Implementation of ATLAS I:
a Single-Chip ATM Switch with Backpressure

Georgios Kornaros, Dionisios Pnevmatikatos, Panagiota Vatsolaki, Georgios Kalokerinos, Chara Xanthaki, Dimitrios Mavroidis, Dimitrios Serpanos, and Manolis Katevenis[1]

Institute of Computer Science (ICS)
Foundation for Research and Technology – Hellas (FORTH)
Science and Technology Park of Crete, P.O.Box 1385, Heraklion, Crete, GR 711 10 Greece
http://archvlsi.ics.forth.gr/ Email: katevenis@ics.forth.gr Tel: +30 (81) 391664, fax: 391661

IEEE Hot Interconnects VI Symposium, Stanford University, California USA, 13-15 August 1998
`http://archvlsi.ics.forth.gr/atlasI/atlasI_hoti98.ps.gz`

ABSTRACT: ATLAS I is a single-chip ATM switch with 10 Gb/s throughput, a shared buffer, 3 priority levels, multicasting, load monitoring, and optional credit-based flow control. This 6-million-transistor 0.35-micron CMOS chip is about to be taped out for fabrication. We present here the implementation of ATLAS I; we report on the design complexity and silicon cost of the chip and of the individual functions that it supports. Based on these metrics, we evaluate the architecture of the switch. The evaluation points in the direction of increasing the cell buffer size and dropping VP/VC translation, while other possible modifications are also discussed. The cost of credit support (10% in chip area and 4% in chip power) is minuscule compared to its benefits, i.e. compared to what alternative architectures have to pay in order to achieve comparable performance levels.

KEYWORDS: *single-chip ATM switch, general-purpose building block for universal networking, credit-based flow control, backpressure, VLSI switch implementation cost & evaluation.*

[1] most authors are or were also with the Department of Computer Science, University of Crete, Greece.

1. Introduction

ATLAS I is a single-chip ATM switch with optional credit-based (backpressure) flow control. This 6-million-transistor 0.35-micron CMOS chip will offer 10 Gbit/s aggregate outgoing throughput, sub-microsecond cut-through latency, 256-cell shared buffer containing multiple logical output queues, priorities, multicasting, and load monitoring. The architecture and rough internal organization of ATLAS I were presented at the Hot Interconnects IV Symposium [KaSV96].

This paper presents the design complexity and implementation cost of the chip and of the various functions in it, and evaluates the switch in view of these metrics. We estimate the design complexity in terms of (Verilog) code size and (approximate) human effort. We measure the implementation cost in terms of gates, flip-flops, and SRAM bit counts, silicon area, and power consumption. We break these metrics down by function supported, rather than by hardware block. Then, we proceed to evaluate the design style and library used: semi-custom versus full-custom, and compiled memory availability. Finally, we evaluate the architecture of the switch, in view of the design metrics. We look at the functions that turned out to have a high cost, and discuss how a different organization could lower this cost, or whether they should rather be dropped. In particular, we discuss backpressure, and show why its important benefits are well worth its cost.

2. Overview of ATLAS I

ATLAS I is a general-purpose, single-chip, gigabit ATM switch with advanced architectural features [KaSV96]. It is intended for use in high-throughput and low-latency systems ranging from wide area (WAN) to local (LAN) and system (SAN) area networking, supporting a mixture of services in a range of applications, from telecom to multimedia and multiprocessor NOW (networks of workstations). Figure 1 presents an overview of ATLAS I.

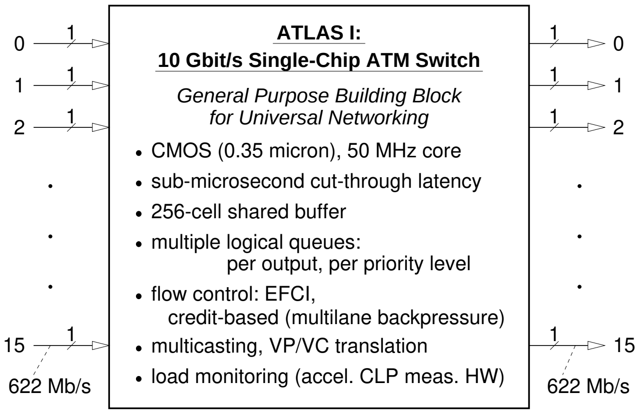

Figure 1: ATLAS I chip overview

The most distinctive and innovative feature of ATLAS I is its (optional) provision of credit-based flow control (backpressure), using a protocol that resembles QFC [QFC95] but is adapted to hardware implementation. A cell in a backpressured service class (priority level) can only depart if it acquires both a buffer-pool credit for its outgoing link and a flow-group credit for its connection. A flow group is a set of connections that are flow-controlled together; ATLAS I supports up to 32 thousand flow groups (2048 per link). Credits are transmitted on the links using 1 control and 3 data characters for each of them. Credit-based flow control is useful in building switching fabrics with internal backpressure that provide the high performance of output queueing at the low cost of input buffering [KaSS97], and in making ATM sub-networks that never drop cells of data connections while fully and fairly utilizing all the available transmission capacity, so as to provide the lowest possible message delivery time. Our evaluation of the credit protocol has shown its superior performance, especially in isolating well-behaved connections from bursty and hot-spot traffic [KaSS98].

In the rest of this paper, when breaking down the chip cost by function, we use the following list of functions provided by ATLAS I. These can also be seen in the floorplan of the chip, in figure 2.

