Pipelined Multi-Queue Management in a VLSI ATM Switch Chip with Credit-Based Flow-Control*

George Kornaros[†], *Christoforos Kozyrakis*[‡], *Panagiota Vatsolaki*[†], *and Manolis Katevenis*[†]

Institute of Computer Science (ICS), Foundation for Research and Technology – Hellas (FORTH)
Science and Technology Park of Crete, Vassilika Vouton, P.O.Box 1385, Heraklion, Crete, GR 711 10 Greece
E-mail: katevenis@ics.forth.gr Tel.: +30 (81) 391664 Fax: +30 (81) 391661

Proc. of 17th Conf. on Advanced Research in VLSI (ARVLSI'97), Univ. of Michigan, Ann Arbor, USA, Sep. 1997
URL: ftp://ftp.ics.forth.gr/tech-reports/1997/1997.ARVLSI.Pipe_MultiQueue.ps.gz

ABSTRACT: We describe the queue management block of ATLAS I, a single-chip ATM switch (router) with optional credit-based (backpressure) flow control. ATLAS I is a 4-million-transistor 0.35-micron CMOS chip, currently under development, offering 20 Gbit/s aggregate I/O throughput, sub-microsecond cut-through latency, 256-cell shared buffer containing multiple logical output queues, priorities, multicasting, and load monitoring. The queue management block of ATLAS I is a dual parallel pipeline that manages the multiple queues of ready cells, the per-flow-group credits, and the cells that are waiting for credits. All cells, in all queues, share one, common buffer space. These 3- and 4-stage pipelines handle events at the rate of one cell arrival or departure per clock cycle, and one credit arrival per clock cycle. The queue management block consists of two compiled SRAM's, pipeline bypass logic, and multi-port CAM and SRAM blocks that are laid out in full-custom and support special access operations. The full-custom part of queue management contains approximately 65 thousand transistors in logic and 14 Kbits in various special memories, it occupies 2.3 mm^2, it consumes 270 mW (worst case), and it operates at 80 MHz (worst case) versus 50 MHz which is the required clock frequency to support the 622 Mb/s switch link rate.

KEYWORDS: *single-chip ATM switch, VLSI router, pipelined queue management, credit-based flow control.*

1. Introduction

Processors, memories, switches (routers), and I/O interfaces are the general-purpose building blocks of contemporary and future unified computer-communication systems. Interconnection networks play a central role at all scales – system, local, or wide area networking. Several architectural issues are similar in all these levels. High-speed interconnection networks use point-to-point links and switches in order to avoid turn-around and arbitration delays and in order to provide increased communication parallelism.

Quality of service (QoS) is of increasing concern in interconnection networks. Improved QoS requires, on one hand, reduced latency for high-priority traffic. Among others, this means preemptive scheduling, i.e. packet switching based on small-size quanta. In two popular technologies, this is achieved using *cells* in Asynchronous Transfer Mode (ATM) [14] or *flits* in Wormhole Routing [5]. An ATM switch is a system that buffers and routes ATM cells, i.e. fixed-size quanta consisting of a 5-byte header and a 48-byte payload each. User packets are segmented into cells at the source, and reassembled from cells at their final destination. ATM is connection-oriented: cells are routed based on their connection ID (virtual path / virtual circuit number,

† also with the University of Crete, Department of Computer Science

‡ currently with the University of California at Berkeley, Computer Science Division. Email: kozyraki@cs.berkeley.edu

VP/VC), which appears in their header; connections (fixed paths), and their corresponding routing-table entries, must be set up before cells can flow along them.

Improved QoS also requires sophisticated flow control, which is based on managing each flow (connection, group of connections, or set of packets) individually. To achieve this, switches need to maintain and manage *multiple queues*, thus avoiding head-of-line blocking, and allowing high-priority or unblocked flows to overtake lower-priority or blocked traffic. Implementing multiple queues and managing them at high speed will play an increasingly important role in switch design for high-performance networks.

This paper presents the implementation and management of multiple queues and of sophisticated flow control information in hardware, inside a high-speed VLSI router. We describe the core of the control section of *ATLAS I*, a 20 Gb/s single-chip ATM switch. In the past, switching and routing was often done "in software", on general-purpose computers or using embedded processors, especially in LAN and WAN environments; purely hardware routers were used mostly in multiprocessors. Today, with the advent of gigabit networking, with the desire for improved QoS including short latency, and with the resulting small cell/flit size, high-performance switching has to be implemented in hardware [22], and switches must employ flexible data-structures for the cells stored in them, which were not necessary before [4]. Contemporary VLSI technology provides the necessary capacity to fit such advanced switches on a single-chip, thus avoiding the high cost and performance bottlenecks of multi-chip organizations. Hardware management of non-trivial data structures inside VLSI switches has appeared before in wormhole routers, e.g. [20], [3], [7]. VLSI ATM switches have also been developed, e.g. [11], [18], [6].

Relative to such previous work, ATLAS I differs as follows. ATLAS maintains multiple logical output queues as a shared pool of identifiers with a single shared controller, avoiding the high area cost of dedicated buffers and controllers per output queue, as in the Prizma switch [6]. In addition, the operation of the single controller in ATLAS I is pipelined, since it has to manage the cell data structures at much higher rates than previous switches [20] [11]. The ATLAS switch implements multilane credit-based flow control (section 2), while other switches either provide no flow control at all [6] [11] [20], or only single-lane flow control [18]. Certain wormhole routers, like iWarp [3] and Spider [7], support multi-lane backpressure flow control for a small number of channels or flow-groups, and implement queuing with a dedicated fixed-sized buffer per channel. ATLAS I, on the other hand, supports 65,536 flow-groups (4096 on each of the 16 ports), manages 54 output queues of "ready" cells, and keeps track of a number of "creditless" cells; all cells and queues *dynamically share* a single buffer space. Finally, ATLAS implements the above mentioned features in a single chip instead of a "closet-sized" box [21], providing a cost effective solution for high-speed interconnection networks.

Thus, ATLAS I provides a combination of high speed, credit-based flow control, multiple queues inside a shared buffer, and multicasting that is unique among wormhole routers and VLSI ATM switches. To implement this combination, a sophisticated queue management unit is needed; it uses multi-port and content-addressable memories, which operate inside two parallel pipelines where bypassing resolves dependencies without stalling. This paper presents the queue management unit of ATLAS I.

Section 2 gives an overview of the ATLAS I single-chip ATM switch. Section 3 presents the architecture and internal organization of the queue manager of ATLAS I, including a description of its dual pipeline. Finally, section 4 describes the VLSI implementation of this queue manager, placing particular emphasis on the parts of it that were laid out in full-custom.

2. Overview of the ATLAS I Switch Chip

ATLAS I is a general-purpose, single-chip, gigabit ATM switch, intended for use in high-throughput and low-latency systems ranging from wide area (WAN) to local (LAN) and system/desktop (SAN/DAN) area networking, supporting a mixture of services in a range of applications, from telecom to multimedia and multiprocessor NOW (networks of workstations). ATLAS I is being developed within the "ASICCOM"[1] project.

