VC-level Flow Control and Shared Buffering in the *Telegraphos* Switch

Manolis Katevenis†, Panagiota Vatsolaki, Aristides Efthymiou†, and Manolis Stratakis

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

ABSTRACT: We present three implementations of a fixed-size-packet switch featuring shared buffering, virtual circuit (VC) flow control, and 3 clock cycle cut-through latency, for applications in gigabit LAN's, networks of workstations, multiprocessor networks, and WAN's. VC-level credit-based flow control is essential in data communications, to offer full throughput utilization while preventing throughput collapse under heavy, bursty, and hot-spot traffic. We describe in detail the implementation of VC flow control -- one of few such implementations in hardware. We present an FPGA-based prototype that was built and is now functional, a standard-cell ASIC CMOS single-chip implementation that is currently under fabrication, and a full-custom layout of the pipelined memory shared buffer, which is also under fabrication inside a test chip. VC flow control costs less than 15 % of the switch area in both the FPGA and the ASIC versions. Full-custom yields 4 times less area and 3 times higher speed for the buffer and crossbar, compared to the semi-custom implementation.

KEYWORDS: *VLSI switch, VLSI router, virtual circuit flow control, back pressure credit flow control, shared buffer, gigabit LAN, NOW, ATM.*

1. Introduction

This paper describes the *Telegraphos* switch implementation, with emphasis on its virtual-circuit, hop-by-hop, credit-based flow control, its shared buffering, and its low latency. We present *Telegraphos I,* a board-level FPGA-based prototype, *Telegraphos II,* a standard-cell CMOS ASIC implementation, *Telegraphos III,* a full-custom version of the shared buffer and switching fabric, and we give measurements on the percentage of the hardware cost (gates, area) that goes into each switch function.

Contemporary high-performance interconnects are based on point-to-point links, and switches are basic building blocks for them. In the *Telegraphos* project [Kate94], switches are used to make *networks of workstations (NOW)* that can provide multiprocessor performance [AnCP95]. Switches of the same architecture are also useful for building *parallel computer interconnection networks, gigabit LAN's, or WAN's* – with variable or fixed size (e.g. ATM) packets.

The *Telegraphos* project, at ICS-FORTH, also investigates network interfaces for efficient parallel processing on networks of workstations. The name of the project – from the Greek words τηλέ (remote) and γράφω (write) – comes from its shared address space architecture, in which remote memory writes are the basic operation. Remote writes, triggered by store instructions executed at user-level, allow message passing without system-call overhead, while the existing address translation mechanism ensures message protection. Other remote memory operations, including eager updates and hardware monitors, provide effective and low-cost support for virtual shared memory. The network interface architecture of *Telegraphos* is described elsewhere [Kate94].

Telegraphos networks use fixed, small size packets, virtual circuit routing (like ATM), shared buffering, and credit-based back-pressure flow control. The flow control signals refer to specific virtual circuits – not just indiscriminately to the entire link traffic; coupled with appropriate buffer allocation policies, this ensures tolerance to bursty and hot-spot traffic. These points are discussed in section 2. Section 3 presents the switch block diagram, explaining the implementation of VC flow control, and describing how the cut-through latency of 3 clock cycles is achieved. The switch prototypes that have been implemented are presented in section 4, and measurements of their cost are also given in that section.

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

2. Switch Architecture

Contemporary high-speed interconnects use multiple point-to-point links, in order to provide increased parallelism and avoid the disadvantages of shared physical media, i.e. driver turn-around delays and arbitration overhead. Switches (also called *routers*) are basic building blocks for such packet-switched interconnection networks. Their function is to make routing decisions and to forward packets to the proper outgoing link(s); since output contention may arise, switches also need to provide buffering.