1. **Buffer** – *Cell Buffer Memory and Switching:* Internally, ATLAS I operates as a crossbar, with ATM cells buffered in a memory of capacity 256 cells (13.5 KBytes) which is shared among all links. The cell buffer is implemented as a 28-stage pipelined memory [KaVE95] (number of stages = 53-byte ATM cell + 2-byte routing tag / 2 bytes per stage). In the floorplan (figure 2), one can see the 28 SRAM's, one per pipe stage, plus 2 additional SRAM's to record the incoming link and flow group.
2. **Queue Mgt.** – *Output Queue Pointer Management:* ATLAS I implements three levels of priority, each level having its own queues. Fifty one logical output queues (3 times 16 outputs plus 1 management port) and 3 multicast queues are maintained in the shared buffer. These queues are implemented as linked lists of cell buffer pointers.
3. **Rt'ng/Tran.** – *Cell Header Processing, Routing, VP/VC Translation:* ATLAS I can be configured to operate on normal 53-byte ATM cells, or on 55-byte packets consisting of a 2-byte routing tag followed by a 53-byte ATM cell. The chip contains a routing table of size 8K entries and a VP/VC translation table of size 4K entries; these tables are shared among all links. Each routing table entry specifies a service class (priority level) and an outgoing link bit mask (multicast support). Each translation table entry specifies a 28-bit new-VP/VC value. A 40-entry content-addressable memory identifies which connection group an incoming cell belongs to, using pattern-matching on the incoming link-ID (5 bits) and cell header (32 bits). For each connection group, different sub-fields of the header or routing tag can be used to index into sub-tables of the routing and translation tables. Header processing includes (figure 2): two cell delay blocks, which synchronize the cell buffer pipeline to the header processing pipeline; the header serialization block, which queues headers in arrival order (order is important when using link bundling –see below); and the header generation block, which assembles 5-byte headers from a 2-byte datapath.
4. **Sched. Cnt.** – *Scheduling and Population Counts:* A scheduler circuit controls the access to the shared cell buffer by all outgoing and incoming links, using time multiplexing. It bases its decisions on the status (empty or not) and service class of the output queues, the existence of multicast cells, and

link bundling. Link bundling allows the chip to be also configured as 8×8 switch at 1.25 Gbps/link, or 4×4 at 2.5 Gbps/link, etc. There are also counters to monitor and restrict the cell buffer occupancy by service class and by link; these are useful in implementing various quality-of-service policies.

5. **Ctrl. Mgt.** – *Chip Control and Management:* ATLAS I has a parallel interface to a generic microprocessor bus, which is used to set and monitor the internal chip state, and to inject and extract cells to and from the network. Alternatively, chip state can be monitored and set through the network links themselves; in this latter case, no local microprocessor is needed, and the chip is initialized through a Serial ROM port. Inside ATLAS I, these three mechanisms have access to a bus that runs throughout the switch and can read and write almost all registers, memories, and chip state.

6. **Credits** – *Credit-Based Flow Control (Backpressure) Support* is a composite function split in several blocks of the chip: it includes circuits for serializing incoming credits, queueing outgoing credits, storing credits per flow group (Credit Tables), searching cells by flow group on credit arrival, and it accounts for extra ports and operations in the queue management memories. For blocks that would anyway exist in the switch (even without credit flow control), and whose complexity increases in order to support credits (such as queue management), we have charged the extra cost to the "credits" function.

7. **Load Mon.** – *Accelerated CLP Measurement for Load Monitoring:* ATLAS I includes hardware support for the accelerated measurement of the cell loss probability (CLP) of the (non-backpressured) traffic, according to the algorithm described in [Cour95]; this is useful for real-time network-management decision making. The hardware measures the cell loss rate of an emulated set of buffers, whose size is smaller than the real chip buffer size. Based on such measurements over short periods of time, software uses extrapolation to compute the CLP of the actual traffic; this CLP is often so low that much longer observation periods would be needed, if it were to be measured directly.

8. **Elastic I/F** – *Elastic Buffers and I/O Link Interfaces:* The ATLAS I links run ATM on top of IEEE Std. 1355 "HIC/HS" as physical layer. HIC was preferred over SONET because of simpler circuitry, lower latency, and the capability to encode (unbundled) credits. The chip core (switch) and

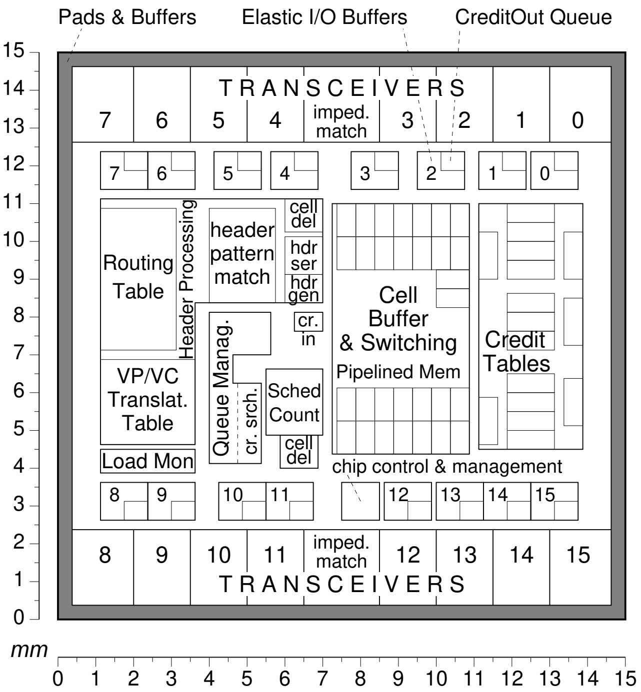

Figure 2: Floorplan of the ATLAS I chip

the serial link transceivers operate in different clock domains, with datapaths of different width. Thus, we use elastic buffers to handle frequency differences and provide synchronization. Additionally, this function contains the logic for the 8b/12b encoding scheme of HIC/HS, control character recognition, and the protocol procedures for link start-up, error detection, and link resetting.

9. *Link Transceivers and Pad Frame:* The GigaBaud serial link transceivers ("STRINGS" macrocell) are provided by BULL, France [MCLN93]. BULL also provides the pad frame. In the rest of the paper we differentiate clearly between the chip core that we designed at FORTH, and this chip periphery that is supplied by BULL.

3. Implementation and Cost

ATLAS I will be fabricated in a 0.35μm CMOS technology with five metal layers, by ST Microelectronics, in Crolles, France. The chip size is determined by the pre-existing pad frame: it is $15 \times 15\ mm^2$, as seen in figure 2. The core of the chip operates with a clock of about 50 MHz, which is the frequency required to support the 622 Mbit/s rate over 16-bit datapaths. This is a conservative clock frequency for the 0.35-micron technology, thus simplifying our logic partitioning and pipelining task; one access to an on-chip memory plus several levels of combinatorial logic fit within a clock cycle with relative ease. At the time of this writing (July 1998), we are connecting the pad frame to the ATLAS I core and performing global LVS checks for immediate tape-out for fabrication.

