ATM Shared-Memory Switching Architectures

ATM will provide flexibility in bandwidth allocation and will allow a network to carry heterogeneous services ranging from narrowband to wideband services. The challenge is to build fast packet switches able to match the high speeds of the input links and the high performance requirements imposed.

Joan Garcia-Haro and Andrzej Jajszczyk

JOAN GARCIA-HARO is an assistant professor at the Polytechnic University of Catalonia.

ANDRZEJ JAJSZCZYK is a professor and the head of the Telecommunications Switching and Networks Group at the Franco-Polish School of New Information and Communication Technologies (EFP).

The CCITT has standardized the asynchronous transfer mode (ATM) as the multiplexing and switching principle for the Broadband Integrated Services Digital Network (B-ISDN). ATM is a packet and connection-oriented transfer mode based on statistical time division multiplexing techniques. The information flow is organized in fixed-size packets called cells, consisting of a user information field (48 octets) and a header (5 octets). The primary use of the header tag is to identify cells belonging to the same virtual channel and to make routing possible. Cell sequence on a virtual channel is preserved, a very low cell loss probability must be guaranteed ($< 10^{-12}$), and intensive error and flow control protocols are provided at the edges of the network. The line speeds are specified with nominal rates of 155.52 Mb/s and 622.08 Mb/s [1].

ATM will provide flexibility in bandwidth allocation and will allow a network to carry heterogeneous services ranging from narrowband to wideband services requiring real time. However, the challenge is to build fast packet switches able to match the high speeds of the input links and the high performance requirements imposed.

ATM switches can be broadly classified into two main categories, namely internally blocking and internally nonblocking. In an internally nonblocking switch two or more packets at different input ports can be simultaneously forwarded to the requested output port provided that they have distinct output port addresses. A switch is classified as an internally blocking switch if two or more packets with distinct output port destinations cannot always be transferred to the output ports due to contention within the switching fabric. For instance, resource contentions occur in the switching fabric when more than one packet has to access the same internal link.

Recently, a large number of switching architectures has been proposed [2]. All the approaches point to the need of a very high speed hardware switch because of the high transfer rates involved; on the other hand, due to the statistical multiplexing, buffering is also required in order to avoid packet loss whenever there are multiple input packets arriving simultaneously on different input ports and destined for the same output. Only one packet at a time can be transmitted over an output link, the rest must be temporarily stored in a buffer for later transmission.

The proper placement and arrangement of the buffering system have a dramatic impact on the switch performance. Assuming only nonblocking structures, if dedicated buffers for each input port are considered and they operate in first-in first-out (FIFO) fashion, the implementation is very simple in the sense that the switch has to operate at the same rate as the input/output lines and not many times faster. Therefore, an internal speedup of the switch is not required and the hardware complexity can be lower than in other buffering approaches. However, when a cell at the head of the line waits for transmission to the destined output port, successive cells in the queue must also wait. Consequently, the throughput is reduced if output contention arises [3].

In a nonblocking switch with individual output buffers for each output, an output buffer must be able to receive up to N cells at a time, where N is the switch size. Therefore, the switch does not suffer from the above described head-of-line (HOL) blocking effect and hence there is no throughput/delay degradation. However, the main drawback of the output buffered switch is that it needs to operate N times faster than the input/output line speed, increasing the implementation complexity [3]. There are also switches that combine input and output buffering techniques, and in this case, the speed of operation of the switching fabric can be lower than in the case of the output buffered switch [4].

Finally, another possibility to implement output buffering and therefore to attain the best throughput/delay/cell loss performance, is using a shared memory [3]. In shared-memory type switches that operate without blocking, all input and output ports have access to a shared-memory module able to write up to N incoming cells and to read out N outgoing cells in a switching time cycle, so that, as in output-buffered switches, throughput is not reduced by output port contention, and an optimal throughput/delay performance is achieved. Furthermore, to provide a specified level of service (i.e., cell loss rate) a smaller number of buffers is required (in comparison with the amount of memory needed by other buffering schemes) for all traffic patterns.

A shared-memory switch also has some additional features, e.g., its basic architecture can be easily modified to handle several service classes through priority control functions to meet different service requirements, multicasting and broadcasting can be also easily implemented in contrast to other types of architectures. Shared-memory type switches are, therefore, very attractive for the industry and the majority of implemented prototypes use them as the building blocks for larger switching fabrics [5].

The main reasons are:

- The previous experience in this kind of devices acting as space division modules in conventional packet switches, or as time-division switching stages in the current circuit switching environment, where the principal component of the packet switch is a random access memory (RAM).
- As single-stage switches, they not only perform better, but also use less hardware, have smaller size, are suitable for VLSI implementation, and have the highest efficiency in hardware utilization because the two required functions of every packet switch, queueing and switching are carried out via buffering.

It is evident that a single stage common-memory switch provides the lowest delay and the highest link utilization and, therefore, the prime interest is to build a single-stage switch with the maximum achievable size, constrained only by technological and physical limits. For instance, input/output pin limitations and power dissipation at higher frequencies are some of the major technological constraints for implementation.

In shared-memory based approaches the limitations on the size come from the memory control logic (which must be able to handle N incoming and N outgoing packets in each time slot), and the memory bandwidth that must be at least the sum of the bandwidths of the incoming and the outgoing lines. The memory bandwidth depends on the word length, which in turn is limited to the cell size (53 octets = 424 b), therefore for a given memory cycle time (or memory access time) the number of links N must fulfill the following relation (for single ported memories):

$$\text{cycle time} = \frac{\text{cell length}(b)}{2 \cdot N \cdot \text{link speed}(b/s)}$$

showing a clear limitation in the maximum achievable size of a shared-memory type switch.

Therefore, we have to distinguish single-stage and multistage approaches. In single-stage structures, cells are switched in a single phase providing the best throughput/delay performance. To construct larger switches, multistage arrangements may be considered. The first consequence of such an approach is the loss of a complete sharing of the overall buffering capacity, and furthermore, switching is performed through several phases and information packets cross multiple switching elements arranged in consecutive stages, creating several stages of queueing delays that may result in severe performance degradation if congestion occurs at intermediate modules. Even if the interconnection of these basic packet switches is made according to an internally nonblocking pattern, output contention is still possible resulting in nonoptimal delay/throughput performance. In this article, we review the most representative single-stage switches based on the shared-memory principle. Such switches are obvious candidates to be the main building blocks for the ATM switches. We also briefly survey larger switching fabrics that use common-memory switches as switching elements as well as we present related growability issues.