Figure 1 presents an overview of ATLAS I; the features that are underlined in the figure are dealt with in this paper. This 16x16 switch uses point-to-point serial links running at 622 Mbits/s each (link bundling also allows configurations of 8x8 at 1.25 Gbps/link, 4x4 at 2.5 Gbps/link, etc.). The links run ATM on top of IEEE Std. 1355 "HIC/HS" [1] as physical layer, using the BULL "STRINGS" GBaud serial-link transceivers [15]. HIC was preferred over SONET because of simpler circuitry, lower latency, and the capability to encode (unbundled) credits. Internally, ATLAS I operates as a crossbar, with

[1]funded by the European Union "ACTS" (Advanced Communication Technologies and Services) Programme. The ASICCOM Consortium consists of industrial partners (INTRACOM, Greece; SGS THOMSON, France and Italy; BULL, France), telecom operators (TELENOR, Norway; TELEFONICA, Spain), and research institutes (FORTH, Greece; SINTEF, Norway; Politecnico di Milano, Italy; NCSR Democritos, Greece).

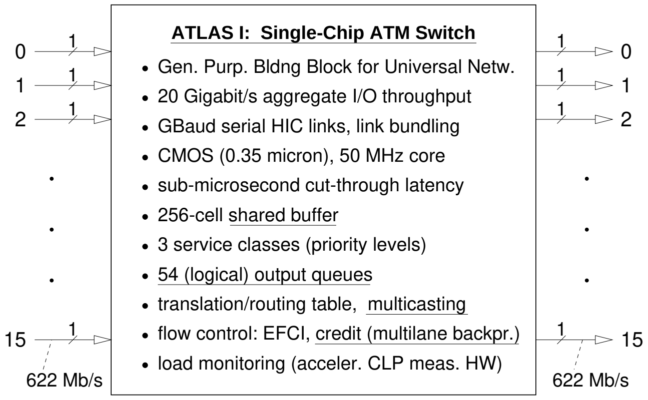

Figure 1: *ATLAS I chip overview*

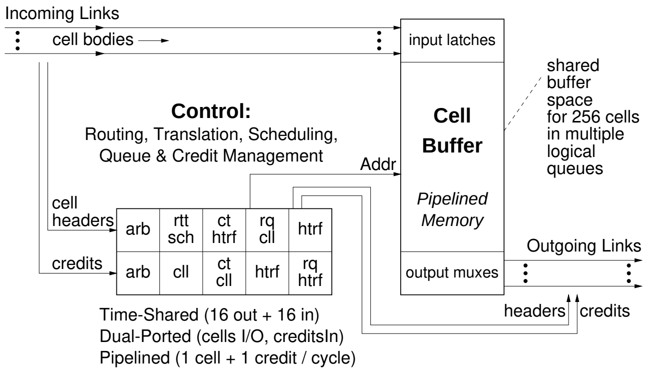

Figure 2: *Simplified ATLAS I block diagram*

a 256-cell shared buffer. Three levels of priority are implemented; each level has its own queues. The cells in the shared buffer are logically organized in 51 output queues (3 times 16 outputs plus 1 management port) and 3 multicast queues; output queueing eliminates head-of-line blocking. Owing to the all-hardware control and to the virtual cut-through provided by the crossbar, cell latency through the switch chip at low network load is well below one microsecond.

One of the most distinctive features of ATLAS I is its (optional) provision of *credit-based flow control* (multilane backpressure). Under backpressure [5], cells are never dropped, because they are only transmitted when buffer space is known to exist at the receiver (i.e. after having acquired a credit). Single-lane backpressure indiscriminately starts or stops the transmission of any cell according to buffer space availability; it performs poorly due to head-of-line blocking effects. Multilane (per-connection) backpressure fixes this problem: connections or groups of connections are queued and flow-controlled independent of each other, allowing the selective bypassing of cells, according to priorities and feedback from downstream congestion. The credit protocol of ATLAS I is reminiscent of QFC [2], but is adapted to hardware implementation over short and reliable links.[2] Provision of credit-based flow control offers significant advantages [13] [8], and places important requirements on the queue management circuits of ATLAS I (section 3).

More information on the aspects of ATLAS I that are not covered in this paper can be found in [9].

Figure 2 shows an abstract block diagram of ATLAS I . Cell bodies are stored in the shared buffer, which also provides switching and virtual cut-through. This buffer is implemented according to the pipelined memory organization, which offers considerable advantages [10]. The pipelined buffer can initiate one cell operation (write or read) per clock cycle. The control section of ATLAS I makes routing decisions, translates ATM connection numbers (VP, VC), schedules the use of the pipelined memory buffer and the servicing of links, and manages the cell queues and the credits. This control section operates on cell headers, buffer addresses, and credit flow-group ID's only – it need not and does not see cell bodies.

ATLAS I uses a clock of frequency 50 MHz or higher, so that there are at least 33 clock cycles per cell-time (the cell-time is approximately 700 ns for the 622 Mbps OC-12 link rate). Sixteen cycles per cell-time are needed in the worst case to service the 16 input links, another 16 cycles are needed for the 16 outputs, and one occasional extra cycle is necessary for the management/microprocessor port. The fabrication technology offers quite fast circuit operation, so there is a considerable safety margin in the speed of the ATLAS I core: we design the circuits to operate at a 12 ns worst-case cycle time, although chip operation will be at 50 MHz eventually.

ATLAS I Core Complexity		
Block	Logic *(Kxtors)*	Memory *(Kbits)*
Queue Management:		
CAM's, multiport RAM's (full-custom)	65	14
Credit Table (semi-custom)	10	70
Control (semi-custom)	10	3
Other Core (semi-custom):		
Link Interfaces, Elastic Buffers	260	-
Cell Buffer	30	110
Routing, VP/VC Translation	20	300
Scheduler, Control, Miscellanies	75	6
Total	470	503

[2]each credit corresponds to one cell-slot in a buffer, and identifies one *flow group* for which it is destined; flow groups are sets of connections that are flow-controlled together; there can be up to 4096 different flow groups on each ATLAS I link; there can be at most 1 cell of any given flow group inside ATLAS I at any given time, for the reasons explained in section 3.1.

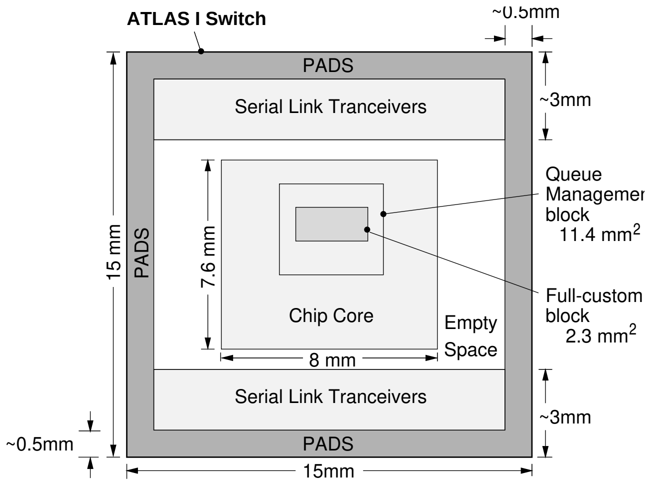

Figure 3: *ATLAS I switch chip floorplan*

At the time of this writing, most of the switch blocks have been fully designed, at the gate or transistor level, and verification is in progress. ATLAS I will be fabricated in a 0.35 micron CMOS process, by SGS Thomson, Crolles, France. The majority of the chip uses semi-custom logic and compiled SRAM blocks, while a small part had to be laid out in full-custom; the reasons for using full-custom and the circuits in it are analyzed in section 4. As shown in figure 3, the chip will consist of a core, placed inside a pre-existing ring. The ring is supplied by BULL, Les Clayes sous Bois, France; it contains the bonding pads and the GBaud serial-parallel converters. The core contains the switch logic and occupies approximately 60 mm^2. Its complexity is given in the table below, measured in thousands of transistors for logic circuits and in Kilobits for SRAM. The elastic buffers are implemented out of discrete flip-flops, so the logic count of the link interfaces appears inflated. The rest of this paper deals with the Queue Management Block, which is the hardest part of the ATLAS I core; section 3 describes its architecture, and section 4 discusses the implementation of its full-custom part.