3.1 Design Flow and Design Cost

The majority of the ATLAS I blocks are semi-custom designs, using cells from a standard cell library. Models for all chip blocks were initially developed in Verilog, and simulated for correct function. As a second stage, these models were elaborated to structural Verilog code, where low level hardware elements, such as registers, counters, and adders, were instantiated. This latter model was a detailed description of the physical implementation and was ready for synthesis with Synopsys, after minor adjustments. We set our target clock to 10 ns (half cycle) for the synthesized design, allowing 2 ns clock skew, and 8 ns for signal propagations and interconnect delays. Since the setup time for library registers is 0.5 to 0.7 ns, the usable time of a clock cycle for memory access and combinatorial logic was 7.5 ns.

The synthesis was performed hierarchically, using the natural block boundaries that are defined both by functionality and by the pipeline partitioning. These boundaries dictate the general floorplan of the ATLAS I switch core. The hierarchical compilation permits a hierarchical placement of the cells in the larger blocks, allowing short interconnect delays. At the top level of the hierarchy, 15 blocks were defined, and placement and routing was performed individually, for each of them. Within each of these 15 blocks, the hierarchy is removed after placement and before routing; this is to allow for global clock-tree generation, and in order to minimize the rather lengthy setup time of the routing tools.

The back-end design flow was as follows. First, the top floorplan and the block floorplans were made, using the Cadence environment, after importing the EDIF netlist. Next, for each block, placement and routing is performed with Cell3, using the EDIF netlist and the block floorplan. Back-annotation follows, in order to calculate the clock skew and the overloads; these are fed back to Synopsys for in-place optimization and fix. A new round starts with Cell3 (ECO), to incrementally fix the placement and routing, and so on until no more changes are needed. Final verification (DRC, LVS) is performed using Dracula.

Table 1: Design Cost of ATLAS I Core

Function:	Verilog K lines	Synth. hours	Human Effort p-yrs	Human Effort %
Buffer	1.0	5	1.0	7
Queue Mgt.	1.4	2	2.3	15
Rt'ng/Tran.	2.1	12	2.2	15
Sched. Cnt.	1.5	18	2.0	13
Ctrl. Mgt.	.7	6	1.0	7
Credits	1.0	7	3.0	20
Load Mon.	.4	24	.5	3
Elastic I/F	3.6	22	2.0	13
Miscellany	1.8	4	1.0	7
Total	13.5	100	15.	100%

One ATLAS I block was laid-out in full-custom; it performs key functions for queue management and credit-based flow control, and was described in [Korn97]. The functions required are multiport (3 and 4 port) SRAMs, content addressable memories (CAMs), and memories with simultaneous read and modify accesses, which can not be generated automatically by a silicon compiler. Together with their peripheral circuitry, they were laid-out using the Cadence Virtuoso editor, and were verified for function and timing at the transistor level with ELDO simulations. At the logic level, we used IRSIM to verify this block.

The design cost of the main ATLAS I functions is shown in table 1. Code size, in thousands of Verilog lines, refers to the structural model that was used for synthesis. The code is very dense and compact, since it uses all the basic hardware components from a large internally developed library. The figures in table 1 do *not* include: the component library (2 Klines), the behavioral models of the full-custom blocks (3.2 Klines), and our test environment (3.4 Klines of Verilog and 5 Klines of PERL).

The second column of the table lists the time required to synthesize the Verilog code of the blocks into gate level descriptions, on a SUN Ultra-2 with a 200 MHz processor and 384 Mbytes of main memory. This main memory size was sufficient for the synthesis of each individual block, so the reported time was spent entirely on synthesis optimizations. The synthesis times are relatively small, but they are the result of an extensive trial and error process in which more than 1,000 hours of CPU time were spent to establish the best paramenters and balance the constraints among different blocks of logic. We found that the hierarchical compilation process does speed synthesis up considerably, and gives better results

Table 2: Logic Cost of ATLAS I Core

Function:	Gates K	Gates %	Flip-Flops K	Flip-Flops %	SRAM Kbits	SRAM %
Buffer	28	19	9.6	22	120	21
Queue Mgt.	12	8	.9	2	10	2
Rt'ng/Tran.	20	13	9.2	21	300	53
Sched. Cnt.	4	3	.7	2	1	0
Ctrl. Mgt.	2	1	.3	1	1	0
Credits	15	10	2.3	5	134	24
Load Mon.	4	3	1.6	3	0	0
Elastic I/F	65	43	19.4	44	0	0
Total	150	100%	44.0	100%	566	100%

Table 3: Area Cost of ATLAS I

Function:	Logic mm^2	SRAM mm^2	Routing mm^2	Total mm^2	Total %
Buffer	5.1	8.4	9.5	23.0	10
Queue Mgt.	1.4	1.0	.7	3.1	1
Rt'ng/Tran.	5.6	11.7	5.6	22.9	10
Sched. Cnt.	.5	.9	.9	2.3	1
Ctrl. Mgt.	.2	.4	.2	.8	0
Credits	2.2	14.6	8.0	24.8	11
Load Mon.	1.0	.0	.4	1.4	1
Elastic I/F	10.7	.0	3.0	13.7	6
Total *Blocks*	26.7	37.0	28.3	92.	41
Glob. Rout.			54.	54.	24
Total *Core*			82.	146.	65
Transceivers				57.	25
Pads, Driv.				22.	10
Total *Chip*				225.	100

Table 4: Power Dissipation of ATLAS I

Function:	Logic *mW*	SRAM *mW*	Total *W*	Total %
Buffer	1,040	1,360	2.40	27
Queue Mgt.	60	170	.23	3
Rt'ng/Tran.	410	250	.66	7
Sched. Cnt.	40	120	.16	2
Ctrl. Mgt.	10	2	.01	0
Credits	190	180	.37	4
Load Mon.	100	0	.10	1
Elastic I/F	280	0	.28	3
Total *Core*	2,130	2,080	4.2	47
Transceivers			4.0	44
Pads, Driv.			.8	9
Total *Chip*			9.0	100%

than a flat one, but only after a long experimentation phase and considerable manual labor for tuning. We also found that the automatic hierarchical compilation scripts were only successful for the parts of the switch that were both small and relatively simple.

The last two columns of table 1 show the approximate human effort spent for the entire ATLAS I core design. This includes both full-custom and semi-custom design, synthesis, simulation, timing verification, placement, routing, back-annotation. The total effort was 15 person-years, or an average of 6 full-time-equivalent persons working for 2.5 years (early 1996 through mid 1998). The figures in table 1 do not include the architectural design of the switch, which went on in 1995.