▪ ▪ ▪ ▪ ▪

Shared-memory type switches are very attractive for the industry, and the majority of implemented prototypes use them as the building blocks for larger switching fabrics.

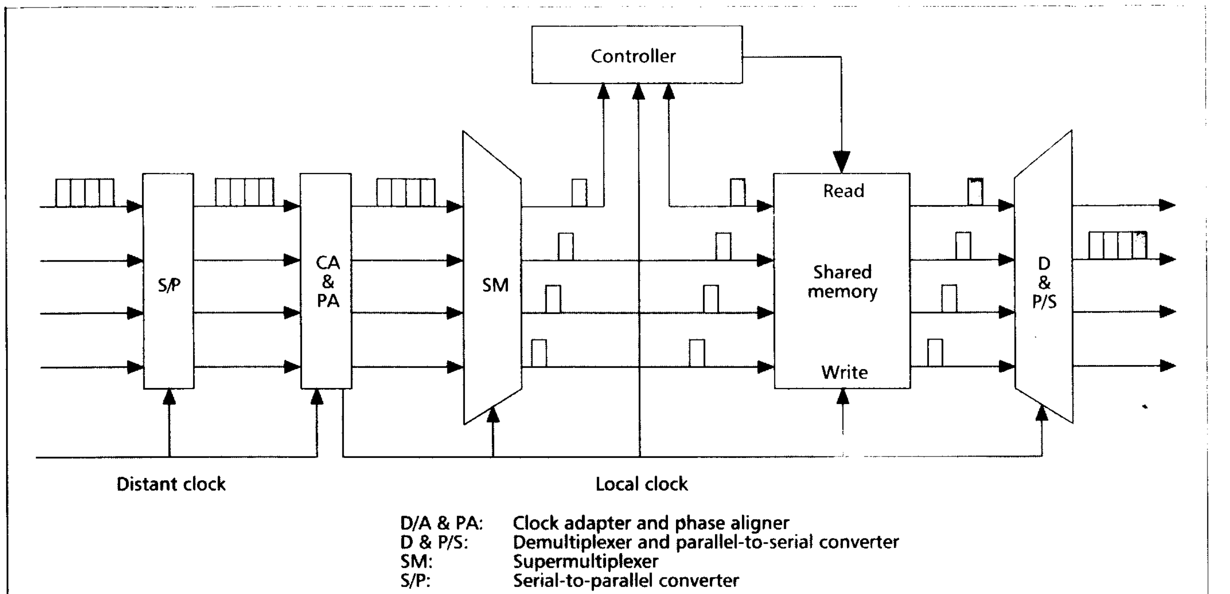
Single-Stage Shared-Memory Type Switches

The Prelude Switch

The Prelude was the first shared-memory type switch architecture using the ATM-like concept, described in the literature [6]. Because it was the pioneering shared-memory ATM switch and since some features of this architecture are still used in various switches today we will start with its description. The Prelude switch is a modified conventional time-division switch. In this prototype, cells consist of 16 octets (1-octet header and 15 octets for the user information field). The switch size N must be equal to the number of octets in a cell, so it is a nonblocking 16 inputs x 16 outputs (at 280 Mb/s) switch element. The switch operates synchronously with the time cycle equal to the duration of an octet. The switch can be divided into parts performing the following functions (see Fig. 1): serial/parallel conversion, clock adaptation and phase alignment, supermultiplexing, control, buffering, demultiplexing and parallel/serial conversion.

After a serial to parallel conversion of the incoming cells (to reduce the required physical memory speed), cells join the clock adaptation queues from where they will be extracted and aligned in such a manner that they present a shift of one octet, so the headers on different inlets arrive at consecutive time slots.

The supermultiplexing function is performed by a 16 x 16 space-division switch that sets its internal paths every time cycle according to a cyclic modulo N algorithm, that is, in state 1, input i is connected to output i, in the following state input i is connected to output $(i + 1)$ mod N, an so on, where N is the switch size. As the result, all the headers of different input packets are multiplexed on the first output of the supermultiplexing stage, all the first user information octets on the second output, etc. After this so-called parallel-diagonal transformation, header octets are successively processed and translated by a central controller to define the switch action. The cells are stored diagonally in the buffer memory which is organized in N banks, each bank being dedicated to an octet of the cell, i.e., the header is stored in an empty location of bank 1, and the remaining octets of the same packet are stored in consecutive banks. The address register is increased by one for each successive octet. The address where the packet header is placed is


Controller
Read
Shared memory
Write
S/P
CA & PA
SM
D & P/S
Distant clock
Local clock
D/A & PA: Clock adapter and phase aligner
D & P/S: Demultiplexer and parallel-to-serial converter
SM: Supermultiplexer
S/P: Serial-to-parallel converter

■ **Figure 1.** *Prelude switch.*

stored by the controller in a dedicated queue for the destined output port.

To read out the memory packets to the output links, the controller, at each time cycle, delivers address A in the first bank where a cell header of a packet to be extracted is stored, the reminder of the packet is easily retrieved from the other banks during the following time cycles incrementing A. Retrieved packets are fed to a rotative space-division switch which acts in a reversed order as in the supermultiplexing stage, reconstructing the packets. Finally, the packets are converted from parallel to serial form and are transmitted to the corresponding output lines.

The output control queues have fixed size, which means that the memory is not completely shared. However, since all input and output lines have access to the shared memory it would not be difficult to extend this design to achieve a higher degree of sharing.