The Telegraphos network uses *small, fixed size packets;* packets have a fixed size in order to simplify hardware design, and this size is small in order to reduce latency and fragmentation cost. The *Telegraphos I* packet size is 9 bytes, and the *Telegraphos II* size is 18 bytes; although not equal to the ATM cell size (53 bytes), nothing in principle differentiates the two, so readers should think of Telegraphos packets as essentially equivalent to *ATM cells.* There is also some analogy between Telegraphos packets and *flits* in wormhole routing [DaSe87]; under that viewpoint, each wormhole packet is analogous to a *burst* of Telegraphos packets or ATM cells traveling in one virtual circuit (VC). Telegraphos uses *static routing* in order to guarantee in-order delivery of packets from a given source to a given destination; in addition, it uses the ATM-style *virtual-circuit (VC) routing,* where each packet travels along a VC that was set up ahead of time; the VC identifier is translated in each switch.

2.1 Preventive Flow Control

There are two sources of packet loss in networks: errors due to electrical or optical noise, and buffer overflows due to output contention. For data communication, which needs reliable delivery, retransmission protocols are needed in order to cope with packet loss. Many link media in contemporary networks have such low error rates that retransmissions are very rare, and so they can be handled in software. On the other hand, packet loss due to contention may occur quite frequently in many modern computer networks when no flow control is employed – e.g. networks carrying multimedia or parallel or distributed application traffic. In high-throughput networks with small or medium packet size, such packet losses would be so frequent that retransmission protocols must be implemented in hardware in order to achieve good performance. However, in hardware, it is much simpler to implement a preventive flow control mechanism that *never drops packets,* than to implement retransmission protocols. Additionally, not transmitting packets that would be dropped by the receiver preserves network throughput and *prevents throughput collapse* at high offered load.

For this reason, Telegraphos uses *preventive flow control,* like many multiprocessor networks, and unlike ATM and traditional LAN/WAN architectures. Telegraphos uses *hop-by-hop credit-based* (back-pressure) flow control: packets are only transmitted when buffer space is known to exist for them in the next downstream switch. To keep track of such buffer space, a credit counter is maintained, which is decremented on each packet departure, and incremented when credit tokens are received from the downstream neighbor; a credit token is sent upstream upon each (downstream) packet departure. Several modern switches use similar flow control – e.g. iWarp [Bork90], J-Machine [NoWD93], C104 [Inmo93], DEC GIGAswitch/ATM [Dec94], Myrinet [Bode95].

2.2 Burst and Hot-Spot Tolerance

The nature of data traffic in multiprocessors, in networks of workstations (NOW) executing parallel applications, and in gigabit LAN's running distributed multimedia applications, is nowhere similar to the traditional network traffic. The latter is characterized by statistical multiplexing of a large number of independent and predictable (or controllable) sources, where the traffic load presented to the network by its sources does not exceed a certain limit – often around 50 to 90 percent of the maximum network throughput capacity. On the contrary, the former kind of traffic has none of these good properties [Kung92]. In parallel and distributed applications, the traffic sources are not independent (e.g. heavy traffic and hot spots arise at synchronization points), and their number may not be large (e.g. a file server, or a supercomputer, or a small number of currently active multimedia workstations). Throughput mismatches are not infrequent in gigabit LAN's: for example, high-speed servers or supercomputers can send messages at peak throughput to destinations that are incapable of receiving them at such a high rate – e.g. bridges to Ethernet. And, above all, parallel and distributed system communication is often characterized by *bursty* traffic; for example, in a study of eight scientific applications explicitly written for message-passing multicomputers, [CHKM93] found that half of the applications had an average message length of 8 Kilobytes or larger; in one application, this average was half a *Mega*byte.

Old solutions are no longer applicable in this environment [Kung92]. We cannot "over-design" the network so that it never gets congested, since that would require enormous network throughput. We cannot apply admission control, since that would allow the network to be monopolized by a few heavy-load connections. We cannot effectively apply end-to-end flow control, because that responds too slowly to network load fluctuations. We cannot restrict the peak throughput of sources to a small multiple of their average throughput, since the latter is usually minuscule while we want the former to be very large. And we don't want the network to provide very large buffering inside itself, because that merely postpones the solution of the throughput mismatch problem by carrying it over from the sources to the network itself. What is needed is VC-level flow control and buffer management.