3. Queue Management Organization

The heart of the ATLAS I control is the Queue Management (QM) Block, which keeps track of the cells that are waiting for credits, the credits that are waiting for cells, and the cells that are ready for departure. This section presents the architecture and hardware organization of the QM block.

3.1 Logical Queue Architecture

Cells are treated differently according to whether or not their connection (VC) is subject to backpressure. The high priority class is intended for real-time policed traffic, like voice, where dropping cells is preferable over delaying them; hence, this class is not subject to backpressure. The middle priority class is intended for policed, backpressured traffic (e.g. data with service guarantees); low priority is for non-policed, backpressured traffic (e.g. flooding data). Hence, only middle and low priority connections are subject to credit-based flow control, when such control is enabled. Cells that are not subject to backpressure, and cells that are backpressured but have acquired the credit that is necessary for their departure are called *ready* cells, because they are ready for departure as soon as their outgoing link(s) become available. Ready cells are placed in the *ready queues*. There is one ready queue per outgoing link and per priority level for unicast cells, as shown in figure 4. This output-queueing organization eliminates the head-of-line blocking associated with input queueing.

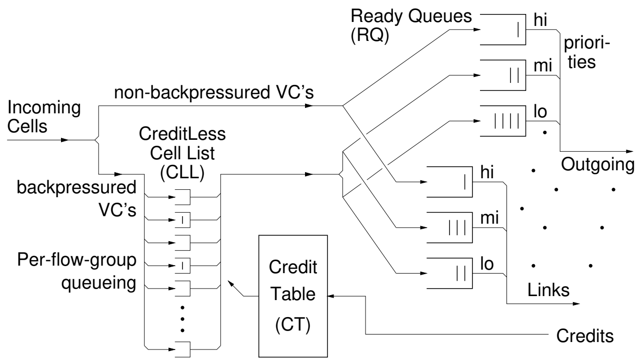

Figure 4: *ATLAS I logical queue architecture*

There is also one ready queue per priority level for multicast cells (not shown in figure 4), shared among all links. Each cell can be in at most one queue at a time; this restriction, imposed in order to control hardware complexity, is the reason why multicast queues are shared among links. If the head cell in a multicast queue is destined to a set of outputs A, multicast cells behind it are blocked, including those destined to other outputs, B. We estimate that this blocking will have a minimal effect on performance, because *(i)* unicast cells destined to outputs B are not blocked; *(ii)* within each service (priority) class, the scheduler services the multicast queue with higher priority than the unicast queues; and *(iii)* cells in the multicast queue have already acquired *all* credits needed, so they depart to all of their destinations within one cell-time.

Backpressured cells that have not found an appropriate credit wait in the *CreditLess Cell List* (CLL) until such a credit arrives. The CLL is conceptually organized as a set of queues, one per flow group, so that, when a credit arrives, it can be acquired by a cell of its flow group without being blocked by other flow groups (a flow group is a set of connections that are flow-controlled together). ATLAS I can handle up to 64 K flow groups (4 K per link), so a straightforward implementation of the CLL using RAM would be very expensive; instead, some key observations led to a manageable implementation using content-addressable memory (CAM). First, there can be at most one cell per flow group inside an ATLAS I chip[3]. In this way, there is no need to remember any cell sequence inside a flow group, so no real queues are needed inside CLL. Second, given that there can be at most 256 cells inside ATLAS I at any given time, there can be at most 256 non-empty "queues" in the CLL. Hence, the CLL is implemented as a CAM, as discussed in section 4.1.

3.2 *Centralized versus Distributed Controller*

ATLAS I uses a single control unit, which is time-shared among all links. This centralized controller was preferred over a "distributed" architecture –where each input or output link or both would have their own control circuits– for a number of reasons:

- a different number of cells in the data buffer may be destined to each output; if each output had its own controller, each one of these controllers would have to keep track of a potentially large number of cells (256 in the worst case); by maintaining all logical queues inside a single, central data structure (of size 256), unnecessary extra storage is eliminated;
- multicasting and link bundling are simpler with a centralized control architecture;
- when two controllers in the distributed architecture need to communicate, arbitration is needed, which introduces complexity; in the centralized architecture, arbitration is only needed at a single place: the first stage of the control pipeline;
- operation of the central controller matches well and is synchronized to the operation of the pipelined memory buffer.

The centralized controller needs to have a higher *average* throughput than each controller in a distributed architecture (but its *peak* throughput may not be higher). Thus, the design of the central controller is more of a challenge, but it is doable and advantageous, as we show in this paper. In order to provide the required switch throughput, the ATLAS I control unit must handle cell (arrival/departure) events at the rate of one per clock cycle; to achieve this, the unit is pipelined. Incoming credits are handled by a second, parallel pipeline of the control unit, at the rate of one per clock cycle; in this way, up to 2 credits per link per cell-time can be serviced, which is twice the worst-case average rate. The two pipelines of the control unit (figure 2) start with an arbitration stage, that chooses one of the incoming links that are currently requesting service. The cell pipeline continues with the routing/translation stage, in parallel with the scheduler that decides whether to service an output or an input. The rest of the stages concern queue management, and are explained more at length below.

3.3 *Hardware Data Structures*

Figure 5 presents the data structures maintained and managed by the ATLAS I hardware in order to implement the above queue architecture. All information that concerns a given cell is stored in memories that are "parallel" to the cell buffer, i.e. the address of the cell body in that buffer is the *identifier* of that cell, and is used to access all related information. Ready queues are maintained as linked lists of cells; since each cell can belong to at most one ready queue (section 3.1), one next-pointer per cell is needed, held in the Queue Pointer memory (the primary port field in that memory concerns multicast cells only, and is used while scheduling their transmission). The fact that each cell can be in at most one ready queue also restricts the peak throughput requirements on the controller: otherwise, inserting cells into multiple queues at certain times would dictate a much higher instantaneous rate of operations. The 54 queues that are needed in ATLAS I are identified by the contents of a 54x17 memory, the *Head/Tail Register File (HTRF)* (the valid bit distinguishes empty queues from non-empty ones). One alternative that was examined but rejected was to implement ready queues using *sequence numbers*. For each queue, a *first* and *last* sequence counter would be kept (this is similar to the HTRF). For each cell, a sequence number would be stored along with the rest of the cell status (this is similar to the next-pointer field). However, in order to locate the front cell in a queue, a content-search would be needed on the sequence number field, instead of a simple RAM access with the linked-list organization.