Linked-List-Based Shared Memory Switch

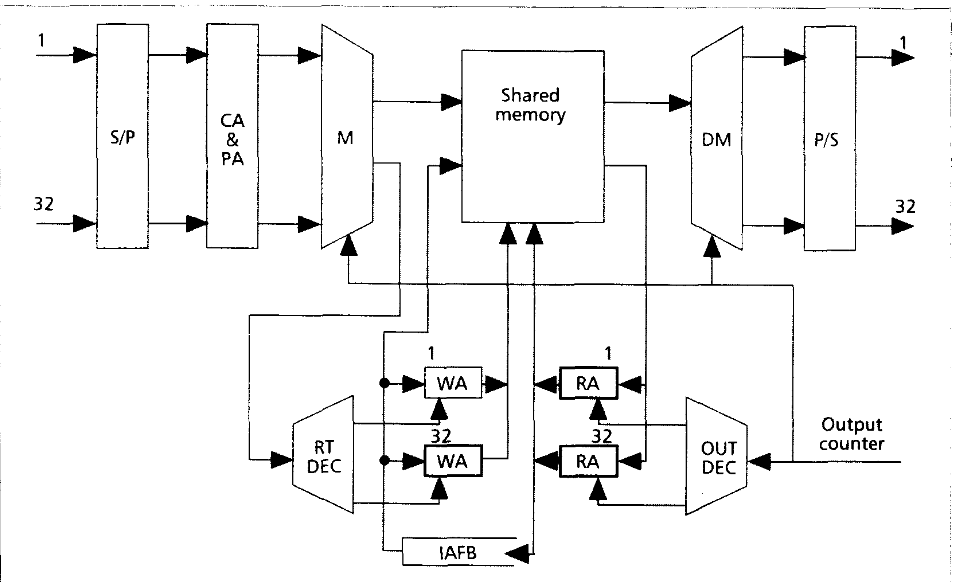
Another group of shared memory switches employs a linked list to logically organize the buffering system. We present in this paper the ATM switch described by Kuwahara *et al.*, as an representative example [7]. In this switch the buffer memories for output queueing are completely shared by all the switch output ports and can be assigned to any output port as required by traffic conditions.

The switch architecture is illustrated by Fig. 2, and its basic operation is as follows. First, cells are converted from serial to parallel format, the cell headers are analyzed and appropriately translated by header converters. Subsequently, cells from all input ports are multiplexed and written into the shared buffer memory. The buffer memory is logically organized as N ($N = 32$) linked lists, one for each output line. A linked list is a set of chained memory locations that indicate the place where successive cells for a particular output are stored, maintaining, in that way, the cell sequence. A pair of address registers (one to write WA and one to read RA) control the proper operations on the shared memory. Write register i indicates the end of the chain and so designates an empty memory location where the last cell that arrives for output i can be stored. Read register i indicates the beginning of the list, and therefore, the location of the first cell that can be delivered to output port i. Both the cells and the linked list structures are stored in the same buffer memory.

For each input packet, the appropriate WA register (designated by the route decoder RT DEC) is accessed to get the memory location for that packet. Simultaneously, a new address of an empty buffer is read from the idle address FIFO buffer (IAFB), which keeps a pool of empty buffer locations, to update the content of the WA register. Analogously, at each switching cycle a packet from each linked list is identified through the content of RA registers, retrieved, demultiplexed and transmitted. At the same time, the contents of the RA registers rejoin the idle address buffer and are replaced by the next chain addresses obtained from the same addresses as the output cells, thus updating the pointers.

The basic design can be easily modified to accommodate different service classes organizing the packets addressed to a given output port into multiple linked lists, one for each service class and serving these lists according to a prioritized scheme. This requires the addition of a pair of WA and RA registers to the control block for each service class and each output port.

The multicast function requires the addition of a multicast circuit [8]. Multicast and broadcast cells form a broadcast queue in the shared-memory, irrespective of their destinations. This queue has the same structure as all other output queues and is handled by WA and RA registers. Before a multicast cell is sent out from the switch, its header is analyzed to define the output port numbers



DM:	Demultiplexer
HC:	Header converters
IAFB:	Idle address FIFO buffer
M:	Multiplexer
OUT DEC:	Output decoder
P/S:	Parallel-to-serial converters
RA:	Read address register
RT DEC:	Route decoder
S/P:	Serial-to-parallel converters
WA:	Write address register

■ **Figure 2.** *Linked-list-based shared mamory switch.*

In shared-memory-based approaches, limitations on the size come from the memory control logic and the memory bandwidth.

to which the cell is to be multicast and this information is stored in a multicast routing table. The routing information is used to deliver multiple copies of the same packet to the specified output ports.

A complete study on this switch approach, including all the abovementioned features as well as performance issues and LSI implementation details, can be found in [9].

Hybrid Shared and Dedicated Output Buffer Switch Approach

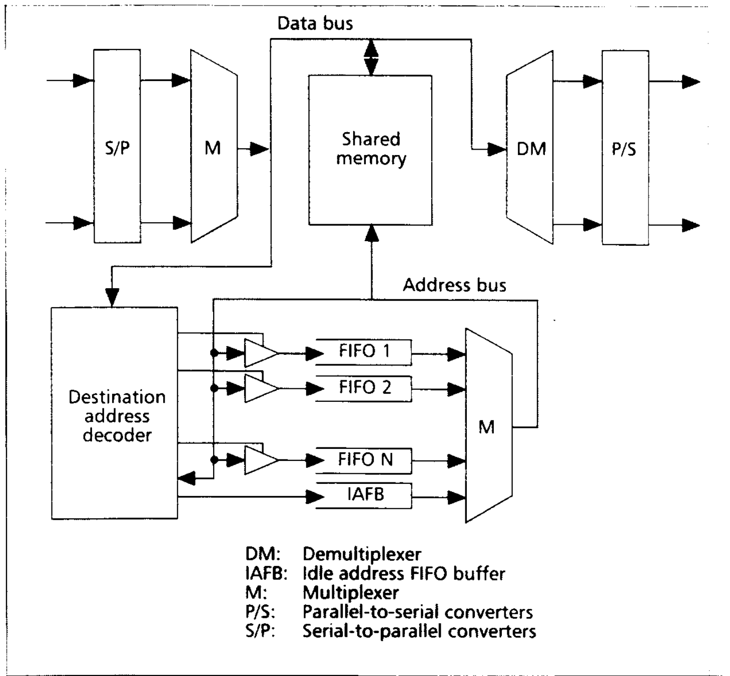
In this approach the control mechanism to route the cell in the shared memory to its output port uses *N* dedicated FIFO buffers, one for each output port [10]. Thus, instead of arranging cells which have the same output port destination in a linked list by the address chain pointer, their addresses are stored into a FIFO buffer which is dedicated to a specific output port.

The switch architecture is shown in Fig. 3. Serially incoming cells are converted to bit parallel data to reduce the required internal speed of the switch, sequentially multiplexed in time and stored in a shared buffer memory. Their respective buffer addresses are recorded on separate FIFO buffers associated with the specific destinations of every cell. A particular FIFO buffer is selected by examination and conversion of the routing header information of each cell. The write addresses to store the incoming cells into the memory pool are provided by an idle address FIFO buffer that holds all the empty buffer locations of the shared memory. The addresses in the FIFO buffers are used to read the cells out from the shared buffer memory, demultiplex and reconvert them to serial format, and finally to send them to their destinations. The new free addresses are returned to the idle address FIFO queue.