3.2 *Silicon Cost and Power Dissipation*

Table 2 shows the ATLAS I cost in terms of logic elements. Almost half of the gates and flip-flops are in the link interfaces and elastic buffers, for a number of reasons: *(i)* there are 16 copies of this block; *(ii)* link start-up, CRC calculation, and 8b/12b encoding account for considerable logic; *(iii)* the elastic buffers were designed using discrete flip-flops and gates rather than two-port SRAM, because the capacity needed was smaller than the smallest available compiled SRAM. The cell buffer memory has a lot of flip-flops (22%) because of its input latches ((16+1) links × (53+2) bytes/cell) and pipeline registers, and has a lot of gates (19%) because of its output drivers (55 byte × 17 port crossbar with multicasting support, built as a tree of gates for speed). In terms of SRAM, the buffer memory contains a significant amount (21%) and could even contain more (section 5.1); the routing and translation tables ended up consuming too many (half) of the chip's SRAM bits, which is undesirable (section 5.2). The SRAM for credit support would normally be quite less than it is: for the credit tables we needed 16 memories organized as 2K × 1, while the narrowest compiled SRAM available for the job was 2K × 2. This and other issues related to credit support are discussed in sections 4 and 6.

Table 3 breaks down the silicon area cost of ATLAS I by function. The transceivers account for 1/4 of the chip, and the core occupies 2/3 of the chip. One third of the core is taken up by (mostly uncompacted) global routing and power rails; this area includes our "safety margin" –the external dimensions of the chip were fixed a priori. The core is taken up by three –almost equal-sized– functions:

buffer, routing/translation, credits. The link interfaces & elastic buffers form a fourth, smaller function, while all other functions occupy very small area.

Table 4 shows the power dissipation of the ATLAS I blocks. Off-chip communication consumes half of the 9 Watt chip power, with the Gigabaud transceivers consuming the large majority of that. In the core, more than half of the power goes to the buffer memory which also performs the switching. This is reasonable, in view of the fact that, in each clock cycle, 480 bits are read or written to/from the memories in this block. The other functions consume much smaller amount of power, each.

3.3 ATM Cell Cut-Through Delay

When an ATM cell arrives at an idle ATLAS I switch, the cell is forwarded to its intended outgoing link using *cut-through:* the head of the cell starts going out before the tail of the cell has yet come in. The minimum cut-through delay of ATLAS I, measured from the output of the incoming transceiver to the input of the outgoing transceiver, is approximately 30 clock cycles, i.e. 600 ns. This delay is analyzed as follows:

elastic buffer (incoming)	10 cycles
header proc., routing/transl.	10 cycles
enqueue, schedule-out, dequeue	5 cycles
buffer memory read & switch	3 cycles
elastic buffer (outgoing)	2 cycle
Total Cut-Through Delay	30 cycles

This delay is longer than originally intended, and most of it is due to secondary effects rather than intrinsic, core functions; in the next sections we discussed how it could be reduced. Elastic buffer delay is discussed in sections 5.3 and 6.5. Header processing and routing/translation delay is broken down as follows: 3 cycles for header generation (assemble a 5-byte header from a 2-byte/cycle datapath); 2 cycles for the header serializer (to maintain arrival order, which is important for link bundling); 2 cycles for header pattern matching (CAM); and 3 cycles for routing/translation. These header processing delays are discussed in sections 5.2 and 5.4.

4. Full-Custom versus Semi-Custom

A high-throughput switch with an advanced architecture, like ATLAS I, requires several sophisticated circuits. These can be implemented in full-custom VLSI or in semi-custom logic (standard cells and compiled SRAM). Full-custom reduces the area cost −in some cases dramatically− and usually power consumption too, at the expense of human effort. We did not feel that the design effort for full-custom blocks was excessive; on the other hand, we did opt for semi-custom implementation for quite a few of the more demanding blocks. Here follows a list of sophisticated functions, with comments on how better RAM compilation would help, or what we saved or could have saved by going to full-custom.

- *Compiled SRAM size.* As mentioned in section 3.2, for the credit tables we needed 2K × 1 memories, while only 2K × 2 was available. We would save 5 mm^2 if the SRAM compiler could generate precisely the needed memory size.
- *Precise write-control in compiled SRAM.* Actually, instead of the 16 credit tables of size 2K × 1 each, a single 2K × 16 memory would suffice, provided we could selectively write individual bits rather than entire 16-bit words −a simple option that the SRAM compiler could provide.
- *Multi-port compiled SRAM.* Only single- and dual-ported compiled SRAM were available to us. In ATLAS I, the small (54 × 17) four-ported memory that keeps the head and tail pointers for the output queues was designed in full-custom; it takes up 0.2 mm^2 and dissipates less than 30 mW. If designed using standard cells, the cost doubles both in area and in power dissipation.
- *Content-addressable memory (CAM).* The header pattern match block, in header processing, is a 40 entry × 37 bits one-port CAM, which addresses a 40 × 42 SRAM. It was implemented in semi-custom, with discrete flip-flops and gates, and occupies 4 mm^2. In queue/credit management, a 256 × (29+17) *three-port* CAM occupies just 1.6 mm^2 [Korn97]. The difference speaks for itself.
- *Pipelined memory for buffering and switching.* Savings in area and power dissipation would also come from designing the cell buffer memory and its peripheral circuits as a full custom block. Extrapolating from the results of our previous pipelined memory design in 1μm technology [Efth95], we estimate that the cell buffer of ATLAS I would occupy approximately 13 mm^2 in full-custom, instead of 23 mm^2 now, in semi-custom. The main reason for this improvement is that manual design exploits the regularity of the peripheral circuits (input and output latches, drivers, and paths) much better than the automatic tools for placement and routing do.

5. Architecture Evaluation 1: General

We proceed now to evaluate the overall architecture of ATLAS I, in view of the implementation cost metrics (section 3) and of the technology characteristics (section 4). This section discusses all other architecture aspects except for credit-based flow control (backpressure), while section 6 is dedicated to that latter topic −the central characteristic of ATLAS I.