Other information maintained for each cell is: its *flow group* number (12 bits), i.e. the identifier of the set of connections to which it belongs for flow-control

[3]this restriction does *not* limit the maximum achievable throughput per flow group in SAN environments, because ATLAS I is fast enough for the cell-credit round-trip-time to be less than one cell-time for short links; thus, even a single active connection can saturate such a short link. Additionally, this restriction has a beneficial effect on burst and hot spot tolerance [8]: a mis-behaving flow group is not allowed to use up more than one slot in the ATLAS I buffer memory.

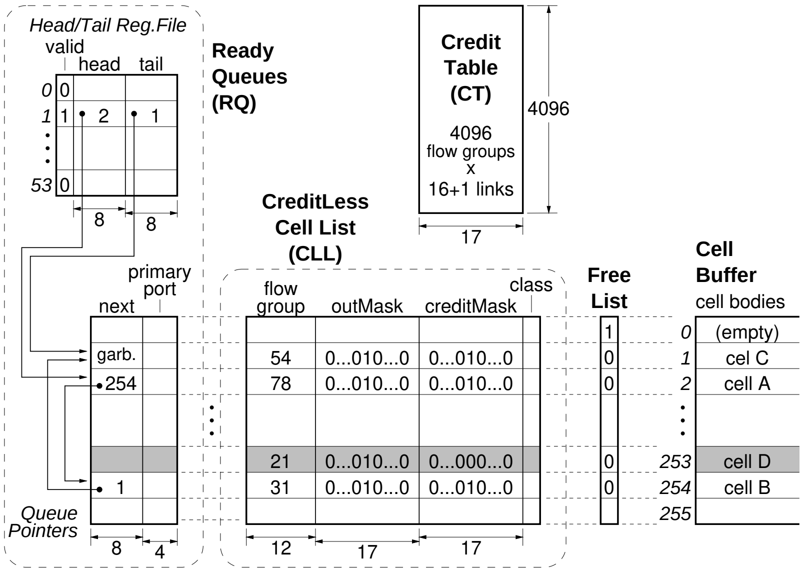

Figure 5: *Queue management data structures*

purposes; the *outMask*, which is a 17-bit mask identifying the desired outgoing link(s) for the cell (the 17th link corresponds to the management/external-microprocessor port); the *creditMask*, a 17-bit mask indicating the outgoing link(s) for which the cell has obtained a credit; and the service *class* identifier. All this information is important for *creditless* cells, so the memory that holds it is called the CreditLess Cell List (CLL). The outMask field is also of interest for ready multicast cells (other ready cells can deduce all information needed from the queue number that they belong to). Creditless cells do not belong to any ready queue, so their next-pointer field is not used. In the example of figure 5, the (shaded) cell 253 is creditless (its creditMask and outMask differ in one bit).

Five separate memory blocks are used in ATLAS I to implement the above organization. This separation is discussed in section 4, and was necessary because of the different access types and timing for each field. A sixth, 256x1 memory with a priority encoder is used as the *Free List*, identifying empty slots in the cell buffer.

3.4 Queue Management Pipelines

The operations performed on the above data structures are cell arrival, cell departure, and credit arrival (credit departures occur upon cell departures, and are included in that operation). As explained in section 3.2, the ATLAS I controller performs these operations using two parallel pipelines –one for cell operations and one for credit arrivals. Cell operations are initiated by the switch scheduler. The scheduler gives higher priority to servicing outgoing links over incoming links, since delaying the outputs causes unnecessary loss of throughput, while incoming cells will never be lost, independent of when in the cell-time they are serviced. For cell arrivals, routing and translation occurs in parallel with scheduling, thus determining the cell's flow group, outMask, and service class. Queue management takes place in the last three stages of the cell pipeline, and in the last four stages of the credit pipeline, as illustrated in figure 6 and explained below.

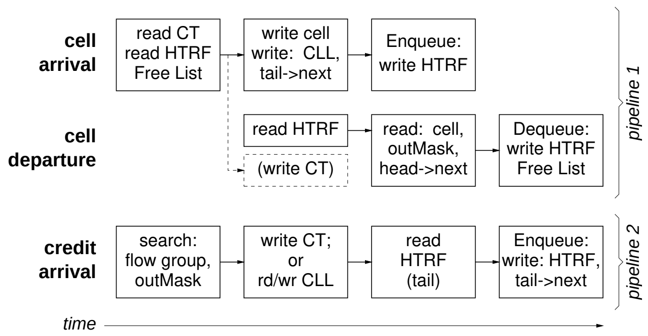

Figure 6: *The two queue management pipelines*

Cell Arrivals: During the first QM pipeline stage, the Credit Table (CT) is read, looking for credits for this cell. In parallel, the service class and the outMask of the incoming cell are used as index to read the proper RQ tail pointer from the HTRF; this will be used to enqueue the cell, in case it is ready. Also, the Free List is searched for the first empty slot.

In the next stage, the cell is recorded in the CLL. If the cell is ready, the tail pointer of the proper queue is used to write the address of the arriving cell in the next-pointer entry. The credits that were read from CT and that were needed by the cell (CT & outMask) are recorded in the CLL and have to be cleared in the CT. This clear operation would normally occur during this stage, and would require an additional port in the CT memory. In order to avoid this additional port, so that the CT can be a compiled SRAM rather than laid out in full custom, the clear operation is recorded in a "write buffer" and performed during the first stage of a cell departure operation (cell departures do not need to access the CT). There are 17 registers in this write buffer, together with the corresponding comparators and bypass logic.

In the final stage, if the cell was ready, the new tail pointer and the non-empty bit are written into the HTRF memory. If the queue was previously empty, the head pointer must be written as well, and the scheduler is notified that the queue is no longer empty. These HTRF accesses could alternatively be performed one cycle earlier, but that would conflict with the HTRF access pattern of cell departure operations.

Cell Departures: During the cycles when the scheduler instructs the cell pipeline to handle a cell departure on a given output at a given priority class, operations proceed as follows. In the first stage, the head and tail pointers for the corresponding ready queue are read from the HTRF. The head pointer is used to locate the cell body (for transmission) and the rest of the information about it; the tail pointer is read in order to detect if this was the last cell in the queue (head==tail).

In the second stage, the OutMask memory is accessed as follows: the bit corresponding to the current output link is reset, while the remaining 16 bits are read. If these other 16 bits are all zero, the cell must be dequeued and its slot given to the free list, because it was either a unicast cell or a multicast cell currently departing on its last output. The pointer to the next cell in the queue is also read in parallel, since it will be needed in the case of dequeueing. In the third stage, the cell is dequeued if necessary. This involves notifying the Free List and updating the queue head pointer in the HTRF, and the valid bit if this was the last cell in the queue; in the latter case, the scheduler is notified that this queue is now empty.

Credit Arrivals: Credit arrivals are handled by the second pipeline, at the rate of one per clock cycle, in parallel with cell events. This is a 4-stage pipeline. In the first stage, the credit flow-group and a 16-bit mask indicating the link it arrived from are used to search the CLL (flow-group and OutMask fields) for cells waiting for this credit. This search requires a full match on the flow-group field, while the OutMask is only searched for a bit that is set, which simplifies the circuit. No creditMask read is necessary: the fact that a credit just arrived implies that this flow group had no credit before.