The performance of the switch is the same as that of the shared switch using linked lists, but the necessary amount of buffering is slightly higher due to the buffering required for the FIFO queues. Nevertheless, the implementation is simple and the multicasting and priority control easy to implement.

The multicasting function requires an addition of a broadcasting service control circuit which keeps track of the number of destination output ports where the multicast cell is to be sent. To assure that the multicast cell stays in the shared buffer memory and its address is not recovered by the idle address FIFO buffer until the cell is sent to all the destinations, only the last addressed logical queue authorizes the release of the corresponding shared-buffer memory cell location.

If there is the need to discriminate cells to support services with different quality requirements, a priority control has to be implemented. This can be easily achieved by allocating several address FIFO queues to each output port. During the write phase, a cell is written to the shared buffer memory and its corresponding address is stored into a FIFO queue that matches the suitable cell priority. At the read phase, the FIFO queues associated with a particular output port are scanned according to their priorities.


Data bus
S/P
M
Shared memory
DM
P/S
Address bus
Destination address decoder
FIFO 1
FIFO 2
FIFO N
IAFB
M
DM: Demultiplexer
IAFB: Idle address FIFO buffer
M: Multiplexer
P/S: Parallel-to-serial converters
S/P: Serial-to-parallel converters

■ **Figure 3.** *Hybrid shared and dedicated output buffer switch.*

It is also necessary to indicate that in this architecture it is easy to shift from a complete sharing to a complete partitioning of the shared memory space since the sizes of the address FIFO buffers determine the number of cells which can be stored into the shared buffer memory with the same destination output port. This feature can be used to improve the performance by preventing each output port from overloading the shared buffer in the case of unbalanced traffic patterns that heavily load the shared buffer with cells destined to only some output ports [11].

Other Shared-Memory Switch Designs

Although well known shared-memory type switches have one of the architectures already described, we can note some interesting variations. Henrion *et al.* [12] describe an integrated switching element that operates as a matrix of 32 inlets by 32 outlets at 150 Mb/s. The switch also incorporates broadcast and selective multicast capabilities. The control of the shared buffer memory is accomplished by using a linked list technique. This switch can be applied in non-ATM environments due to its compatibility with variable packet lengths by using "multi-slot cells." In the multi-slot transfer mode, an external packet is mapped into a chain of elementary fixed length slots, called a multi-slot cell, having a single header plus a few bits to indicate the cell slot sequence, as well as idle slots. All slots of a single packet follow the same path through the switch. For each multi-slot cell, the routing logic analyzes the self-routing label and, depending on the specific routing mode (point-to-point or multicasting), routes the cell to a logical queue served by a group of 4, 8, 16, or 32 outlets. Such a multi-service discipline and internal transfer mode, reduce the required buffering capacity of the shared buffer memory.

Kitamura [13] proposed a switch in which input cells are not multiplexed and, therefore, a very fast shared buffer memory is not required. The switch has more than one buffer and a buffer stores a single cell, so that, cells are written or read independently. Consequently, it is not necessary to align the timing of the arrived cells at the inputs. Moreover, the write or read speeds in the cell buffer can be different. To achieve this, the multiplexer part, usual in shared-memory switches, is replaced here by a space-division switch matrix that provides fully interconnected paths between any input and any cell buffer. The matrix crosspoints are set up by an empty cell buffer select controller.

Each output port has access to all outputs of the cell buffers through a multiplexer called cell buffer selector. To identify which cell buffer will be selected to send its stored cell to the output port destination, a FIFO queue (one for each output port), corresponding to the destined output line, controls the state of its particular cell-buffer selector. In order to preserve the cell arrival sequence per output port, these queues contain the encoded cell buffer number where a cell addressed to a specific output port is placed.

In this kind of architecture, the number of input and output lines can be arbitrary, therefore an asymmetrical switch suited to line concentration can be easily implemented. The priority control and broadcasting capabilities can be also readily achieved. However, in this approach the hardware complexity increases as the product of the number of lines and the number of buffers, making the switch, for large sizes, difficult to implement.

Kitamura's proposal is further refined in two recent works [14, 15]. Oshima *et al.* describe an ATM switch design based on an Space-Time-Space (STS)-type shared buffering scheme. In this architecture (Fig. 4), the usual multiplexing and demultiplexing stages are replaced by crosspoint space switches, so that, the memory access speed can be relaxed.

Multiple buffer memories (able to hold more than one cell) are shared among all input/output ports via the crosspoint space switches. The input-side crosspoint switch connects each input port to one shared buffer memory (SBM) module. The output port destination of incoming ATM cells is inspected by header detectors at the input ports. This information is used by a write address control block to set the state of the input-side space switch, thus, connecting the input ports to the appropriate buffer memories.

As far as the number of SBMs is equal or greater than the number of input ports and each SBM is not full, ATM cells are stored into the SBMs in parallel, otherwise cell dropping occurs. To implement the shared buffering function efficiently, the cells are written with higher priority into the less occupied SBMs. The write addresses along with the corresponding buffer memory number are queued in the respective address FIFO queue that is individually provided for each output port. A read address control block selects the first address from each address FIFO queue and commands the output-side space switch to deliver the cells to the requested output line. The overall control block also maintains an idle addresses pool to store empty location addresses for reuse.

With this approach it may be the case that two or more cells with different output port destinations should be read from the same buffer module at the same time. This fact leads to contention and consequently, to the degradation of the system throughput. Two methods can be used to alleviate the effects produced by this type of contention: one is to increase the number of SBMs, the other is to accelerate the memory read out process. Both methods are evaluated in [14] and it is concluded that the speedup method gives better results. A speedup factor of 3 is chosen, therefore, an SBM has to be able to perform one write and three read operations in a cell time.

Finally, we will mention some other recent works as an example of the intense research and prototyping efforts that are being conducted in Europe, North America, and Japan to build the ATM switches and crossconnect systems of the future B-ISDN networks. A common factor in all of them, as it was highlighted before, is the use of shared-memory ATM switch architectures as the core blocks of larger switching fabrics.

In [16], 32 x 16 shared-memory ATM switching elements are taken as the building blocks arranged in a funnel-like structure. Eng *et al.* [17] describe a 2.5 Gb/s ATM switch prototype based on the Growable Switch Architecture [18] in which the main building block is an 8 x 8 shared-memory switching element. In [19] a 2 x 2 shared-buffer switch element is used to form higher size crosspoint-buffered matrix switches, called element modules, by placing the 2 x 2 elements at the crosspoints of the structure.

