A Scalable Shared Buffer ATM Switch Architecture
A. Agrawal, A. Raju, S. Varadarajan, M. A. Bayoumi
Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504

Abstract
In this paper, a scalable shared buffer switch architecture for asynchronous transfer mode (ATM) with $O(\sqrt{N})$ complexity for memory bandwidth requirement and maximum crosspoint switch size, and $O(N)$ scalability for buffer memory size is proposed. Access time to buffer memories has been reduced by virtue of parallel access. The switch architecture features multiple buffer memories between the input and output side crosspoint switches. The new switch architecture is better than the standard shared buffer approach as it eliminates the use of input and output time division multiplexing and makes it possible to meet buffer memory access time limitations for larger switches. At the same time, the proposed switch architecture is able to keep the crosspoint switches from growing as $O(N^2)$ as is the case in the pure multibuffer architecture. The proposed architecture offers a good compromise between the simple shared buffer and shared multibuffer architectures. Architectural and implementation details will be discussed and a quantitative comparison between the buffer architectures will be given. Implementation of an 8 X 8 switch in 1.0um CMOS technology is described.

1.0 Introduction
High speed networking is an important prerequisite for making parallel and distributed computing systems ubiquitous and more usable. Data communication between computing systems which are operating in parallel at large geographical distances from each other require a wideband connection (tens of megahertz); but for mere milliseconds at a time. The asynchronous transfer mode (ATM) is emerging as the prominent technology in providing a widely acceptable common format for bursts of high-speed data transfer and multimedia applications.

The asynchronous transfer mode (ATM) is one of the keystones on which Broadband Integrated Services Digital Network (B-ISDN) is going to be based. According to CCITT recommendations, the operating speed of an ATM network will be either 155.52 Mb/s or 622.08 Mb/s. This high speed of bit traffic (6.4ns and 1.6ns respectively) means that switching will necessarily have to be done using high-speed self-routing architectures. Buffer memories have to be employed to prevent cells from being lost from being lost since at any time more than one input could arrive destined for the same output. Recently, many ATM switch architectures have been proposed which include input buffer switches, output buffer switches [1], shared buffer switches [2] and buffered Banyan switches [3,4]. It is known that shared buffer architecture ATM switch chips have lesser cell loss probability than self-routing banyan networks for bursty traffic conditions [3,6]. Amongst the buffer architectures, output buffer architectures offer the advantage of simple control. A shared buffer architecture, on the other hand, shares buffer memory among output ports and achieves better buffer utilization efficiency and has better cell loss ratio than that of output buffer switches having the same memory capacity. However, memory bandwidth requirements for shared buffer architectures become prohibitive for larger switch sizes. This tendency makes the development of larger high-speed ATM switches difficult.

The shared buffer switch architecture is shown in fig. 1. The bandwidth required for the buffer memory here is 2NR, where R is the rate at which data bits are being received by the switch, and N is the number of input/output ports.

![Fig. 1. Simple Shared Buffer ATM Switch](image1)

![Fig. 1b. Shared Multibuffer ATM Switch](image2)

Scalability of switch sizes keeping buffer memory access times constant, can be partly achieved by using a shared...
multibuffer architecture [7]. However, this architecture uses \( N \times N \) crosspoint switches to connect inputs and outputs to the buffer memory. As the size of the switch increases, these crosspoint switches prove to be a bottleneck and make the architecture less scalable.

In this paper, a scalable shared multibuffer ATM switch architecture with \( O(n) \) scalability in buffer access time requirements, as well as maximum crossbar switch size is proposed. For constant cell loss probability, buffer size is scalable as \( O(n) \).

Section 2 describes the concept behind the architecture. Section 3 discusses the VLSI implementation of the switch. A comparative analysis with respect to comparable switch architectures is provided in section 4 followed by conclusion in section 5.

2.0 Switch Architecture

In this section, the concept behind the scalable shared buffer switch architecture is described and control issues are discussed.