In the second stage, if no cell was waiting for this credit, the credit is written into the CT; since there can be at most one credit per flow group, a simple write-1 suffices –no read-modify-write is required. Else, the creditMask of the selected cell is updated by setting the bit that corresponds to the arrival link, while reading the other 16 bits; the outMask is also read. By comparing the two masks, it is determined whether the cell became ready, i.e. whether it received the last credit it was waiting for and has to be enqueued. The third and fourth stages perform the enqueue operation, if the cell just became ready. Using the cell's outMask and service class, the proper queue address is formed to read the HTRF. In the next stage, tail→next is written, and tail and valid are updated; if the queue was previously empty, head must also be written and the scheduler must be notified.

3.5 Control of the Queue Management Pipeline

The queue management pipeline is controlled in a way similar to the usual RISC processor pipelines: control signals are generated during the initial stage, and they are propagated down the pipeline, through pipeline registers, until they are "consumed" in the appropriate stage. Pipelined and multiported operation can lead to data hazards: operations performed in parallel but without coordination could lead to inconsistent results, like destroying the connectivity of a queue, generating non-existing cells, dropping cells or credits, etc.

In the ATLAS I case, all data hazards can be resolved using *bypassing* (forwarding) [17], without ever needing to stall the pipeline. There are 2 bypass cases to handle (almost) simultaneous arrivals of a cell and its corresponding credit, and 12 bypass cases to handle concurrent or back-to-back operations on the same Ready Queue. Five comparators (one 16-bit, one 12-bit, and two 8-bit) and a few tens of gates detect and control the bypass conditions. Six multiplexors are used to perform the bypasses (two 8-bit 2-to-1, one 8-bit 3-to-1, one 1-bit 2-to-1, and two 1-bit 3-to-1 multiplexors). Correctness of the pipelines and their control was verified by the parallel simulation of two models: *(i)* behavioral model of queue management transactions, performed one-by-one, without interleaving; *(ii)* functional (RTL) model of the two pipelines and their control logic. The test vectors included some random patterns, as well as a computer-generated exhaustive list of all 432 possible cases of concurrent or back-to-back cell and credit oper-

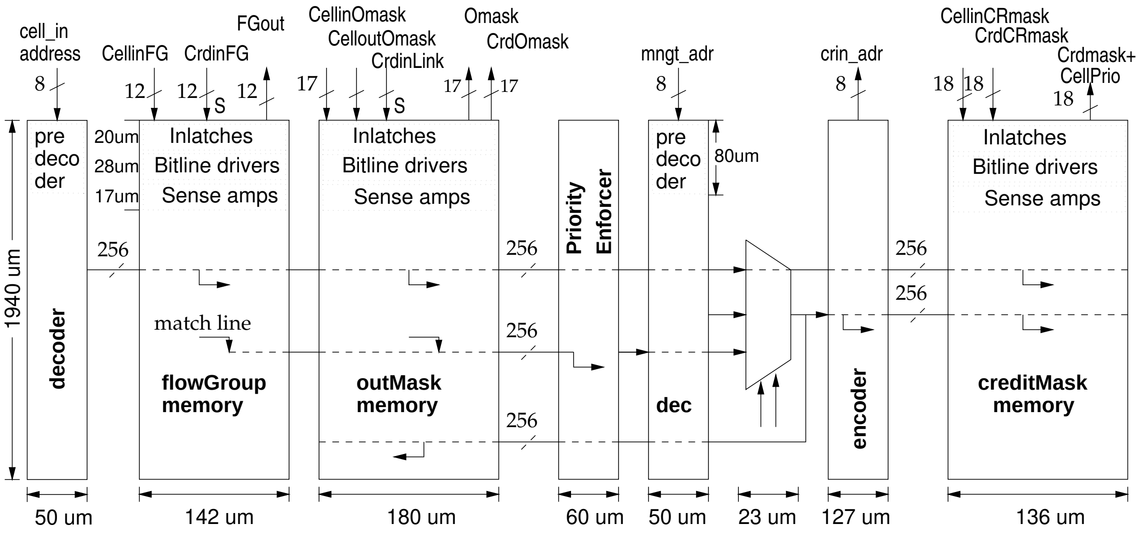

Figure 7: *The main full-custom block*

ations, each one being applied to queues in 3 different states (empty, single cell, two or more cells). Correctness was checked by comparing the queue's status and the outputs of the two simulators. A detailed description of the ATLAS I queue management unit, including all the bypassing rules, can be found in [12].

4. VLSI Implementation of Queue Management

As seen in section 3, the ATLAS I queue management (QM) unit requires RAM and CAM blocks with multiple access ports. In the design environment that we use, single- or dual-ported SRAM's can be generated by a compiler, while SRAM's with 3 or more ports and CAM's have to be laid out in full-custom. The table below summarizes the port requirements of the QM memory blocks; for full-custom memories we also give the type of the ports required.

Memory Block	Ports	Memory and Port Type
Credit Table	2	compiled 2-port SRAM
next pointer	2	compiled 2-port SRAM
flowGroup	2	read/write (1), CAM (1)
outMask	3	read (1), read/modify (1), CAM (1)
creditMask	2	write (1), read/modify (1)
Head/Tail RF	4	read (2), write (2)
Free List	2	CAM (1), write (1)

The Head/Tail Register File (HTRF) is the most demanding memory block in terms of number of ports. Alternative organizations were examined to see if we could reduce this number. First, one could maintain separate memory banks for head and tail pointers. Still, enqueue operations on empty queues require 4 ports for the tail register file, because both head and tail pointers must be updated. Second, one could keep separate head and tail banks, while also replicating the tail bank. While updates to tail pointers would have to be made in both copies, read operations could be served by either one, thus reducing the number of ports to 3 for each bank. However, the total area would be larger than one 4-port memory, and full-custom layout would still be needed.

During the design of the full-custom parts, we opted for simplicity and robustness of operation, rather than using complicated techniques for achieving highest speed. ATLAS I has a considerable architectural complexity, so we preferred to reduce the design time and risk of the full-custom part. As mentioned in section 2, we designed all of our circuits to operate at a 12 ns cycle time under worst-case conditions, although the OC-12 link rate of 622 Mb/s only requires a 50 MHz clock. In this way, we have a good safety margin, we need less timing optimization after placement and routing, and we are prepared for a higher-speed ATLAS II switch. We preferred synchronous designs, which are less sensitive to delay skews, are easier to verify, and do not need carefully-tuned on-chip timing signal generation. For memory configurations, we used the conventional static CMOS memory cell, which has lower power dissipation in standby mode and greater immunity to transient noise and voltage variation than other cells. Large loads, such as common clock signals, are driven by a tree of buffers rather than a single, large driver, so as to reduce power dissipation. All branches of the clock tree were balanced to avoid skew. We used multiple power supply lines in order to reduce the amplitude and duration of transients.

4.1 *Full-custom part description*

The main full-custom block consists of three memories –flowGroup, outMask, creditMask– two decoders, one priority enforcer, one normal encoder, and their peripheral circuits. Its block diagram and floor plan is shown in figure 7. Another full-custom block is the four-ported Head/Tail Register File SRAM, and the third full-custom block is the Free List, a chain of 256 D-flip-flops, a decoder, a priority enforcer, and an encoder.