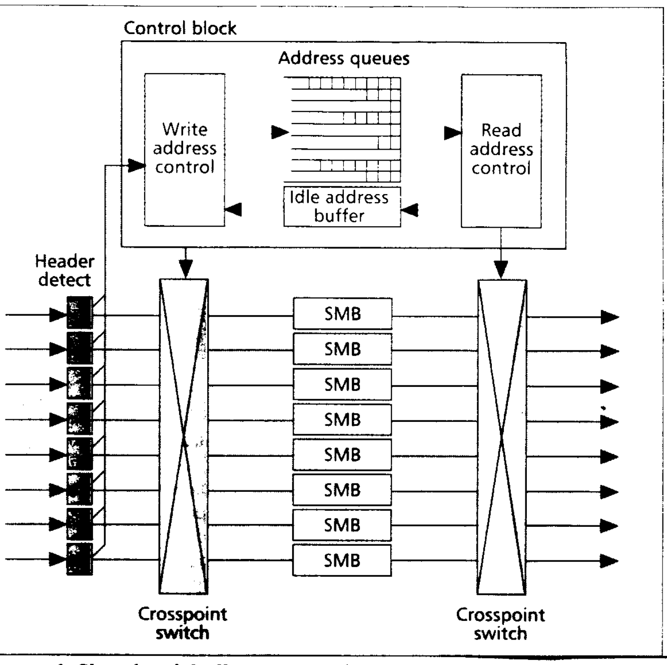
Kumar and Agrawal have recently proposed a shared buffer direct access ATM switch that parallelises the memory management and memory access operations, thus increasing the processing speed [20]. A linked list mechanism to control the shared memory has been replaced by a content addressable memory (CAM) in the switch described by Schultz and Gulak [21]. In this case, the CAM stores tags (containing output port and sequence numbers) used to reference the cells. Cells are accessed for reading by searching for the desired tag. Incoming cells are written into the first free location, the address of which is irrelevant. The discussed architecture requires fewer memory blocks, fewer total bits of memory, and less glue logic than in the case of the linked list approach. This, in turn, facilitates single-chip implementations.

Single-Stage Shared-Memory Type Fabrics

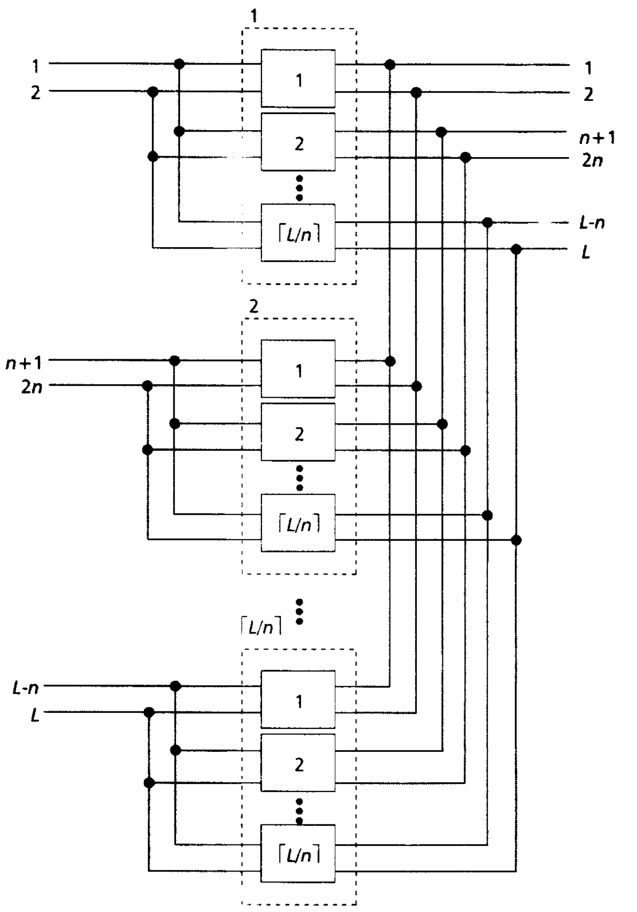
The maximum capacity of a single switch is limited by technology constraints. To obtain larger capacities more switches can be connected together. The conceptually simplest method is based on parallel connections between switches, as shown in Fig. 5. The number of individual switches in such an arrangement is $\lceil L/n \rceil^2$, where L is the total number of the incoming and outgoing ATM links, and n is the number of links per a single switch. The single-stage fabric has the following advantages [15]:

- Nonblocking properties.
- Suitable for high-speed interfaces because no intermodule speedup stage is required.
- Easy to change the virtual path capacity while keeping the virtual path connection.

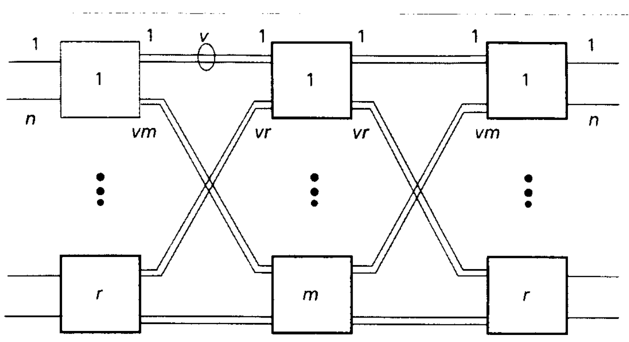
An important implementation issue is associated with branching points denoted by dots in Fig. 5. Such



■ **Figure 4.** *Shared multibuffer ATM switch.*



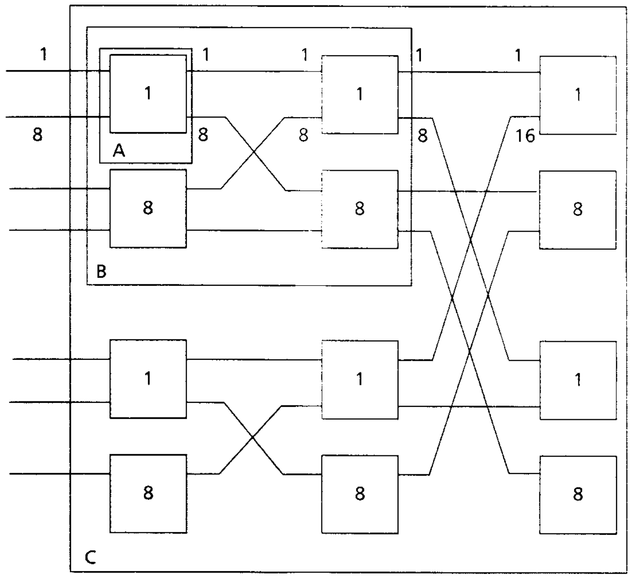
■ **Figure 5.** *Single-stage switching fabric.*



■ **Figure 6.** *Three-stage two-sided fabric.*

points, along with complicated wiring and high speed ATM signals, may lead to great technological difficulties due to electrical signal reflections. Even more important is the possibility of cell collisions at combining points. To prevent this, appropriate arbitration and buffering are required.