5.1 Cell Buffer Size

Cell buffering is a major function of a switch; to improve network performance, we want as much buffer capacity as possible. When originally designing ATLAS I (for a 0.5µm technology), we conservatively specified a buffer size of 256 ATM cells. At that time, we were mostly concerned about queue management: larger cell buffer sizes would lead to a larger CAM and priority enforcer. Now that the design is finished, we know that queue management (including CAM) could handle more cells without jeopardizing the clock cycle time [Korn97], and the area penalty on the current 1.6 mm^2 circuit would be minor for the overall chip.

In retrospect, it would have been desirable for the ATLAS I cell buffer capacity to be 512 or 1024 ATM cells i.e. two to four times larger than it is now. Such a capacity increase would lead to less than twice or four times the buffer SRAM area −currently 8.4 mm^2 (SRAM area is a sublinear function of capacity), while the peripheral circuits of the buffer would not be significantly affected. The resulting area increase of roughly 8 or 20 mm^2 could be absorbed by reducing the header processing area (section 5.2), or by better compaction of the 54 mm^2 global routing area, or with a full-custom pipelined memory implementation (section 4). The cell buffer consumes considerable power −one fourth of the entire chip− but this is due mostly to the peripheral circuits and to the number of bits read/written per cycle (30 memories × 16 bits/memory = 480 bits), so it would not increase much when increasing just the number of words per memory.

5.2 VP/VC Translation

Header processing, routing, and header Translation were designed to provide considerable flexibility, but their implementation results indicate a large cost: they account for half of the chip's SRAM bits, and for as much area as the cell buffering & switching function. The main cost of this function comes from the routing table, the translation table, and the header pattern matching block (figure 2).

ATLAS I provides VP/VC translation because the standards specify that an ATM switch must provide that function. On the other hand, our analysis of network addressing mechanisms (see [Man97]) indicates that translation is not necessary in small ATM (sub-) networks, while large ATM networks need large VP/VC translation tables; furthermore, large translation tables can*not* be synthesized as a cascade of multiple small tables in successive stages of switching fabrics. Therefore, the −necessarily small− translation table in ATLAS I is not very useful: large networks need off-chip memory to implement translation, while small networks do not need translation.

VP/VC translation costs in multiple respects: *(i)* the translation table occupies about 1/4 of the header processing area; *(ii)* header pattern matching occupies another 1/4 of the block's area −the function of this 40-entry CAM is to provide a high degree of configurability regarding the ATM standard VP/VC switching modes; *(iii)* the large number of sequential header processing steps resulted in an excessive number of clock cycles being spent for these functions (section 3.3) −the 5 clock cycles of latency that are due to routing and translation would have been reduced to 2 cycles if translation were not performed and less configurability options were provided. We conclude from the above that, for someone who wanted to simplify ATLAS I or needed a larger cell buffer and did not have the silicon area for that, the first priority would be to eliminate VP/VC translation from the chip functions.

5.3 Clock Domains and Number of Links

There is an interesting relation between the number of links and the ATM cell size, which also involves the link rate, the datapath width, and the frequencies of the various clocks. It essentially boils down to the fact that when the number of links is not approximately an integer submultiple of the cell size, the cell cut-through delay and the elastic buffer size increase. To explain it, we start with the switch control, which is time-shared among all links [KaSV96], and the cell buffer operation as a pipelined memory [KaVE95]. These circuits have the ability to initiate the processing of one new outgoing or incoming cell in each clock cycle. Since there are 16 output and 16 input links, plus a lower-rate internal switch port for control and management, ATLAS I has 33 clock cycles per cell-time. Assume, for the moment, that the link rate is 53 bytes per cell-time (in reality, there are HIC/HS control characters as well on the link, so the actual rate is different). If the 53 bytes per link per cell-time were to be transferred to/from the buffer memory at a rate equal to the link rate, this rate would have to be 53 bytes / 33 clock cycles, or $53 \times 8/33 = 12.85$ bits per clock cycle.

To achieve (almost) this rate we would have to use a *13-bit* datapath; besides being unconventional, this would complicate the conversion between the byte-oriented HIC/HS link protocol and the internal datapath. Instead, we decided to use 16-bit internal datapaths. This, however, means that the pipelined memory "absorbs" an incoming cell in $53/2 \approx 27$ clock cycles, or 27/33 = 82% of a cell-time. The pipelined memory −as well as cut-through operation− need an uninterrupted flow of incoming data, for each cell. It follows that the pipelined memory can only start receiving an incoming cell 33-27 = 6 clock cycles after cell arrival on the link. This initial delay of 6 cycles is provided by the incoming elastic buffer (section 3.3); the actual number of cycles varies according to the 53- or 55-bytes-per-cell mode (section 2) and according to whether HIC/HS backpressure (single-lane) and/or ATLAS backpressure (multilane) are enabled or not.

There are two ways to fix the increase to the cut-through delay from the above effect. One method is to overlap this delay with header processing. We intended to do this but then run out of time during chip design and decided to drop this optimization. Even if input elastic buffer delay is overlapped with header processing, it is still interesting to reduce the former, given that there are methods to reduce the latter, too (sections 5.2 and 5.4).

The input elastic buffer delay can (almost) be eliminated by matching the switch core clock to the link clock, which boils down to setting the number of links to an integer submultiple of the cell size. In our case, this would mean using a internal clock that has 27 or 28 cycles per cell-time, which would accommodate up to 13 or 14 links for the switch. This is entirely OK, except for the fact that some customers may be prejudiced against buying a 13×13 switch −at least as much as we were prejudiced against using 13-bit datapaths....

Synchronizing the core clock frequency to (half of) the (outgoing) link clock would also simplify the outgoing elastic buffers: that interface would become synchronous, and the only reason to have a buffer there would then be to provide HIC/HS-style (single-lane) backpressure. Notice that on the input side we need to accept different clock domains, because inputs originate from different sources, which are normally not synchronized to each other. Also notice that, in order to eliminate the input elastic buffer delay, one should not allow ATLAS credits to interrupt cells, as further discussed in section 6.5.

5.4 The Cost of Generality

ATLAS I is a general-purpose building block for universal networking. Consequently, like all general-purpose systems, it supports several different features and configuration options. While the commercial advantage of such an architecture is very big, there is nevertheless a cost, which we are obliged to point out. Here is a list of features and options, where each one of them individually incurs a small incremental cost, but all of them collectively amount to non-trivial complexity.