2.3 VC-level Flow Control

If the back-pressure flow control signals indiscriminately refer to the entire link traffic, serious throughput degradation will result in the following frequent case. When a burst of identically destined packets enters the network (may be as pieces of a single large message), and if these encounter a throughput mismatch or if they collide with other packets on their way, indiscriminate back-pressure will cause these packets to accumulate inside the network switches, thus filling up the switch buffers. In turn, these full buffers prevent traffic going to other, uncongested destinations to pass through. This situation is analogous to wormhole routing without virtual-channel flow control [Dally90]: 20-flit messages passing through single-lane switches with 16-flit buffers, caused saturation at about 25% of link capacity in Dally's simulations (fig. 8 in [Dally90]).

To prevent such throughput degradation from happening, Telegraphos uses *virtual circuit (VC) flow control.* Each credit token contains a VC number, and gives permission to transmit one more packet belonging to that *specific virtual circuit.* If this were applied to large ATM systems, where multiple VC's are combined into one virtual path (VP) along a common portion of their route, each credit would refer to one specific VP. Virtual circuit hop-level flow control dates back to Tymnet [Rind77], GMDNET [Gies78], and Transpac [SiDa79]; see also the survey [GeKl80]. VC flow control and round-robin scheduling were used for achieving fair throughput allocation in [HaGa86], [Kate87], and [KaSC91]. The very similar technique of virtual channel flow control was used for deadlock-free wormhole routing in [DaSe87] and for improving the performance of wormhole routing in [Dally90], [Bork90]. Virtual circuit flow control extensions for ATM have been proposed and studied, e.g. in [KuBC94], [OzSV95].

In contemporary switches, with link rates in the Gbps range and packet/cell/flit sizes in the hundred-bit range, VC flow control has to be implemented in hardware. Few such hardware implementations (Torus [DaSe87], DEC GIGAswitch/ATM [Dec94], iWarp [Bork90]) are known to the authors or have been described in the literature. *Telegraphos* is one of them, and sections 3 and 4 describe this aspect of it in detail.

2.4 VC-level Buffer Management

To be effective, VC-level flow control must be combined with an appropriate buffer allocation policy. In our first prototypes, Telegraphos I and II, we used the simplest buffer allocation policy: one separate buffer slot, of size one packet, is statically allocated to each and every VC that passes through a switch. In this way, each VC is independent of all other VC's. In future switches, we will use a more economical policy: each individual VC (or VP) will still be allowed to use at most one buffer slot (one packet) per switch, but the switch buffer pool will be dynamically shared among the VC's. Under this policy, a packet can only be transmitted to the next downstream neighbor if a credit for its particular VC is available, *and* a credit for the proper buffer pool as a whole is available.

It is known that the buffer space allocation per VC and the credit-packet round-trip delay between neighboring switches determine the peak throughput that the VC can use as the ratio of the two former values. To allow high enough throughput per VC, when there is only a single busy VC, the buffer space allocation per VC should be large enough. Telegraphos adopts the following philosophy on this. Switches allow at most one buffer slot (one packet/cell) per VC. This simplifies the hardware because there is no need to maintain packet sequence information. In multiprocessor interconnection networks, in computer-room networks, and in office-area networks, links are only up to a few meters long, and round-trip delays are in the hundred-nanosecond range (with fast enough switch logic); with ATM cell sizes and Gbps link rates, a single-cell buffer slot is enough to allow a VC to use up the entire link throughput. For larger networks, extra interface chips (and perhaps additional RAM's) will be needed at the endpoints of long links, to provide additional buffer space; the extra buffer space is proportional to the link length, so the cost of these extra chips scales with the cost of the link, and will always be a small fraction of the latter.

2.5 Shared Buffering

Among the various buffer architectures for switches, input queueing is the simplest but also the one offering the least performance. A considerable number of real switches use it, like e.g. Nectar [Arno89], Meiko CS-2 [HoLa93], Inmos C104 [Inmo93], DEC GIGAswitch [Souz94]. Its head-of-line blocking property makes a switch with equal input and output throughput and with independent, randomly destined packet traffic, saturate at about 60% of link capacity [HlKa88] (bursty traffic causes worse performance). To improve the performance, one can go to non-FIFO input buffering. The disadvantages are more complicated buffer management [TaFr88], and more complicated output scheduling: the scheduling of each output depends on the scheduling of the other outputs – only one output port is allowed to use each buffer at any given time; [AOST93], [TaCh93], [LaSe94].