In some systems, simple passive branching gives satisfactory results [22, 23]. In this case, the switching elements should have the capability of accepting only calls with the appropriate values in self-routing headers. Another option is to apply demultiplexers in splitting points. Such demultiplexers can be controlled by bits in a cell header. More complex functions have to be performed by the multiplexers at combining points. These functions include the prevention of the internal congestion when two or more cells compete for the access to the same multiplexer output. In some systems a standard ATM switching element is used as a set of output multiplexers. In some others, arbiters are introduced to multiplex the output cells from switching elements to the same destination output ports [15]. We should note that the cost of branching points can considerably increase the overall cost of the fabric of Fig. 5.



■ **Figure 7.** *Growability of a folded switching fabric.*

Another single-stage approach has been proposed by Wei and Kumar [24]. In this approach, shared-memory switches are placed in parallel, with every input and output port having access to every one of the switches. The advantage of this approach is that it permits global sharing of the total buffer space. However, interconnection schemes between elementary switches are yet to be developed.

Multistage Fabrics

Structures

The rapid growth of the number of required switching elements in single-stage solutions as well as the problems related to their interconnections, lead to multistage switching fabrics. Most common structures contain two or three stages composed of shared-memory switches. An example of a three-stage fabric is shown in Fig. 6. To reduce the probability of blocking, the number of parallel lines *v* between switches located in consecutive stages can be increased. Another possibility is to increase the ratio *vm/n* or to use the speed up of internal links [13].

To simplify cell routing inside the three-stage fabric, first-stage and second-stage switches can be grouped in blocks [25], where some first-stage switches have access to the second stage only inside their blocks. The same technique can be used to switches of the second and the third stage. The use of additional overflow switches in each block can lower the cell loss probability [26].

For larger fabric capacities, in some cases, more than three stages are used. For example, the UT-XC ATM node employs a five-stage arrangement [27]. However, the multistage approach results in considerable cell delay as well as requires more complicated routing procedures. To overcome these difficulties, single switches in one or more stages can be replaced by single-stage modules of Fig. 5 [23].

Along with two-sided fabrics, such as shown in Fig. 6, one-sided (also referred to as folded) fabrics are used in practice. Folded fabrics allow interconnections between all its terminals, and therefore can be used for switching nodes where local connections are set up. Two-sided fabrics are rather used in transit nodes where cells are delivered from a set of incoming links to a separate set of outgoing links. Mixed one- and two-sided fabrics are also possible.

We can distinguish two basic one-sided architectures. The first is based on looping fabric input and output lines having the same numbers. The second solution uses one- sided switches and leads to so-called triangular fabrics, shown for example in Fig. 7. In triangular architectures some parts of the fabric transport information in one direction, and the same parts transport the information in the opposite direction [5].

Fabric Growability

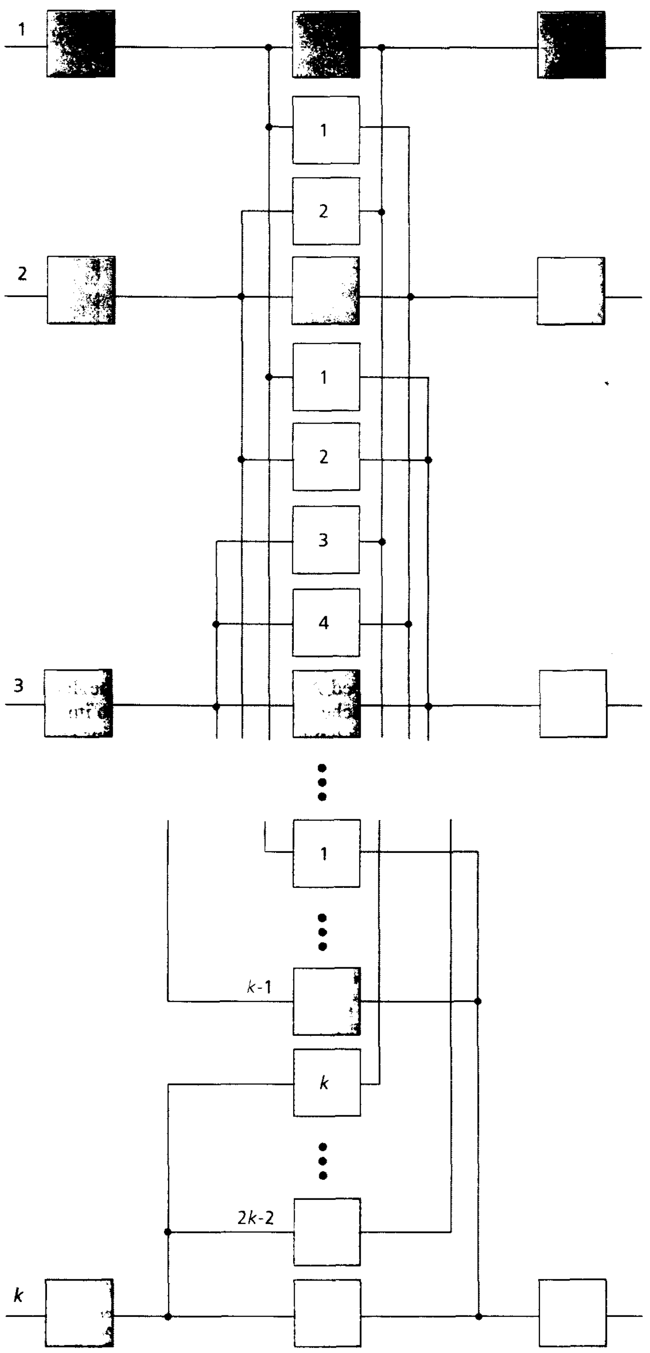
One of the important requirements for any ATM switching fabric is its growability. The fabric architecture should permit the modular growth of its size from a small number of ports to very large switch sizes. It is also desired that this growth does not result in a performance degradation as well as the reliability requirements are met at each stage of the expansion.