- *Switching fabric and stand-alone.* ATLAS I is intended for system (SAN), local (LAN), and wide area networks (WAN). When connected directly to other ATLAS I chips, it can form either switching fabrics [KaSS97] or LAN's. In a switching fabric, the chip is shielded from the external ATM network, while in LAN's the chip must behave like an ATM switch. This has implications for a number of features and options listed below.
- *VP/VC translation;* many users may want it in LAN's, but it is not useful in switching fabrics (section 5.2).
- *Header options:* routing and translation can be based on various subfield combinations of the cell header; useful in ATM networks but not in switching fabrics with regular topology.
- *Routing table size;* switching fabrics with regular topology only need a small-size routing table, while ATM LAN's need larger routing tables.
- *53/55 byte cell format;* the cell format with prepended routing tag (55 bytes) is useful in switching fabrics.
- *Link bundling* (allowing pairs of links to behave as one link of 1.25 Gbps, etc.). High-end switching fabrics prefer e.g. a 4×4 ATLAS I configuration, at 2.5 Gbps. ATM LAN's and line multiplexors/demultiplexors need many, lower-rate (622 Mbps) links. Link bundling introduces the need for the header serializer to maintain arrival order, which accounts for about 1.5 of the clock cycles of cut-through delay (section 3.3).

- *Backpressure options.* HIC/HS (single-lane) backpressure and ATLAS (multilane) backpressure can each be enabled or disabled separately; elastic buffer and cell counter operation are considerably complicated by the special cases that arise.
- *Number of flow groups:* switching fabrics need fewer of these; see section 6.2.
- *Dedicated links for cells and credits* could be used in high-end switching fabrics, but not in LAN's; see section 6.5.

6. Architecture Evaluation 2: Backpressure

Credit-based flow control (multilane backpressure) is the most distinctive and innovative feature of ATLAS I. It is useful in building switching fabrics with internal backpressure [KaSS97] and ATM sub-networks that utilize fully and fairly all the available transmission capacity and provide lowest message delivery time [KaSS98]. This section first evaluates various aspects of the backpressure mechanism used in ATLAS I and discusses alternative designs (sections 6.1 through 6.5). Then, in section 6.6, we discuss the overall cost versus benefit of backpressure, and conclude that this is a highly advantageous architecture.

6.1 Straightforward Cost Reduction

As already explained in section 3.2, support for credit-based flow control would normally cost 21 mm^2, rather than 25 mm^2, if compiled memories organized "by 1" (rather than "by 2") were available. A further reduction would result if compiled SRAM with precise write-control were available (section 4).

6.2 Number of Flow Groups

The largest contribution to the area cost of the credit function comes from the credit tables (figure 2). Their size is proportional to the number of flow groups supported. Can this number be reduced? A flow group, in ATLAS I, is a set of connections that are flow controlled together; credits are at the granularity of flow groups. Flow groups were introduced in ATLAS I because it would be infeasible and undesirable for credits to operate on individual connections (VC's). Flow groups allow for a hierarchical organization of flow control: all connections with a common path or a common destination are placed in the same flow group over that common path. In an ATM network, there can be up to 256 million connections going through a link; we felt that providing 2 thousand flow groups per link was as good a compromise as we could do in ATLAS I. However, the situation with switching fabrics is different.

In a switching fabric made of ATLAS I chips, like e.g. inside a large ATM switch box for WAN's, flow groups are normally defined to consist of all connections going to a common output port of the fabric [KaSS97]. Figure 3 illustrates this, using a banyan fabric and one color per flow group. When using ATLAS I in such a switching fabric, we can have up to 2048 flow groups per outgoing link, hence the fabric can have up to $16 \times 2048 = 32$ thousand output ports. High-speed (622 Mbps) switching fabrics of practical interest, however, have a much smaller size –e.g. up to 1024 links, suggesting that it would suffice for ATLAS I to support far fewer flow groups in such environments. Another point related to the inefficient use of credit tables in switching fabrics is the following: when links are bundled together, only the credit table of the "primary" link in the bundle is usable. In switching fabrics, ATLAS I configurations like 4×4 @ 2.5 Gbps are preferable; then, 75% of the credit tables remain unused. In conclusion, if the designer needs to reduce the area cost of credits to less than the 9 or 10 percent of the chip area achieved as in section 6.1, then he or she should consider supporting less flow groups; in switching fabric applications, this would not hurt.

6.3 Per-Flow Queue Table

Conceptually, a multilane-backpressure switch needs one queue of cells per flow group, so that cells in different flow groups can bypass each other. With 32 thousand flow groups, in ATLAS I, having that many queues is problematic. The solution used is to only maintain information about the *non-empty* per-flow queues [Korn97]; since there are at most 256 cells in the buffer, at most 256 of the per-flow queues can be non-empty. There are two disadvantages with this organization: *(i)* (full-custom) content-addressable memory is needed to find the queue for a given flow group; and *(ii)* at most one cell per (incoming) flow group is allowed in ATLAS I, in order to keep the overall complexity manageable by simplifying the queue structure. The restriction of at most one cell per flow group is acceptable in switching fabrics, where round-trip times are on the order of 1 cell time, but it may be a severe restriction in LAN environments.

If one wanted to avoid full-custom circuits altogether, or to support multiple cells per flow group, or both, and if fewer flow groups suffice (section 6.2), then an alternative queue organization can be considered[2], which we briefly outline here. The overall rate of queue operations is comparable to ATLAS I, while the CAM is replaced by a table of per-flow queues; each of these is a queue of cells. Each arriving cell is enqueued in its flow-group-queue. The output queues become circular lists of flow groups, instead of queues of cells as in ATLAS I. Each cell or credit arrival can result in its flow group being inserted in an output queue (list). The output scheduler walks through the output lists, and for each flow group in them it dequeues and transmits a cell from the corresponding per-flow queue. Each cell departure can result in its flow group being removed from the output queue (list) that it belonged to.

6.4 Credit Output Queues

Besides the credit tables, whose cost reduction was discussed in section 6.2, another large contribution to the area cost of the credit function comes from the credit output queues; these are physically located in the I/O link interfaces (figure 2). ATLAS I includes 16 credit output queues –one per link. Each of them has 256 entries, as determined by a worst case scenario: 256 cells having arrived from a single input port may depart in consecutive cell times from all outgoing ports, generating a burst of credits to be sent back to the upstream originating switch. The credits are produced at a rate of 16 per cell time, and consumed when they are sent out of the switch, at a rate of 2 per cell time (the rate at which the upstream ATLAS chip can process arriving credits).