Telegraphos uses *shared buffering,* thus avoiding head-of-line blocking and complicated scheduling, and achieving maximum throughput and minimum latency. The single, shared buffer needs a throughput at least as high as the aggregate throughput of all switch links, as opposed to input queueing/buffering where each buffer only needs twice the link throughput. However, the aggregate throughput of all input buffers amounts to the same figure as the shared buffer throughput. Our Telegraphos experience indicates that the high aggregate throughput of the shared buffer is not a problem, and that shared buffering for high-speed switches is feasible, realistic, and has better cost-performance ratio than input queueing/buffering. Contributing to these virtues of shared buffering is a novel organization for such a buffer, including its input, output, and cut-through fabric, which we call *pipelined memory,* and which we use in our Telegraphos switches; it leads to simpler and more compact buffer implementation than the traditional wide or interleaved memory organizations. The pipelined memory organization is described elsewhere: [KaVE95]. Also note that, at least under "traditional" network assumptions, shared buffering is the queueing architecture that requires the fewest bits of total buffer memory to achieve a given low level of cell loss probability [HlKa88].

3. Hardware Organization of the Switch

This section describes the block diagram and the hardware operation of the Telegraphos switch. Emphasis is placed on the implementation of VC-level flow control; we also discuss the round-robin service properties and the (quite short) cut-through delay. To be specific, the switch described here is *Telegraphos I;* this is almost identical to Telegraphos II, with the exception of some path widths, memory sizes, and packet size – see section 4, table 1. Telegraphos I is our first prototype, and was designed using rapid prototyping techniques; it is made mostly out of field-programmable gate arrays (FPGA) and static RAM's. Several choices of parameters for this design are due to the technology used – the reader should not interpret them as being inherent characteristics or limitations of the Telegraphos architecture.

3.1 Physical Link Protocol

In Telegraphos I, links are unidirectional and consist of *ribbon cable* carrying 8 data bits and 1 flag bit in parallel; the flag identifies the header of every new packet. The packet size is 72 bits (9 bytes), so it takes 9 clock cycles for a packet to cross a link. The packet header (8 bits) is the VC number, and the remaining 64 bits are the payload. The body of the packet is always transmitted on a packet link during the 8 clock cycles immediately following the transmission of the header; after these 8 cycles, the link is idle until the next packet header appears. The Telegraphos I switch has 4 incoming and 4 outgoing packet links. *At most 256 VC's* are supported on each link.

It is assumed that a Telegraphos (prototype) network operates *synchronously* with a single clock, so no clock recovery and data synchronization circuits are included in the switch. This assumption was made for mere convenience, and should obviously be changed in a production switch; similarly a production switch would probably use high-speed serial links, and would only revert to parallel links (copper or fiber ribbon) if even higher throughput is desired.

Credits travel along upstream links. In Telegraphos I, for convenience reasons, we use separate, dedicated credit links. Like packet links, credit links also carry 1 flag bit and 8 data bits per clock cycle; the flag bit distinguishes between a valid credit and an idle link, while the data part of a valid credit (8 bits) identifies the VC that the credit refers to. In Telegraphos II (single chip version), we time-multiplex packets and credits on the same links, in order to economize on pins. Each link carries 8 data bits, 1 header flag, and 1 credit flag. Each packet may be "interrupted" by at most one credit, which is interposed between the packet bytes; when more credits must be transmitted, they are sent in between two different packets. Interruption of a packet by a credit is allowed in order to reduce the average credit transmission latency, thus reducing the credit-packet round trip delay (section 2.4). Interruption of a packet by two or more credits is not allowed in order not to have too many "bubbles" in a packet body (bubbles disrupt the operation of the pipelined memory [KaVE95]). Notice that the long-term average rate of credit transmissions over a given link is no more than 1 per packet time, although the short-term rate may reach as many credits per packet time as the number of outgoing links per switch.