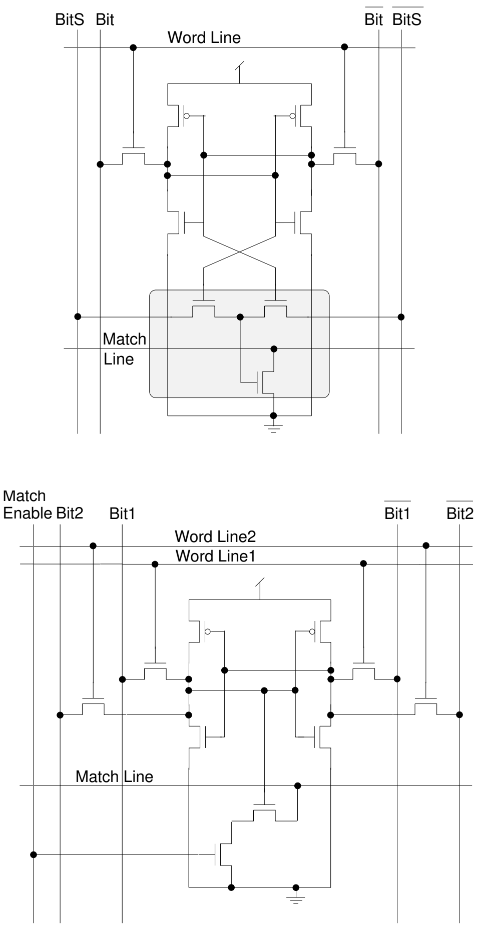

Figure 8: *The dual-ported flowGroup cell and the three-ported outMask cell*

The CMOS technology used has 0.35 μm gate length (NMOS & PMOS), twin-tub, a single polysilicon layer, and five metal layers. The power supply is 3.3 Volts (+0.3/-0.6 V).

In the block of figure 7, the *creditMask* is the simplest part. It uses a conventional dual-ported SRAM cell, but its peripheral circuits are modified to support the read/modify operation: one bitline is externally pulled down to set one bit, while the other bitlines are used for reading the remaining bits. The *flowGroup* memory has 3072 associative memory elements, organized as 256 rows of 12 bits each. The cell contains 9 transistors and is shown in figure 8. Although the NAND version of the CAM cell (with 10 transistors) occupies less area, we did not use it due to its slower discharging of the matchline [16]. The *outMask* memory contains 256x17 memory cells. The first port is used for normal read/write accesses, the second port for read/modify, while the third port is used for a special type of content addressability. When searching outMask, any stored word that has one or more common "1" bit(s) with the search pattern will give a match. Thus, a single bitline (match enable) per column is used to search this memory [19].

As shown in figure 7, the sense amplifiers and output latches are laid out between the bitline drivers and the memory core, thus decreasing the length of clock wires and reducing potential skew problems. Address predecoders and sense amplifiers are only enabled only during the cycles when they are needed, thus economizing on power dissipation.

The priority enforcer is necessary for the proper operation of the CAM, since multiple words can match during a search. Its input is a vector consisting of the matchline results. The output of the priority enforcer is a vector of the same size with a single "1" at the "first" 1 in the input vector. A dual-tree structure was designed (figure 9) to implement a kind of carry lookahead and carry prediction [23], propagating the existence of any single match (the outputs *NE* mean "noone else higher matched"). Dynamic precharged logic with domino timing was used to reduce the delay. Furthermore, the priority enforcer was separated in two stages, each operating on a different clock phase; one stage is precharged while the other one is operating (one-phase pipeline latches exist between the two stages). The first stage includes a two-level tree of dynamic NOR gates, serving as a lookahead generator for the second pipeline stage. The latter consists of Manchester chains operating on separate groups of 16 matchlines. The priority enforcer was laid out in an area of 60μm x 1857μm; the vertical dimension is dictated by pitch matching to the CAM block. Simulation results (figure 10) proved it to be operational at 100MHz clock frequency, under worst case simulation conditions. A detailed description of the architecture and the operation of the priority enforcer can be found in [12].

Another full-custom block, not included in figure 7, is

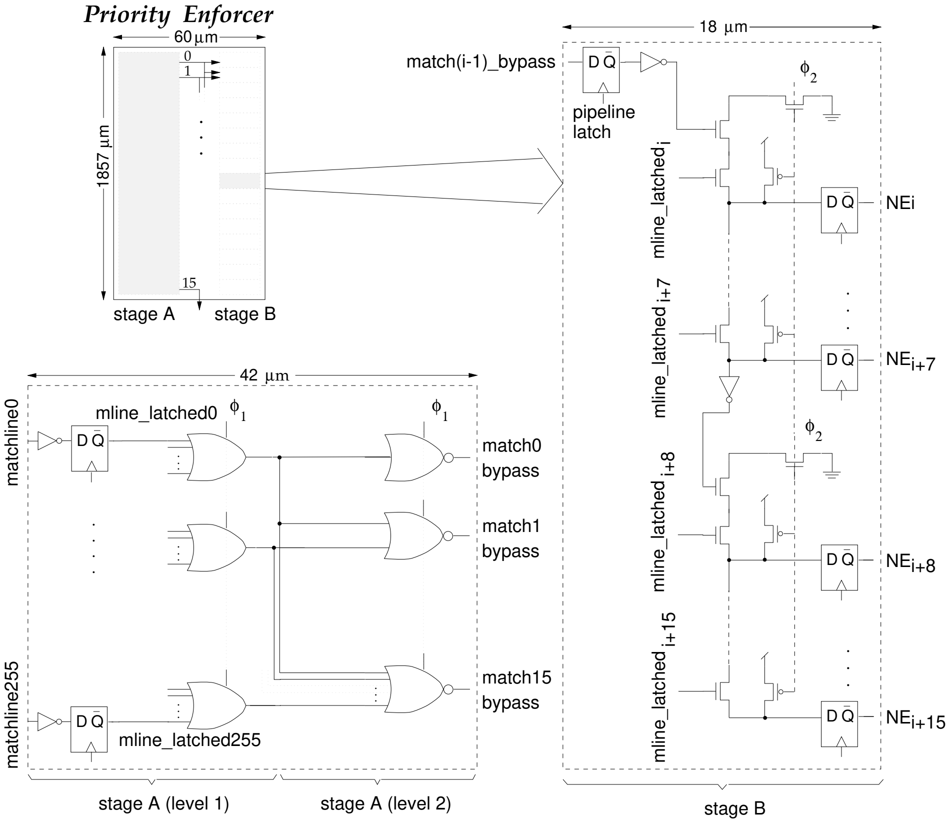

Figure 9: *Priority enforcer circuits and floorplan*

the Head/Tail Register File. It employees a four-ported SRAM cell depicted in figure 11. Its relatively small size –54 rows by 17 columns– easily provides a short access time. The cell size is 7.25μm x 14.1μm. The last block in the full-custom part of queue management is the *Free List* (figure 12). It is a string of 256 flip flops, maintaining the state of available slots in the shared data buffer. Initially all slots are marked as 1 (free). A search for an empty position is processed by a priority enforcer, yielding the first 1 found from the top of the bit string. By the end of the cycle, the selected flip flop is cleared, and its encoded address is supplied as output. An additional decoder is necessary to set the appropriate flip flop when the slot with a given address is freed.