Obviously, the above worst case scenario has a very low probability of occurrence. One can reduce the size of the credit output queues, and simply block the departure of cells on a link while departure of the head-of-line cell in that ready-queue would cause a credit output queue to overflow. Determining the smallest size of the credit output queues that would keep the above loss of performance down to negligible levels requires a careful study and evaluation that we have not yet had the resources to perform.

An alternative method to reduce the cost of the credit output queues would be to store all of them (in the form of linked-list queues) in a single RAM of size 256, shared by all links. We did not implement this organization because we could not afford its design complexity. A hybrid method would be to use small queues per output, plus one shared overflow queue.

[2] multicasting support with backpressure would be problematic, though; ATLAS I uses its CAM to handle that function too.

6.5 Dedicated Links for Cells and Credits

In ATLAS I, credits and cells are transmitted over the same links, in a time-multiplexed fashion. Given the quite smaller bandwidth required for credit transmission, this is a wise decision for LAN's with unbundled or lightly bundled links. Under heavy bundling or in switching fabrics, though, one can consider having separate, dedicated links for cells and for credits (credit links in switching fabrics can also use normal pins instead of HIC/HS transmission). Such a partitioning of functionality would save silicon area and design complexity.

In the current design, all sixteen links are able to handle credits and cells; also, access to shared switch blocks is under arbitration using serializers, and all memories and logic is sized for the sixteen links. In an alternative design, one of the links in a bundle (or separate pins) would be used exclusively for carrying credits, while the rest of the links would only carry cells. Under this configuration, we save the credit datapath on links carrying cells, and the cell datapath on links dedicated for credits. The design is also simplified because cell-credit multiplexing and demultiplexing is eliminated. Finally, the input elastic buffer delay (section 5.3) is also reduced, for the following reason. In ATLAS I, up to one credit is allowed to "interrupt" a cell body and be interleaved in its bytes, on a link; this is in order to reduce the cell-credit round-trip time, so as to alleviate the effects of the one-cell-per-flow-group restriction. On the receive side, such an interleaved credit causes a "hiccup" (a bubble) in the cell reception flow. This hiccup must be removed from the stream of data going to the pipelined memory, by the input elastic buffer; to provide the necessary slack for such removal, the initial delay in that buffer must be increased by one credit transmission time.

6.6 Backpressure Cost versus Benefit

With or without the optimizations discussed so far, backpressure support in ATLAS I costs 5 to 10 percent of the chip area, 4% or less of the chip power dissipation, and 20% of the core design effort. Credits also account for 4/54 = 7.5% of the link throughput; this figure would normally be 3/54 = 5.5% of the link throughput (3 bytes/credit, 54 bytes/cell), if it were not for the "BULLIT" (HIC/HS interface) chip which handles incorrectly transmissions consisting of an odd number of bytes. What are the benefits from backpressure support? Are they worth the above costs?

We will answer these questions for a switching fabric, because the tradeoffs are most easily quantifiable there. The conclusions extend to other networks too. We will discuss fabrics for making $N \times N$ composite switches, where the number of ports, N, is so large that it is uneconomical to build them using a centrally-scheduled crossbar (the central scheduler in a crossbar implicitly implements backpressure, anyway).

Figure 3 illustrates a model switching fabric with internal backpressure; the flow groups are as discussed in section 6.2, above; see [KaSS97] for details. Backpressure effectively pushes the bulk of the queued cells back to the input interfaces. The head cells of the queues, however, are still close to their respective output ports. The fabric operates like an input-buffered switch (a switch with advanced input queueing), where the ATLAS chips implement the arbitration and scheduling function in a distributed, pipelined fashion. Thus, backpressure allows the efficient use of the switch buffers: only the cells that are necessary for quick response are allowed to proceed to those buffers. At the same time, bulky queues are restricted to the input interfaces only, where they can be efficiently implemented using off-chip DRAM memory.

If backpressure is not used inside the switching fabric, the designer has to ensure that not too many cells get dropped due to buffer overflows in the fabric; there are two ways to achieve that. First, a *large buffer memory* can be provided in each switching element in the fabric. To be competitive with the backpressure design, the fabric must be able to operate under heavy load. Then, however, due to traffic burstiness and load fluctuation, the buffers in the switching elements must be very large, thus requiring off-chip memory. Such off-chip buffer per switching element may eliminate the need for the input interfaces, but the buffers introduced at the switching elements are log N times more than the buffers eliminated. At the same time, the off-chip I/O throughput of each switching element is doubled, due to the added off-chip buffer memory. Off-chip communication, in ATLAS I, costs 30% of the chip area and 45% of the chip power (transceivers + elastic interfaces). Doubling *that* part of the chip in order to eliminate backpressure, which costs less than 10% of the chip area and 4% of the chip power, is a *bad* idea.

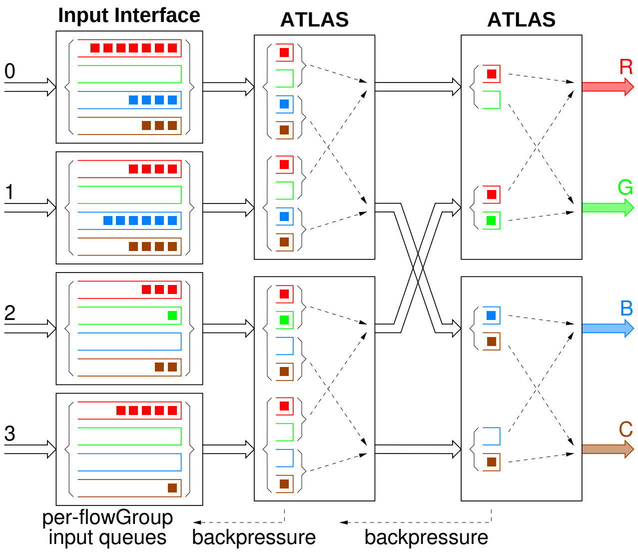

Figure 3: Switching fabric with internal backpressure

The second method to architect a switching fabric without backpressure and without excessive cell drop rate is to provide *internal speed-up:* the links in the switching fabric can have s times higher capacity relative to the external links, where s is the speed-up factor. Such a faster fabric, when subject to the same "heavy" load as the original fabric, is still lightly loaded relative to its own increased capacity. In this way, the switching elements can have relatively small on-chip buffers, and still achieve a low cell drop rate. The bulk of the queues now accumulate on the output side of the fabric, given that the external links cannot absorb traffic as fast as the fabric can deliver it. What speed-up factor s is required in order to achieve low enough cell drop rates with on-chip buffers? Under a uniform load with non-bursty traffic, a speed-up factor modestly larger than unit would probably suffice. Real traffic, however, is bursty and is not uniformly destined. Even speed-up factors of 1.5 or 2.0 would not be able to ensure low drop rate under large traffic bursts or non-uniformity.