2.1 Scalable Shared Multibuffer Architecture

Figure 2 illustrates the concept of the Scalable Shared Buffer Architecture for a 16 x 16 switch. The architecture features \( \sqrt{N} \times \sqrt{N} \times \sqrt{N} \) crosspoint switches. For \( N=16 \), four 4x4 input-side crosspoint switches are needed. Each 4x4 switch has 4 inputs. The outputs of the 4x4 switches go to serial-to-parallel converters associated with memory modules 1 through 4. Each S/P converter operates in parallel storing bits from the incoming bit stream. When the serial-to-parallel conversion is complete, 4 S/P converters associated with each buffer memory write into the buffer in one write operation. The buffer RAM is organized in such a way that the 4 S/P converters can access 4 buffer memory words for simultaneous write. Buffer read operations to send out ATM packets are better done by 4 different memory read operations to the 4 PIS converters associated with each buffer. The output side crosspoint switches are required to switch the PIS converters' outputs to the correct output port. Thus, the required bandwidth of each buffer memory is \( (\sqrt{N} + 1) R \) for a single ported RAM. It is clear that memory bandwidth requirement scales as \( O(\sqrt{N}) \) switch size increases compared to \( O(N) \) access time requirement for simple shared buffer architectures (table 1).

The architecture is modularly extendable to 32X32 and 64X64 switches. The proposed architecture has its origins in the concept of set associative and sector mapping caches [8].

2.2 Crosspoint Switching

Modularity of the crosspoint switches is a significant advantage of this architecture over the simple shared multibuffer architecture. Instead of using a 16X16 crosspoint switch as would have been required in a pure shared multibuffer (fig 1b), architecture [7], this architecture uses four 4x4 switches thus keeping the crosspoint switching manageable. Fig. 3 shows the crosspoint switches the proposed architecture uses.

For the pure multibuffer architecture [7], as the switch size grows to 64 inputs, a 64X64 crosspoint switch will tend to be very large. Even a multistage 64X64 switch will require \( \log_2(N) \) stages and \( (N/2) \log_2(N) \) switching elements. Moreover, the switch control will require \( (N/2) \log_2(N) \) control signals, leading to a very complicated control for the switch. If the control is implemented as a lookup table (LUT), the number of control possibilities becomes factorial(64) which is difficult to implement. Therefore, it is advantageous to keep the switch size small.

This architecture manages to accomplish this by limiting the size of the switch to \( O(\sqrt{N}) \) where \( N=\)number of inputs/outputs being switched.

---

Fig 2. Scalable Shared Buffer Switch Architecture

Fig 3. 4X4 Benes Switching Network
2.3 Buffer Memory Management

The input and output side crosspoint switches must be properly controlled to realize the low cell-loss ratio of a shared buffer architecture. The input side crosspoint switch is controlled to keep the utilization of each of the buffer memories nearly the same. The incoming ATM cells are switched to the buffer memories in order of vacancy of the buffers. Thus, the emptiest buffers are given highest priority during the write operation. The control logic keeps track of the status of each buffer by using control queues which are decremented every time an ATM cell is written out into a buffer. FIFO output queues keep track of the destination port address of each cell written into the buffer, and maintain cell sequence integrity (fig. 4). There is no loss of cell sequence in this architecture as the cells are sent out in the order in which they arrived.

2.4 Switch Control

As stated earlier, control of the crosspoint switches to achieve uniform utilization of buffer memories is essential to obtain low cell loss ratio. How this control is performed is outlined next.

The idle address counters are responsible to keep track of the utilization of each buffer memory. Depending on the state of vacancy of each buffer memory, a priority re-ordering block computes the prioritized list of buffer memories in order of vacancy. Thus, least filled buffer memories are given highest priority and the next ATM cell that arrives will preferably be routed to that buffer.

Typical traffic estimates of the network for BISDN put the network utilization as an average load of 0.8. If connection admission control criteria is tightened, to the detriment of average network utilization, even busier networks can be slowed down to this traffic level. Due to the above fact, not all inputs will have data coming on them at all times. Therefore, we need to control the routing of the input packets depending on how many of the input lines are busy. For example for a 4X4 switch, if input data is coming on only 2 lines, then these 2 valid lines need to be switched to the least populated buffer memories, as determined by the output of the priority re-ordering block (fig. 5). The control to accomplish this has been implemented for our 8X8 switch in the form of a PLA-ROM matrix (fig. 6).