4.2 Design Results

The full custom blocks were simulated extensively, using the actual parasitic capacitances that were extracted from the layout. Timing simulation, under worst-case conditions (minimum-current nmos & pmos models, 105^{o}C temperature, 2.7 Volt power supply), yielded the results shown in the table below (external load driver delay is not included). Searches are the slowest operation, given that the match lines span both FlowGroup and OutMask memory blocks. The address and data setup times were measured with respect to the clock entering the full-custom block.

The power dissipation and the area of the full-custom blocks are listed in the following table. Power dissipation is averaged over a 100ns simulation interval; input activity was maximized. Typical figures refer to 25^oC, 3.3 Volt power supply, typical-current transistor models; maximum refers to 0^oC, 3.6 Volt power supply, high-current transistor models. A major fraction of the power is consumed by the memory blocks, which are relatively large and multiported. The total typical power dissipation of the full-custom part is 210 mW, and its area is 2.26mm^2.

5. Conclusions

High-speed operation and quality of service requirements of modern networks lead to the use of switching schemes with sophisticated flow-control. These need efficient hardware implementations for maintaining and managing multiple cell queues. We presented the architecture and VLSI implementation of the queue management in the ATLAS I single-chip ATM switch

Timing Table				
parameter	min	typ	max	unit
FlowGroup match access (from clock)			6.4	ns
OutMask match access (from clock)			6.3	ns
FlowGroup access (from clock)			4.4	ns
OutMask access (both ports in the same cycle)			5.7	ns
CreditMask modify access			4.9	ns
CreditMask cycle time	10			ns
Priority Enforcer (eval. phase stage A)			4.2	ns
Priority Enforcer (eval. phase stage B)			4.7	ns
HTRF access (from clock, by a single port)			3.1	ns
HTRF access (from clock, by all four ports)			4.9	ns
Address setup time	0.5			ns
Data setup time	0.5			ns

Power Dissipation and Area of Full-Custom Blocks					
block	Ityp (mA)	Ptyp (mWatt)	Imax (mA)	Pmax (mWatt)	Area (mm^2)
decoder A	2.6	8.6	3.6	12.9	0.097
decoder B	2.6	8.6	3.6	12.9	0.097
FlowGroup & OutMask mem	28.0	92.4	31.7	114.1	0.624
Priority Enforcer	1.8	5.9	2.5	9.0	0.116
Mux3-to-1	1.9	6.3	2.1	7.6	0.045
Encoder	2.0	6.6	2.3	8.3	0.246
CreditMask mem	9.7	32.0	11.1	40.0	0.264
HTRF mem	7.0	23.1	7.9	28.5	0.200
Free List	8.4	27.7	10.6	38.0	0.566
Total	64.0	211.2	75.4	271.3	2.255

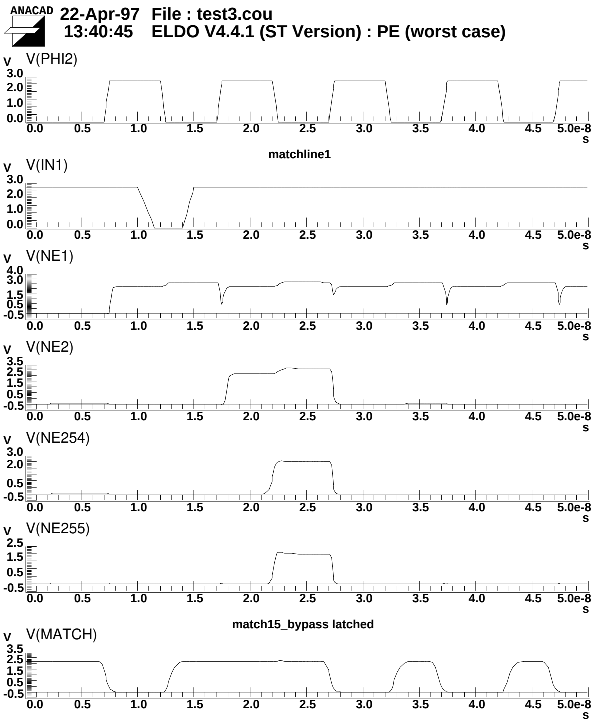

Figure 10: *Priority enforcer waveforms (worst case: no input matched)*

with credit-based flow. This VLSI chip, which is currently being implemented in a 0.35 micron CMOS technology with 5 metal layers and 3.3 V power supply, incorporates several important features: high-speed gigabaud serial links, a 256-cell pipelined memory shared data buffer, 3 levels of priorities, and multicasting capabilities.

The queue management, which implements the central ATLAS I control operations, is responsible for maintaining multiple cell queues and credits, and for handling multicasting. We showed how its pipelined operation supports the high bandwidth requirements, and services one cell and one credit operation per cycle. We presented the implementation of the queue management using compiled SRAM's, semi-custom logic, and full-custom blocks –two, three, and four-port CAMs and RAMs with special ports, pipelined priority enforcers, decoders, and peripheral circuitry. This full-custom part of queue management consists of approximately 65 thousand transistors in logic and 14 Kbits in memories, it occupies 2.3 mm^2, and it has been simulated under worst case conditions to operate at 80 MHz, dissipating 0.27 Watt.

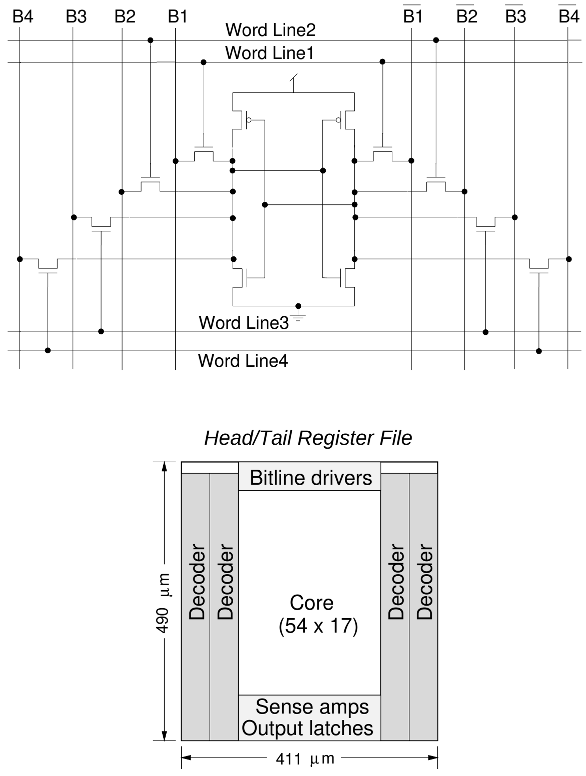

Figure 11: *The four-ported Head/Tail Register File*

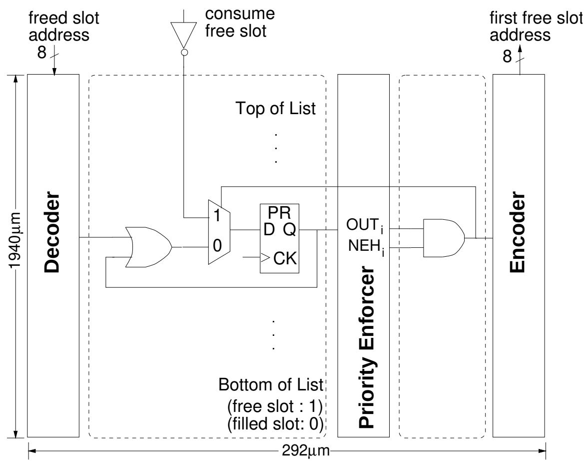

Figure 12: *The Free List circuit*

6. Acknowledgements

This work was carried out within the "ASICCOM" project, funded by the European Union, under the ACTS (Advanced Communication Technologies and Services) Programme. The ATLAS I chip is designed together with Chara Xanthaki, George Kalokerinos, George Dimitriadis, and Dionisios Pnevmatikatos, who have also assisted in the queue management block design. Yannis Papaefstathiou has also helped. We thank them all.