Concerning the cost, a speed-up factor of s is equivalent to multiplying the aggregate link throughput of each switching element by s. In an ATLAS-style chip, this would multiply by s the transceiver and elastic interface cost, and would also multiply by almost s the buffer memory cost (a lot of the buffer cost is due to switching throughput rather than memory capacity). Thus, a speed-up factor of s would roughly cost an extra $(s - 1.05) \cdot 40\%$ in chip area and $(s - 1.05) \cdot 75\%$ in chip power (the .05 in the 1.05 is there to account for the link throughput gained by not having to transmit credits). Obviously, for any realistic speed-up factor s, the added cost of internal speed-up is overwhelmingly larger than the backpressure cost of 10% in chip area and 4% in chip power.

Conclusions

We analyzed the implementation cost of ATLAS I, a 10 Gbit/s single-chip ATM switch with backpressure. We showed that backpressure costs very little in comparison to what it offers: alternative architectures have to pay much more in order to achieve comparable performance (section 6.6). We have also evaluated a number of other architectural characteristics of ATLAS I:

- cell buffer size should be increased (section 5.1);
- VP/VC translation should probably be dropped (section 5.2);
- more flexibility in compiled SRAM's would greatly help (sections 4, 6.1);
- full-custom CAM and pipelined memory can save a lot of space (section 4);
- a switch chip targeted at building switching fabrics costs less than a general-purpose switch chip, in a number of areas (sections 5.4, 6.2, 6.5);
- adjusting the number of links to the cell size reduces the number of clock domains and the cell cut-through delay (section 5.3);
- to reduce the cut-through delay, overlap header processing with the input elastic buffer delay, or better yet try to (almost) eliminate both of them (sections 5.3, 5.4, 6.5).

Acknowledgements

ATLAS I is being developed within the "ASICCOM" project, funded by the Commission of the European Union, in the framework of the ACTS Programme. We also want to thank all the other members of the FORTH ASICCOM team who have worked for the design and implementation of ATLAS I, BULL for providing the GBaud transceivers and pad ring, ST Microelectronics for their extensive assistance with the design tools and libraries and for fabricating the chip, all the other members of the ASICCOM Consortium for their help and support, and Europractice and the University of Crete for many CAD tools.

References

[Cour95] C. Courcoubetis, G. Kesidis, A. Ridder, J. Walrand, R. Weber: "Admission Control and Routing in ATM Networks using Inferences from Measured Buffer Occupancy", *IEEE Trans. on Communications,* vol. 43, no. 4, April 1995, pp. 1778-1784.

[Efth95] A. Efthymiou: "Design, Implementation, and Testing of a 25 Gb/s Pipelined Memory Switch Buffer in Full-Custom CMOS", Technical Report FORTH-ICS/TR-143, *ICS, FORTH, Heraklio, Crete, Greece,* November 1995; http://archvlsi.ics.forth.gr/switches85-95/pipeMem_impl25Gbps.ps.gz

[KaSS97] M. Katevenis, D. Serpanos, E. Spyridakis: "Switching Fabrics with Internal Backpressure using the ATLAS I Single-Chip ATM Switch", *Proceedings of the GLOBECOM'97 Conference,* Phoenix, AZ USA, Nov. 1997, pp. 242-246; http://archvlsi.ics.forth.gr/atlasI/atlasI_globecom97.ps.gz

[KaSS98] M. Katevenis, D. Serpanos, E. Spyridakis: "Credit-Flow-Controlled ATM for MP Interconnection: the ATLAS I Single-Chip ATM Switch", *Proceedings of HPCA-4* (4th Int. Symposium on High-Performance Computer Architecture) Las Vegas, NV USA, Feb. 1998, pp. 47-56; http://archvlsi.ics.forth.gr/atlasI/atlasI_hpca98.ps.gz

[KaSV96] M. Katevenis, D. Serpanos, P. Vatsolaki: "ATLAS I: A General-Purpose, Single-Chip ATM Switch with Credit-Based Flow Control", *Proceedings of the Hot Interconnects IV Symposium,* Stanford Univ., CA, USA, Aug. 1996, pp. 63-73; http://archvlsi.ics.forth.gr/atlasI/atlasI_hoti96.ps.gz

[KaVE95] M. Katevenis, P. Vatsolaki, A. Efthymiou: "Pipelined Memory Shared Buffer for VLSI Switches", *Proc. of the ACM SIGCOMM '95 Conference,* Cambridge, MA USA, Aug. 1995, pp. 39-48; http://archvlsi.ics.forth.gr/switches85-95/pipeMem_sigcomm95.ps.gz

[Korn97] G. Kornaros, C. Kozyrakis, P. Vatsolaki, M. Katevenis: "Pipelined Multi-Queue Management in a VLSI ATM Switch Chip with Credit-Based Flow Control", *Proc. 17th Conference on Advanced Research in VLSI (ARVLSI'97),* Univ. of Michigan at Ann Arbor, MI USA, Sept. 1997, pp. 127-144; http://archvlsi.ics.forth.gr/atlasI/ atlasI_arvlsi97.ps.gz

[Man97] "ATLAS I Architecture Manual", *ICS-FORTH, Crete, Greece,* 1997; Internal ASICCOM working document; to be published on-line at a later time, under the directory: http://archvlsi.ics.forth.gr/atlasI/

[MCLN93] R. Marbot, A. Cofler, J-C. Lebihan, R. Nezamzadeh: "Integration of Multiple Bidirectional Point-to-Point Serial Links in the Gigabits per Second Range", *Hot Interconnects I Symposium,* Stanford Univ., CA, USA, Aug. 1993.

[QFC95] Quantum Flow Control Alliance: "Quantum Flow Control: A cell-relay protocol supporting an Available Bit Rate Service", version 2.0, July 1995; http://www.qfc.org