2.5 Switch Fabric

The speed of operation of the individual switch LSI's can be reduced by using a bit interleaved architectural scheme (fig. 7). The input side aligner LSI splits each incoming byte of each ATM cell into 8 buffer LSI's. Thus, for a 622 Mb/s switching fabric (one LSI needs to operate at merely 82MHz), assuming a 53 byte ATM cell with 3 bytes/cell added for internal routing. This is very much achievable with current CMOS technology.

3.0 LSI Implementation

The scalable multibuffer switch has been implemented as a single chip for 8X8 switching. This section discusses VLSI implementation of the switch LSI. The aligner LSI
from the aligner LSI. Since 16-way bit interleaving is used, each incoming ATM packet of 56 bytes (53 byte ATM cell + 3 bytes routing information added by aligner LSI) is split into 16 parts. Therefore, for each ATM cell received by the fabric, 28 bits are sent to each switch LSI.

For a particular input, ATM packets start arriving at the switch LSI input during clock period t0. At clock t6, 7 bits have arrived and are latched into the S/P converter. In the second phase of t6, the 7 bit latched word is transferred into a shadow register. During t6 to t13, these 7 bits of the incoming packet have arrived. At clock t14, the entire packet has been written into the buffer.

Subsequently, the stored packet is routed to one of the outputs depending on the state of the output queue. The mechanism for routing packets to the output is dealt with in the next section. Finally, parallel-to-serial converters output the packet onto the transmission channel at the 16-bit interleaved ATM rate.

3.2 Datapath

The organization of the datapath is shown in fig. 9. For a buffer write operation, the 5-bit 2–1 mux selects the address written out by the idle address queue. Alternatively, for a buffer read operation, the address comes from the external FIFO. This 5-bit address is fed into the buffer configuration without control. The demultiplexer is used to control the P2S operations. Since depending on the state of the buffer, the number of packets streamed out from the buffer during an ATM cell period can be anywhere from 0 to 4, the P2S converters need to be controlled carefully depending on the buffer read-out requirements.

The S/P converters feed the ATM bits into the buffer memory. The 5-bit address provided to the block is decoded using a 5 to 32 decoder. A 28-bit 1 to 4 demux writes to one of the P/S converters depending on the output destination of the ATM cell which is being read out.

The entire chip can store a total of 128 ATM cells (every sixteen bit is stored in one LSI). This results in 16 cells/input. Simulation results [4] show that 16 ATM cells buffered per input gives a cell loss probability (CLP) of less than $10^{-10}$ for Poisson distributed cell arrivals. Lower cell loss probability is one of the main advantages of shared buffer architectures over self-routing architectures like Batcher-Banyan switches.

Moreover, since each buffer needs to produce only 2 outputs, we could have had just 2 P/S converters. But this would have reduced the throughput considerably (to about 60%) due to the Head-of-Line effect. By using 2 sets of P/S converters, a total of 4 P/S converters, we can improve the throughput to about 90%. Further increases in the number of P/S converters yields diminishing returns.

3.3 4X4 Benes Switch

The 4X4 switching element required for the switch
has been implemented as a 3-stage Benes network (fig. 3). Each component switch is a 2X2 switch consisting of 4 transmission gates. The switch inputs are controlled by the PLA as mentioned in section 3.7.

When cells are written into specific memory addresses, the write addresses are multiplexed in through the input side 5-bit 2 to 1 mux, and the relevant bit is addressed via the 5 to 32 decoder. This bit is then reset to indicate that this memory location is full.

When an ATM cell is streaming into the S/P converters, addresses for writing the incoming ATM cells need to be generated. This is done by the 32 to 5 priority encoder. The priority encoder outputs the 5-bit address of the highest memory location having a vacancy. This 5-bit address is latched into the 5-bit register and the address is output.

This block provides an elegant way to generate the next available address. There is no assumption as to the order in which memory locations are freed, and there is no need to maintain any pointers to the next available memory address.

### 3.5 Priority Calculator

The priority calculator provides control inputs to the PLA to correctly switch the inputs to the requested buffers. The input mask latches in the input bits at the start of each packet. By our routing convention, if this bit is 1, then a valid ATM cell is arriving on that line. If the bit is 0, then no data is coming in on that line. The aligner LSI, which adds a 3-byte header to the 53-byte ATM cell, uses the first 2 bytes to provide this input active information.