3.2 Switch Block Diagram

Figure 1 shows the block diagram of the Telegraphos I switch. No timing information is contained in this figure, and pipeline registers are not shown. We will describe the flow of packets through this switch. The flag bits of the 4 incoming links are continuously monitored in order to know when new packets start arriving. When this happens, two actions are taken in parallel: *(i)* the packet body is collected in registers that convert it into parallel form (64 bits), and *(ii)* the packet header (incoming VC number) competes against packet headers arriving through other links for access to the routing table and to the packet body buffer.

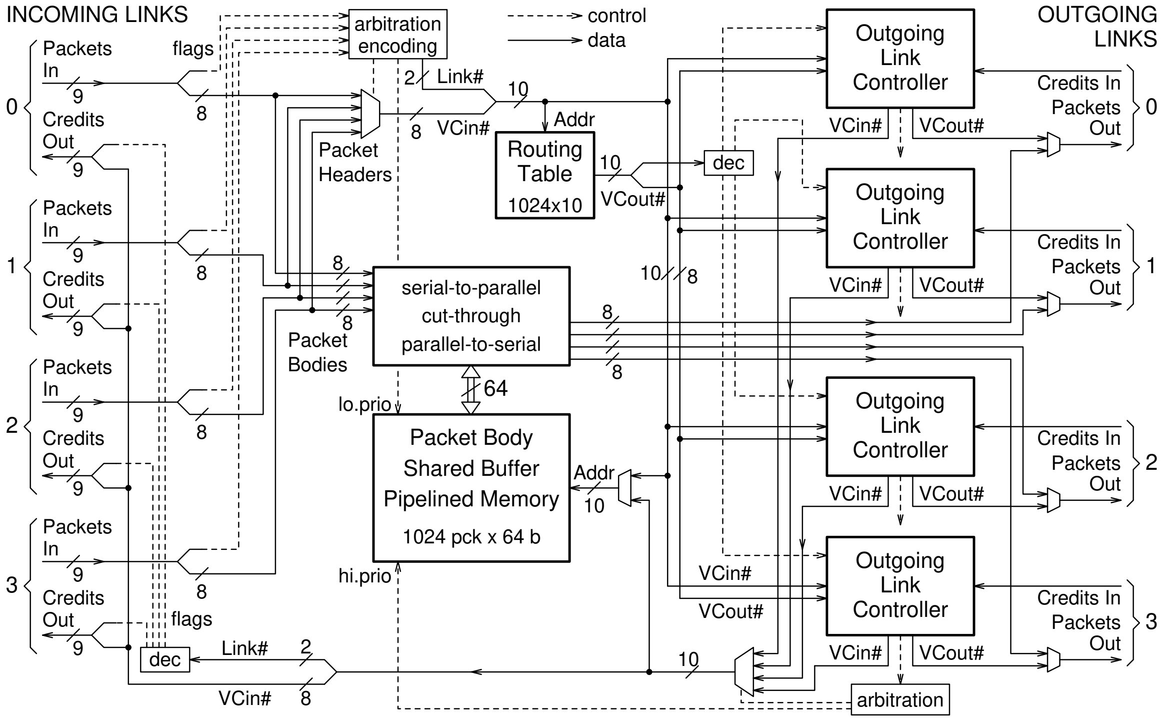

Figure 1: Block diagram of the Telegraphos I switch

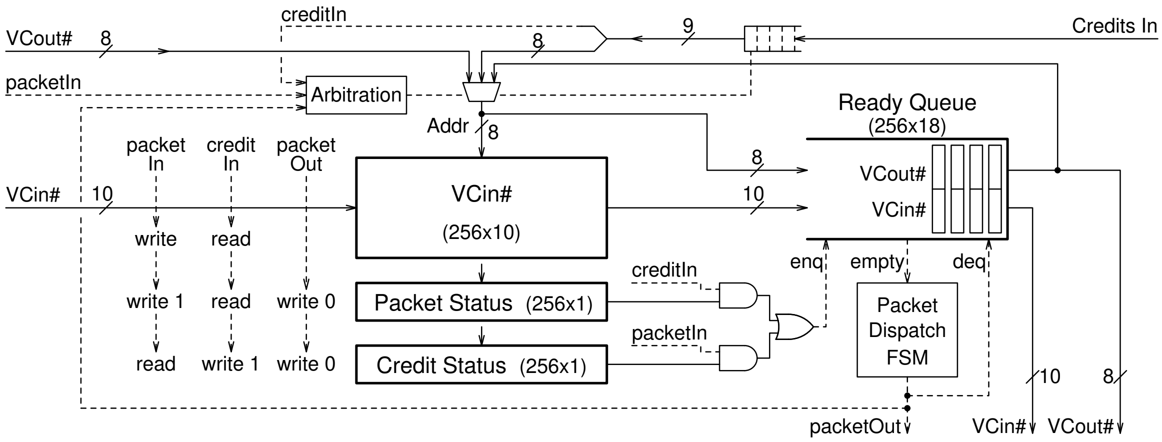

Figure 2: Outgoing link controller (one of four in figure 1)

The routing table (also called translation table in ATM) is a single-ported memory, and can service one request per cycle. At most 4 packet headers may arrive during any 9 consecutive clock cycles, so we have a known, short upper limit for the waiting time of each header. Routing in Telegraphos I is done as in ATM: the incoming link ID and VC number (VCin# in figure 1) of the packet determine, through a table, its outgoing link ID and new VC number (VCout# in fig. 1). Since there are 4 links per switch and at most 256 VC's per link, the Telegraphos I routing table is implemented as a 1024×10 memory (SRAM), addressed by the (incoming) linkID-VC pair. Multicasting is not provided in Telegraphos I or II, but we plan to add it in future switches.

Incoming packet bodies are stored in a pipelined memory shared buffer – see section 2.5 and [KaVE95]. According to the simple buffer allocation policy discussed in section 2.4, the address where a packet is stored is the concatenation of the packet's incoming link ID and VC number. Therefore, the size of the buffer is 1024 packets. At most one write or one read operation of a new packet can be initiated in the packet body buffer in each clock cycle; the buffer can sustain multiple parallel reception (write) or transmission (read) operations, but these must have started in a different clock cycle each. As explained in [KaVE95, § 3.4], this restriction only increases the average switch latency by a small fraction of a clock cycle, while, on the other hand, it simplifies the buffer hardware. During any packet-time interval, i.e. during any 9 consecutive clock cycles, there can be at most 4 new packet arrivals and at most 4 new packet departures, i.e. at