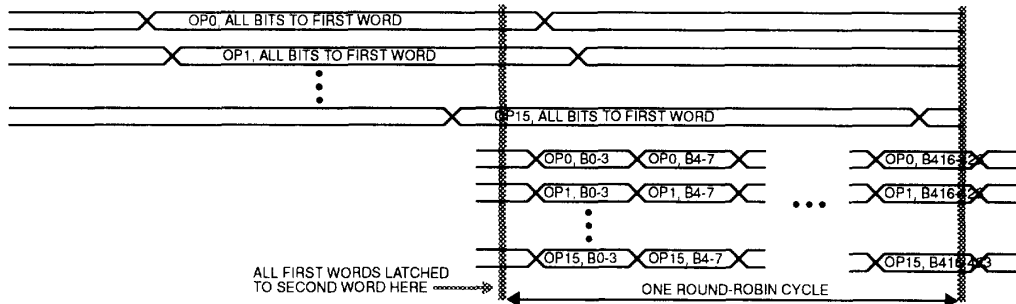


Figure 6: Output timing. OP = Output Port, B = bits.

Figure 7 shows a block diagram of the entire CAM-based single-chip shared buffer ATM switch, including multicast circuitry and PSRAM. If implemented in a full-custom BiCMOS technology [15], over 100 kb of dual port memory can fit on a single chip [16]; this is sufficient to implement the 16x16 switch considered in Table 1. Using data from [16], we can estimate an operating frequency of 100 MHz, and power dissipation of 1.4 W in the memory blocks. The shared buffer has an estimated bandwidth of 42.4 Gb/s, and this configuration would clearly have its bottleneck and major power consumption in chip I/O. A more realizable estimate of operating frequency would be 50 MHz, corresponding to memory power dissipation of 0.7 W, aggregate switch throughput of 21.2 Gb/s, and 256 ATM cell I/O pins at 166 Mb/s, dissipating approximately 5.3 W (CMOS levels, driving 20 pF loads). Identical performance is achievable using single port memories operating at 100 MHz, occupying less area. Table 3 summarizes these estimates.

Since the system bottleneck is shifted from the shared buffer to the chip I/O, increased throughput and significantly lower power dissipation could be achieved using optical chip I/O when this technology becomes available. In fact, the CAM-Access architecture seems ideally suited to this technology.

While the 256-cell capacity of the chip described above is sufficient for a proof of concept (and perhaps when embedded in a public net-

Table 3: Performance Estimates for 3 Implementations of CAM-Based Single-Chip Switch (0.8  $\mu$ m BiCMOS technology; 8 pins per port, 128 input pins total, 128 output pins total; 127,072 bits of memory total)

	Full-Speed Dual Port	Half-Speed Dual Port	Single Port
number of ports per memory	2	2	1
memory area (mm <sup>2</sup> )	53	53	34
memory cycle frequency (MHz)	100	50	100
memory power dissipation (W)	1.4	0.7	0.7
pin data rate (Mb/s)	331	166	166
pin power (20 pF loads, W)	10.6	5.3	5.3
aggregate throughput (Gb/s)	42.4	21.2	21.2

work) larger capacities are generally required to adequately handle the bursty characteristics of LAN switches. This certainly does not preclude the use of this single-chip architecture in such applications, since 16 Mb SRAMs have been fabricated in a research environment (see [17], for example), and will certainly be in production in a few years. If we wish to provide 1024 cells of storage per port rather than 16, approximately 64 x 128 kb of SRAM — or 8 Mb — will be needed on chip.