Thus, depending on how many input lines are busy, the control logic generates sum, carry, and carry 2 bits (S, C, C2). For example, if 3 input lines are busy, S=1, C=1, C2=0 is generated. The priority activator circuit then decodes this information to enable the 3 highest priority decoders.

The inputs to these decoders come from the idle address counter (fig. 5). The top lines for example, provide the number of the most vacant memory. If the memory vacancies in decreasing order are (say) 2, 1, 3, 0, then A7A6...A0 will be 10011100.

The outputs of this block are control inputs to the PLA.

### 3.6 Idle Address Counter

The idle address counter (fig. 10) is an elegant realization of a cascaded ordering circuit. Depending on the states of the 5-bit up-down counters, the output of this block orders them in order of greatest-to-least value. Each counter represents one buffer and stores the number of vacant slots available in the buffer. The counter for a buffer is down-counter every time an ATM cell is written into the buffer, and up-counted every time the buffer is read.

### 3.7 Switch Control PLA

The switch control PLA generates the control inputs needed by the 4X4 Benes switches to correctly route the
inputs. For the previous example, when the inputs are coming in on switch inputs I0 and I2 and the priority order for large switch sizes. The shared multibuffer while having feasible memory bandwidth needs larger and more complex crosspoint switches for larger N. The scalable shared buffer switch proposed, on the other hand manages to keep both the memory bandwidth requirements and crosspoint switch requirements under control even as the switch size increases.

![Diagram of Idle Address Counter]

Fig. 10. Idle Address Counter

for the buffers is BUF2, BUF1, BUF3, BUF0 (BUF2 is the highest priority), the 2 possibilities for switching are:
1. route I0 to BUF2 and I2 to BUF1; I1 and I3 are don’t care
2. route I0 to BUF1 and I2 to BUF2; I1 and I3 are don’t care

Since it is immaterial whether the first of the second approach is used, we choose one option to implement in the PLA. In our case, we have chosen the first one and the corresponding switch control signals are (fig. 3) S1S2S3S4S5 = 01110. This PLA output goes to control the switch.

The claim we make above that it is immaterial whether we chose the first or second approach is only true locally. In other words, if we consider the final destination of the inputs on I0 and I2, one possibility may turn out to be better than the other and we could choose that possibility. For example, consider the case of I0’s final destination being output O5 and I2’s final destination being O6. If BUF1 has a long queue of cells waiting to go to output O4 (say), and input I2 is time critical, it may be better to use alternative 2 and send I2 to BUF2. This way, the packet has a higher probability of being read out earlier, and not getting blocked due to the head-of-line effect (HOL).

4.0 Comparative Analysis

Table 1 compares two important parameters for implementing different shared buffer ATM switches:
1. memory bandwidth required
2. switch size required

It is clear from the above comparison that the shared buffer architecture bandwidth requirement increases rapidly

<table>
<thead>
<tr>
<th>SWITCH</th>
<th>Memory Bandwidth Required</th>
<th>Max. size of crosspoint switch</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 x 4</td>
<td>4R 4R 4R SR 4x4</td>
<td>4 x 4 4x4 4x4</td>
</tr>
<tr>
<td>8 x 8</td>
<td>8R 8R 8R SR 8x4</td>
<td>8 x 8 8x4 8x4</td>
</tr>
<tr>
<td>16 x 16</td>
<td>16R 16R 16R SR 16x4</td>
<td>16 x 16 16x4 16x4</td>
</tr>
<tr>
<td>32 x 32</td>
<td>32R 32R 32R SR 32x4</td>
<td>32 x 32 32x4 32x4</td>
</tr>
<tr>
<td>64 x 64</td>
<td>64R 64R 64R SR 64x4</td>
<td>64 x 64 64x4 64x4</td>
</tr>
</tbody>
</table>

Table 1. Comparison of Buffer Architectures

5.0 Conclusion

A scalable shared buffer switch architecture is described in this paper which maintains low cell-loss ratio. The switch is scalable to handle larger number of inputs at the same time maintaining good buffer-memory utilization ratio. The suggested architecture offers a good compromise between the simple shared buffer and shared multibuffer architectures.

References