most 8 write/read operation initiations in the shared buffer; it follows that the buffer has enough throughput capacity to always handle its peak requirement. The buffer access arbitration circuit gives priority to the outgoing links (read operations), because any delay to supply data to an outgoing link leads to idle time on that link, while delays to store incoming packets into the buffer memory have no direct consequence on the switch performance.

At packet arrival time, besides storing the incoming packet in the shared buffer and translating its input VC number, the appropriate *outgoing link controller* (OLC) is also notified about the arrival of this packet. There is one such controller for each outgoing link. Its function is to keep track of the packets waiting to go out of this link and the credits that this switch has for buffers in the downstream neighbor, to schedule the departure of packets over the link, and to dispatch the next ready packet when the link becomes available. The notification of arrival of a new packet is made by providing to the OLC a control signal and the incoming and outgoing VC numbers of the packet (VCin# is accompanied by the incoming link ID). The OLC also receives all credits arriving from its downstream neighbor.

When an outgoing link controller decides – in the way that we will see below – to transmit a new packet, it requests access to the shared packet buffer, first. When the OLC gains access to the packet buffer, it performs three operations in parallel:

- it sends VCout# to the outgoing link, as the new header of the packet;
- it sends VCin# to the buffer, as the address for reading the packet body; this body will flow out of the buffer, to the link, during the next 8 clock cycles; and
- it sends VCin# to the appropriate credit link out to the upstream neighbor, to let that switch know that now there is buffer space for one more packet of this VC; credit transmissions cannot conflict with one another, since they only occur when an OLC has gained access to the packet buffer.

3.3 Outgoing Link Controller

The internal organization of an outgoing link controller is shown in figure 2. For each of the up to 256 VC's that traverse this link, the presence or absence of a packet waiting to go out on that VC is recorded in the *packet status* SRAM, and the presence or absence of a credit for that VC is recorded in the *credit status* SRAM. These memories are indexed by the *outgoing* VC number, VCout#, because when credits come in, they contain that number (the downstream neighbor knows only that).