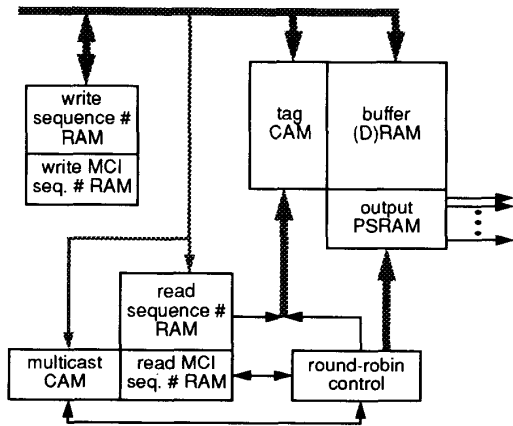


Figure 7: Complete CAM-based single-chip shared buffer ATM switch architecture.

As shown in Figure 1, the CAM-based shared buffer switch may be divided between multiple chips, with CAM and other full-custom memories on a single chip, and buffer RAMs on separate ICs.

## VII. SUMMARY

Technological leverage and design differentiation can still be achieved in the field of ATM switch hardware design by employing a full-custom architecture based on full-custom memory circuits. This paper has described an architecture in which CAMs are used to control access to the shared buffer, thus decreasing the need for address processing and manipulation, and simplifying the storage and retrieval of cells. The architecture requires fewer memory blocks, fewer total bits of memory, and less glue logic than a linked-list approach. An area- and power-efficient parallel-to-serial RAM is proposed as the interface between the shared buffer and output ports. The architecture easily supports cell loss priorities and latency priorities.

Table 4: Single-Chip Shared Buffer ATM Switch Comparison

ref	year	status	tech- nology ( $\mu\text{m}$ )	multi- cast	clock freq (MHz)	mem (kb)	power (W)	thru- put (Gb/s)
[4]	1991	fab&test	0.8 BiCMOS	No	100	134	6	3.2
[5]	1991	fab&test	1.0 CMOS	No	176	80+	3	1.4
[6]	1992	fab&test	1.2 CMOS	No	90	41	1.2	2.8
this	1994	proposed	0.8 BiCMOS	Yes	100	127	6	21.2

Table 4 compares the performance of published single-chip shared buffer ATM switches; the throughput advantage of the CAM-Access approach is clear. Due to the simplicity of the overhead processing surrounding the cell buffer, the buffer can run at the full chip clock rate. In the three other switches, the chip clock is divided down (by factors of 2, 8, and 2, respectively) to clock the buffer.

This is the first report of a single-chip shared buffer switch capable of supporting multicasting. The single-release (one-shot) approach is used to conserve shared buffer bandwidth. A secondary benefit of this mechanism is that, depending on traffic patterns, admissible switch node input bandwidth may not have to be scaled down from output bandwidth by the full fan-out factor. A potential disadvantage is that output bandwidth may be under-utilized when switch traffic is dominated by multicasts; in this case, a call-splitting mechanism [11,18] could be used in place of the one-shot approach. Simulations are required to determine the benefits of each mechanism under various traffic patterns.

A single-chip CAM-based switch is capable of aggregate throughput well over 10 Gb/s, and this chip should be affordable for ATM LANs, customer premises hubs, and other cost-sensitive applications. Further, the architecture offers several advantages as fabrication technologies scale. For instance, two or more cells per cycle could be handled by a denser switch chip, either by employing double-cell-width buffering, or parallel partially-independent circuits in a "super-scalar" approach [19]. As well, the architecture can be augmented by an asynchronous scheduling mechanism, for higher flexibility and efficiency [20]. The relatively small number of ports ( $16 \times 16$ ) should continue to be sufficient as networks become larger, since larger private networks will tend to be made up of a distributed arrangement of small low-cost high-density switches.

## ACKNOWLEDGEMENTS

This work was financially supported by the Natural Sciences and Engineering Research Council of Canada, Bell-Northern Research Ltd., the North American Life Assurance Company, and by a Walter Sumner Memorial Fellowship. Thanks to A. Chapman, E. Dormer, B. Hagglund, E. Munter, S. Nichols, and I. Perryman for their helpful comments. The suggestions of two anonymous reviewers are also appreciated.