The concept of the size expansion in triangular fabrics is illustrated by Fig. 7. Let us assume that the basic ATM switching element has the capacity of 16 bidirectional links. If only a few ports are needed, the fabric consists of a single switching element (denoted by *A*). To achieve a higher capacity than 8 ports, the second stage should be added and the number of first-stage switching elements has to be increased (up to 8). Thus, we obtain fabric *B*, containing at most 64 ports. The required throughput is achieved by adding an appropriate number of second-stage switching elements. The fabric of the maximum size, denoted by *C*, is obtained analogously. If we assume that the ultimate fabric consists of at most three stages, we can use the total capacity (i.e., 16 ports) of the third-stage elements to connect the two-stage modules, without leaving some ports for the connections to the next stage. We can note that the presented method of expansion does not disturb already existing links.

Cells in folded fabrics described above are routed by different number of switching elements, depending on the addresses of input and output ports. For example, if a cell is to be sent between ports belonging to the same switching element, only this element is involved in the switching process. A communication between ports of the same two-stage module but different switching elements requires routing to the second stage where cells are reflected back to the first stage. The intermodule communication moves the reflection point to the third stage. In the latter case a cell traverses five switching elements between the input and the output port.

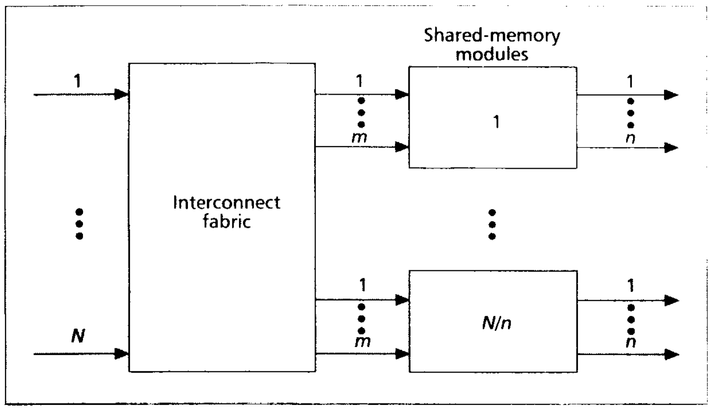
The concept described above is used in various ATM or fast packet architectures, although some of them use a different interconnection pattern between stages. For example, de Prycker *et al.* have proposed the *Athena* switching fabric that employs 16 x 16 switching elements [5], [28]. In the *Roxanne* switching fabric, described by Henrion *et al.*, 32 x 32 switching elements are used [5, 12]. These elements form standard switch module boards of 128 x 128 inlets/outlets of 150 Mb/s each. Similar, three-stage folded architectures have been also described by Obara and Okamoto [29], as well as Tanabe and Ohtsuki [30]. Fischer *et al.* discussed a two-stage folded ATM architecture using 16 x 8 shared-buffer switching elements [22]. They proposed a detailed procedure for the fabric growth which ensures that none of the existing ATM connections is disrupted. A folded architecture that makes it possible to expand the size of an ATM fabric in a smooth way, and preserving its number of switching stages, has been proposed in [31].

The expansion method for a two-sided fabric is shown in Fig. 8 [14]. To increase the number of lines twofold, a second identical fabric is added to the existing structure and both sub-fabrics are interconnected by additional modules. These modules have the same structure as the middle part of the original fabric. To increase the size of the fabric by additional *N* lines, the next *N* x *N* fabric is added and connected to the existing structure. In general, to expand the size of the fabric *k* times, *k*(*k* - 1) interconnecting modules are required. Note that each line in Fig. 8 represents a group of lines incoming to or outgoing from switching modules (which, in general, contain a multiplicity of elementary shared-memory switches).



■ **Figure 8.** *Growability of a two-sided switching fabric.*

A smooth growability and very good performance can be achieved in an architecture proposed by Eng, Karol, and Yeh, shown in Fig. 9 [18]. In this architecture the switch is composed of two sections: a memoryless interconnect fabric that has to provide connections between *N* inlets and *mN*/*n* outlets, and shared-buffer-based output packet modules. The fabric can grow by using more and more fixed and identical *m* x *n* output packet modules and adjusting the size of the interconnect fabric accordingly.



■ **Figure 9.** *Switch containing shared-memory modules and memoryless interconnect fabric.*

Conclusion

One of the most promising solutions of ATM switches is based on the shared-memory principle. Such switches play a leading role in practical, experimental implementations of ATM. Shared-memory switches have good traffic performance, although satisfactory analytical methods to predict such a performance are still to be developed. An attractive feature of the shared-memory switches is that they can be also implemented by using typical, on shelf, VLSI chips (high speed RAMs). Some important problems, including methods of switch design, selection of optimum fabric structure for point-to-point as well as multicast connections, growability limits, or development of appropriate control methods, are yet to be solved.

References

[1] CCITT Recommendation I.121. "Broadband Aspects of ISDN," Blue Book, Fascicle III.7, Geneva 1989.

[2] F. A. Tobagi, "Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks," Proc. of the IEEE, vol. 78, No. 1, Jan. 1990, pp. 133-167.

[3] M. G. Hluchyj and M. J. Karol, "Queueing in High-Performance Packet Switching," *IEEE J. Select. Areas Commun.*, vol. 6, Dec. 1988, pp. 1587-1597.

[4] Y. Oie *et al.*, "Effect of Speedup in Nonblocking Packet Switch," ICC '89 Conf. Rec., Boston, MA, June 1989, pp. 410-414.

[5] M. De Prycker, Asynchronous Transfer Mode: Solution for Broadband ISDN, Second Edition (Chichester, England: Ellis Horwood, 1993).

[6] J. P. Coudreuse and M. Servel, "Prelude: an Asynchronous Time-Division Switched Network," ICC '87 Conf. Rec., paper 22.2, Seattle, WA, June 1987.

[7] H. Kuwahara et al., "A Shared Buffer Memory Switch for an ATM Exchange," ICC '89 Conf. Rec., Boston, MA, June 1989, pp. 118-122.

[8] T. Kozaki *et al.*, "32 x 32 Shared Buffer Type ATM Switch VLSI's for B-ISDN's," *IEEE J. Select. Areas Commun.*, vol. 9, Oct. 1991, pp. 1239-1247.

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

[10] H. Lee *et al.*, "A Shared Output Buffer Switch for ATM," Fourth International Conf. on Data Communication Systems and Their Performance, Barcelona, June 1990, pp. 163-179.

[11] J. W. Causey and H. S. Kim, "Comparison of Buffer Allocation Schemes in ATM Switches: Complete Sharing, Partial Sharing, and Dedicated Allocation," ICC '94 Conf. Rec., New Orleans, LA, May 1994, pp. 1164-1168.