On packet arrival (control signal packetIn), the packet status of that VC is marked as 1 (and its VCin# is recorded for future use), and, simultaneously, its credit status is read; if the latter is 1, then a credit for this VC already exists, so the packet is *ready* for transmission. On credit arrival (creditIn), the credit status of that VC is marked as 1, and, simultaneously, its packet status is read; if the latter is 1, then a packet for this VC already exists, so that packet now becomes *ready* for transmission.

When a packet becomes ready for transmission, its VC numbers, VCin# and VCout#, are enqueued into the *ready queue* (VCin# could be re-generated from the corresponding memory, instead of occupying space in the ready queue, but that would increase cut-through latency by 1 cycle). The ready queue is implemented as a 256×18 SRAM, with two counters as enqueue and dequeue pointers; overflow can never occur, since at most one entry per VC can ever be present on that queue. Whenever the ready queue is non-empty and transmission of the previous packet approaches completion or is already complete, the packet dispatch finite state machine (FSM) dequeues the VC numbers of the next ready packet, and initiates its transmission as described in section 3.2. Also, the packet and credit statuses are marked as 0, indicating that now neither of them exists for this VC.

During any packet time (9 consecutive clock cycles), at most 4 packet arrivals may have to be recorded in the controller's data structures; this occurs when the packets arriving on all 4 incoming links are destined to the same outgoing link. Also, each credit link may carry at most 4 credit tokens during any packet time; this occurs when the packets departing from all 4 outgoing links had entered the switch through the same incoming link. In sum, there can be at most 4 packet-in operations, 4 credit-in operations, and 1 packet-out operation during any packet time; each of these operations requires at most one access to the controller's memories, so we have a maximum of 9 accesses during any 9 cycles. Packet-in and packet-out operations never conflict with each other, owing to the preceding arbitration for access to the shared buffer. In turn, these operations take precedence over credit-in operations; for this reason, a 4-entry credit queue is provided.

3.4 Cut-Through Latency

In a lightly loaded network, the cut-through latency (header delay inside a Telegraphos switch) is just *3 clock cycles.* These three cycles (numbered 1, 2, 3) are spent as follows in the switch:

Cycle 0: Incoming Link: the header of a new packet arrives on the incoming link wires.

Cycle 1: Arbitration: the routing table arbiter looks at the header flags of all incoming links and chooses one of the asserted flags; the corresponding header is multiplexed to the address input of the routing table.

Cycle 2: Routing: read the routing table; this indicates the desired outgoing link. Arbitration for packet-in access to that link's OLC (which is simple and fast) also happens at the end of this cycle. In parallel, arbitrate for access to packet body buffer.

Cycle 3: Outgoing Link Controller: read the credit status for this VC. If the ready queue is empty, the OLC possesses a bypass circuit: the VC numbers are sent directly to the outputs of the OLC. In parallel, buffer write for the packet body is also initiated in this cycle; the data being written are also retained in the buffer's output latches, so that if the credit status turns out positive they can be used for cut-through right away.

Cycle 4: Outgoing Link: VCout# is placed on the outgoing link wires. The first packet body byte is sent from the above latches to the link driver, ready to appear on the link wires during the next cycle.

3.5 Packet-Credit Round-Trip Delay

For a pair of neighboring Telegraphos switches with link latency of 1 clock cycle, the packet-credit round-trip delay is 8 cycles. Thus, a buffer space of one packet per VC is enough to achieve full link utilization even when there is only a single non-idle VC; see section 2.4. Here is a brief outline of how these 8 cycles are spent:

Cycle 0: header of packet p_0 on downstream link.

Cycle 1: header of p_0 competes for routing table.

Cycle 2: header of p_0 is looked up in routing table.

Cycle 3: OLC chooses p_0 for forwarding.

Cycle 4: header of p_0 (VCout#) on next downstream link; VCin# of p_0 on upstream credit link.

Cycle 5: in the upstream switch: credit token competes for access to OLC memories.