## REFERENCES

- [1] A. Pattavina, "Nonblocking Architectures for ATM Switching", *IEEE Commun. Mag.*, pp. 38-48, February 1993.
- [2] M. G. Hluchyj and M. R. Karol, "Queueing in High-Performance Packet Switching", *IEEE J. Selected Areas Commun.*, vol. 6, no. 9, pp. 1587-1597, December 1988.
- [3] H. Kuwahara *et al.*, "A Shared Buffer Memory Switch for an ATM Exchange", *Proc. ICC*, pp. 118-122, 1989.
- [4] S. Tanaka *et al.*, "A 400Mb/s 8X8 BiCMOS ATM Switch LSI with 128kb On-Chip Shared Memory", *ISSCC Dig. Tech. Papers*, pp. 242-243, 1991.
- [5] W. Fischer *et al.*, "A Scalable ATM Switching System Architecture", *IEEE J. Selected Areas in Commun.*, vol. 9, no. 8, pp. 1299-1307, October 1991.
- [6] L. Licciardi *et al.*, "A Fully CMOS 622 Mbit/s Switching Element", *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 14.2.1-4, 1992.
- [7] H. Komiya, "Future Technological and Economic Prospects for VLSI", *ISSCC Dig. Tech. Papers*, pp. 16-19, 1993.
- [8] J. Yamada, "Reviews and Prospects of ASIC Memories", *IEICE Trans.*, vol. E74, no. 4, pp. 902-908, April 1991.
- [9] L. Chisvin and J. R. Duckworth, "Content-Addressable and Associative Memory: Alternatives to the Ubiquitous RAM", *IEEE Computer*, vol. 22, no. 7, pp. 51-64, July 1989.
- [10] K. J. Schultz and P. G. Gulak, "Architecture for Multi-Megabit Integrated CAM/RAM", *Proc. Canadian Conf. on VLSI*, pp. 6A1-6A7, 1993.
- [11] K. J. Schultz and P. G. Gulak, "Distributed Multicast Contention Resolution using Content Addressable FIFOs", *Proc. ICC*, paper 344.4, 1994.
- [12] Heald, R. A. and Holst, J. C., "6ns Cycle Cache Memory and Memory Management Unit", *ISSCC Dig. Tech. Papers*, pp. 88-89, 1993.
- [13] M. Akata *et al.*, "A Scheduling Content-Addressable Memory for ATM Space-Division Switch Control", *ISSCC Dig. Tech. Papers*, pp. 244-245, 1991.
- [14] A. Kokubu *et al.*, "Orthogonal Memory — A Step Toward Realization of Large Capacity Associative Memory", *VLSI 85*, E. Horst, Ed., Amsterdam: North-Holland, pp. 165-174, 1986.
- [15] R. Hadaway *et al.*, "A Sub-Micron BiCMOS Technology for Telecommunications", *Proc. European Solid-State Device Research Conf.*, pp. 513-516, 1991.
- [16] A. L. Silburt *et al.*, "A 180-MHz 0.8- $\mu\text{m}$  BiCMOS Modular Memory Family of DRAM and Multiport SRAM", *IEEE J. Solid-State Circuits*, vol. 28, no. 3, pp. 222-232, March 1993.
- [17] H. Goto *et al.*, "A 3.3-V 12-ns 16-Mb CMOS SRAM", *IEEE J. Solid-State Circuits*, vol. 27, no. 11, pp. 1490-1496, November 1992.
- [18] C.-K. Kim and T. T. Lee, "Call Scheduling Algorithms in a Multicast Switch", *IEEE Trans. Commun.*, vol. 40, no. 3, pp. 625-635, March 1992.
- [19] K. J. Schultz, "CAM-Based Circuits for ATM Switching Networks", Ph.D. Dissertation, University of Toronto, in preparation.
- [20] K. J. Schultz and P. G. Gulak, "Throttled-Buffer Asynchronous Switch for ATM", *IEICE Trans. Commun.*, vol. E77, no. 3, March 1994.