References

[1] *IEEE Standard 1355-1995, ISO/IEC Standard 14575 DIS, Standard for Heterogeneous InterConnect (HIC):low-cost, low-latency scalable serial interconnect for parallel system construction*, 1995. URL: http:-//stdsbbs.ieee.org/groups/1355.

[2] Quantum Flow Control Alliance. Quantum Flow Control: A Cell-Relay Protocol Supporting an Available Bit Rate Service. URL: http://www.qfc.org, July 1995. Version 2.0.

[3] S. Borkar et al. Supporting Systolic and Memory Communication in iWarp. In *Proceedings of the 17th Int. Symp. on Computer Architecture, ACM SIGARCH*, volume 18, pages 70–81, June 1990.

[4] J. Coudreuse, W. Sincoskie, and J.S. Turner. Guest Editorial in Broadband Packet Communications. *IEEE Journal on Selected Areas in Communications*, 6(8):1452–1454, December 1988.

[5] W. Dally and C. Seitz. Deadlock-Free Message Routing in Multiprocessor Interconnection Networks. *IEEE Transactions on Computers*, C-36(5):547–553, May 1987.

[6] W. Denzel, A. Engbersen, and I. Iliadis. A Flexible Shared-Buffer Switch for ATM at Gb/s Rates. In *Computer Networks & ISDN Systems*, volume 27, pages 611–624. Elsevier Science B.V., 1995.

[7] M. Galles. Spider: A High-Speed Network Interconnect. *IEEE Micro*, 17(1):34–39, Jan./Feb 1997.

[8] M. Katevenis, D. Serpanos, and E. Spyridakis. Credit-Flow-Controlled ATM versus Wormhole Routing. Technical Report TR-171, Institute of Computer Science - Foundation for Research and Technology Hellas (ICS-FORTH), July 1996. URL: file://ftp.ics.forth.gr/tech-reports/1996/1996.TR171.ATM_vs_Wormhole.ps.gz.

[9] M. Katevenis, D. Serpanos, and P. Vatsolaki. ATLAS I: A General-Purpose, Single-Chip ATM Switch with Credit-Based Flow Control. In *Proceedings of the Hot Interconnects IV Symposium*, pages 63–73, CA, USA, August 1996. Stanford Univ. URL: file://ftp.ics.forth.gr/tech-reports/1996/-1996.HOTI.ATLAS_I_ATMswitchChip.ps.gz.

[10] M. Katevenis, P. Vatsolaki, and A. Efthymiou. Pipelined Memory Shared Buffer for VLSI Switches. In *Proceedings of the ACM SIGCOMM '95 Conference*, pages 39–48, Cambridge, MA., USA, August 1995. URL: file://ftp.ics.forth.gr/tech-reports/1995/-1995.SIGCOMM95.PipeMemoryShBuf.ps.gz.

[11] T. Kozaki, N. Endo, Y. Sakurai, O. Matsubara, M. Mizukami, and K. Asano. 32x32 Shared Buffer Type ATM Switch VLSI's for B-ISDN's. *IEEE Journal on Sel. Areas in Communications*, 9(8):1239–1247, October 1991.

[12] Christoforos E. Kozyrakis. The Architecture, Operation and Design of the Queue Management Block in the ATLAS I ATM Switch. Technical Report TR-172, Institute of Computer Science - Foundation for Research and Technology Hellas (ICS-FORTH), July 1996. URL: file://ftp.ics.forth.gr/tech-reports/1996/1996.-TR172.QueueManagement.ps.gz.

[13] H. T. Kung. Gigabit local area networks: A systems perspective. *IEEE Communications Magazine*, 30(4):79–89, April 1992.

[14] J. LeBoudec. The Asynchronous Transfer Mode: A Tutorial. *Computer Networks and ISDN Systems*, 24(4), May 1992.

[15] R. Marbot, A. Cofler, J. C. Lebihan, and R. Nezamzadeh. Integration of Multiple Bidirectional Point-to-Point Serial Links in the Gigabits per Second Range. In *Proceedings of the Hot Interconnects I Symposium*, CA, USA, August 1993. Stanford Univ.

[16] N.Troullinos and C.Stormon. Design Issues in Static Content-Addressable Memory Cells. Technical Report 9208, CASE Center Syracuse University, August 1992.

[17] D. Patterson and J. Hennessy. *Computer Organization: the hardware/software interface*. Morgan Kaufman Publishers, 1993.

[18] Y. Shobatake, M. Motoyama, E. Shobakate, T. Kamitake, S. Shimizu, M. Noda, and K. Sakaue. A One-Chip Scalable 8 * 8 ATM Switch LSI Employing Shared Buffer Architecture. *IEEE Journal on Sel. Areas in Communications*, 9(8):1248–1254, October 1991.

[19] S.Sidiropoulos. Fast packet switches for asynchronous transfer mode. Technical Report TR-25, Institute of Computer Science - Foundation for Research and Technology Hellas (ICS-FORTH), Heraklio,Crete,GR, August 1991. URL: file://ftp.ics.forth.gr/-tech-reports/1991/1991.TR25.Fast_packet_switches.ps.Z.

[20] Y. Tamir and G. Frazier. High-Performance Multi-Queue Buffers for VLSI Communication Switches. In *Proceedings of the 15th Int. Symp. on Computer Architecture, ACM SIGARCH*, volume 16, pages 343–354, May 1988.

[21] T.Blackwell, K.Chan, K.Chang, T.Charuhas, B.Karp, H.T.Kung, D.Lin, R.Morris, M.Seltzer, M.Smith, C.Young, O.Bhagat, M.Chaar, A.Chapman, G.Depelteau, K.Grimble, S.Huang, P.Hung, M.Kemp, I.Mahna, J.McLaughlin, M.T.Ng, J.Vincent, and J.Watchorn. An Experimental Flow-Controlled Multicast ATM Switch. In *Proceedings of the First Annual Conference on Telecommunications in Massachussetts*, 1994.

[22] F.A. Tobagi. Fast Packet Switch Architectures for Broadband Integrated Services Digital Networks. In *Proceedings of the IEEE*, volume 78, pages 133–167, January 1990.

[23] N. Weste and Eshraghian. *Principles of CMOS VLSI Design - a Systems Perspective*. Addison-Wesley, 2 edition, 1993.