[12] M. A. Henrion *et al.*, "Switching Network Architecture for ATM Based Broadband Communications," Proc. ISS '90, Stockholm, Sweden, vol. V, May/June 1990, pp. 1-8.

[13] H. Kitamura, "A Study on Shared Buffer-Type ATM Switch," Electronics and Communications in Japan, Part 1, vol. 73, no. 11, 1990, pp. 58-64.

[14] K. Oshima *et al.*, "A New ATM Switch Architecture Based on STS-Type Shared Buffering and its LSI Implementation," in Proc. XIX ISS '92, Yokohama, Japan, Oct. 1992, pp. 359-363.

[15] H. Yamanaka *et al.*, "622Mb/s 8x8 Shared Multibuffer ATM Switch with Hierarchical Queueing and Multicast Functions," in GLOBECOM '93 Conf. Rec., Houston, TX, Nov./Dec. 1993, pp. 1488-1495.

[16] W. Fischer, R. Stiefel, and T. Worster, "An ATM System and Network Architecture in Field Trial," in GLOBECOM '93 Conf. Rec., Houston, TX, Nov./Dec. 1993, pp. 1476-1479.

[17] K. Y. Eng *et al.*, "A High- Performance Prototype 2.5 Gb/s ATM Switch for Broadband Applications," in GLOBECOM '92 Conf. Rec., Orlando, FL, Dec. 1992, pp. 111-117.

[18] K. Y. Eng, M. J. Karol, and Y.-S. Yeh, "A Growable Packet (ATM) Switch Architecture: Design Principles and Applications," *IEEE Trans. Commun.*, vol. 40, Feb. 1992, pp. 423-430.

[19] K. Yamaguchi *et al.*, "ATM Transport System Based on Flexible, Non-Stop Architecture," in GLOBECOM '93 Conf. Rec., Houston, TX, Nov./Dec. 1993, pp. 1468-1475.

[20] S. Kumar and D. P. Agrawal, "A Shared-Buffer Direct-Access (SBDA) Switch Architecture for ATM-based Networks," ICC '94 Conf. Rec., New Orleans, LA, May 1994, pp. 101-105.

[21] K. J. Schultz and P. G. Gulak, "CAM-Based Single-Chip Shared Buffer ATM Switch," ICC '94 Conf. Rec., New Orleans, LA, May 1994, pp. 1190-1195.

[22] W. Fischer *et al.*, "A scalable ATM Switching System Architecture," *IEEE J. Select. Areas Commun.*, vol. 9, Oct. 1991, pp. 1239-1307.

[23] A. Jajszczyk and W. Kabaciński, "A Growable Shared-Buffer-Based ATM Switching Fabric," in Globecom '93 Conf. Rec., Houston, TX, Nov./Dec. 1993 pp. 29-33.

[24] S. X. Wei and V. P. Kumar, "On the Multiple Shared Memory Module Approach to ATM Switching," in Proc. INFOCOM '92, Florence, Italy, 1992, pp. 116- 122.

[25] S. C. Liew and K. W. Lu, "A 3-Stage Interconnection Structure for Very Large Packet Switches," in ICC '90 Conf. Rec., Atlanta, GA, April 1990, pp. 771-777.

[26] P. C. Wong and E. H. Tung, "A Large Scale Packet Switch Interconnection Architecture Using Overflow Switches," in ICC '93 Conf. Rec., Geneva, Switzerland, May 1993, pp. 708-714.

[27] M. Collivignarelli *et al.*, "System and Performance Design of the ATM Node UT-XC," ICC '94 Conf. Rec., New Orleans, LA, May 1994, pp. 613-618.

[28] M. De Prycker *et al.*, "An ATM Switching Architecture with Intrinsic Multicast Capabilities for the Belgian Broadband Experiment," Proc. ISS '90, Stockholm, Sweden, vol. V, May/June 1990, pp. 111-118.

[29] H. Obara and S. Okamoto, "Self-Routing Fast Packet Switch with in-Service Modular Growth," *Electron. Lett.*, vol. 26, Aug. 1990, pp. 1286-1287.

[30] S. Tanabe and K. Ohtsuki, "A Hyper-Distributed Switching System Architecture for B-ISDN," in ICC '91 Conf. Rec., Denver, CO, June 1991, pp. 1431-1435.

[31] A. Jajszczyk, "A Growable Folded ATM Switching Fabric Architecture," in Proc. Broadband Communications '94, Paris, France, March 1994, paper 2.1.

Biographies

JOAN GARCIA-HARO [S '92] received an M.Sc. in telecommunication engineering from the Polytechnic University of Catalonia (UPC) in 1989. In 1990 he began his research activity at the Department of Applied Mathematics and Telematics at the UPC with a grant from the Spanish Science and Education Ministry. From October to December 1991 and from June to December 1992 he was a visiting research assistant at Queen's University in Kingston, Ontario, Canada. Since 1992 he has been an assistant professor at the UPC, Barcelona, Spain. His primary research interest is the design and performance evaluation of ATM switching architectures and mobile communication networks. His e-mail address is: teljgh@upc.es.

ANDRZEJ JAJSZCZYK [M '91] received an M.S. in electrical engineering in 1974, and Ph.D. and Dr.Hab. degrees in communications from the Technical University of Poznań in 1979 and 1986, respectively. From 1974 to 1992 he was with the Technical University of Poznań. In 1993 he joined the Franco-Polish School of New Information and Communication Technologies (EFP) in Poznań, Poland, where he is presently a professor and head of the Telecommunications Switching and Networks Group. In 1989-90 he spent 12 months as a visiting scientist at the Teletraffic Research Center, University of Adelaide, Australia. In 1991 and 1992 he was a visiting scientist at Queen's University, Kingston, Ontario, Canada. He has been engaged in research and teaching in the areas of telecommunication switching and computer communications. He is the author of four books (in Polish) and more than 90 papers, as well as 19 patents in these areas. He has led several projects for industry concerning the design and performance evaluation of digital switching systems. His current research interests include broadband networks and switching, both electronic and photonic. He has served as a consultant to telecommunications industry and operators as well as government agencies in Poland, Australia, and Canada. He is editor of the *IEEE Global Communications Newsletter*, as well as editor for Switching Theory and Fabrics for the *IEEE Transactions On Communications* and associate technical editor for *IEEE Communications Magazine*. His e-mail address is: jajszcz@efp.poz.edu.pl.