Cycle 6: credit written into OLC credit status memory.

Cycle 7: the next packet p_1 can read a positive credit during this or any subsequent cycle; therefore, OLC can choose p_1 for forwarding.

Cycle 8: header of packet p_1 on downstream link.

We see that cycle 8 is completely symmetric to cycle 0, so the period of this phenomenon *could* be as short as 8 clock cycles (=packet-credit round-trip delay). In reality, the period cannot be shorter than one packet time (9 cycles), but this proves that "back-to-back" transmission of packets over a single busy VC is possible with buffer space of just one packet per VC.

3.6 Round-Robin Servicing

The fact that at most one packet from each VC can be present in the switch at each time, and, consequently, each VC can appear at most once in the ready queue, implies that the VC service policy is *round-robin*-like. Specifically, any two VC's that are ready most of the time – i.e. which compete for as much throughput share as possible – will be serviced in alternation, because each of them re-enters the ready queue only after it reaches the queue front and gets serviced, and thus it re-enters behind the other VC. This round-robin service policy gives important fairness properties to the switch, including non-starvation and minimum-through guarantees. Under heavy traffic load, each VC gets an *equal share* of the available throughput, except for VC's that cannot or do not want to use as much throughput as their "fair share"; the throughput that remains unused by the latter VC's is equally distributed to the remaining VC's [Kate87], [Hahn91].

3.7 Switch Management

The switch includes a simple hardware control and management unit which implements all the necessary commands for initialization, set-up (and deletion) of connections through it, observing and changing its internal state, and monitoring its operation. The control and management commands arrive at the switch encapsulated in packets sent over the special VC number FF (hex) (similarly to the ATM pre-reserved control VC). Any incoming packet bearing this special VC number is treated by the switch as a management command. The management of a mesh of switches can be done from its periphery, by establishing VC's with endpoints at the various switches, using VC numbers other than FF along the entire VC path except its very last hop.

In general, management packets follow the normal flow inside the switch, as ordinary packets would do; however, as they pass through the different parts of the switch, their management tasks are triggered at the point(s) indicated by their opcode. The management commands which read the switch's state return their results to the management end-node by injecting reply packets into the network. The switch hardware can only handle one management packet at a time. This restriction greatly simplifies the hardware, while its effect on performance is negligible.

4. Implementations

As discussed above, *Telegraphos I* was our first prototype, using FPGA and SRAM technology. It was built and tested, and is currently working correctly in our labs. It took us about 7 quarters of elapsed time and 3 person-years of design, debug, and test effort. Subsequently, this design, with few modifications, was ported to a CMOS standard-cell ASIC technology; the resulting single-chip switch, *Telegraphos II,* was submitted for fabrication in June 1995. The effort for the porting and optimization to the new technology was about 1 person-year. At the same time, and with 1.5 additional person-years, the pipelined memory shared buffer and associated switching fabric was designed in full-custom CMOS technology and was sent for fabrication inside a test chip; it was named Telegraphos III. This section describes these implementations, measures their complexity, and compares them to each other. Table 1 compares the board-level and the standard-cell switches, and table 2 compares the semi-custom and the full-custom buffers.

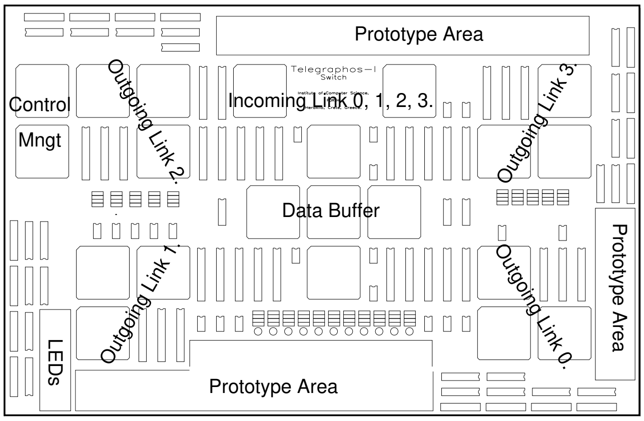

Figure 3: Floor plan of the Telegraphos I switch board

4.1 Telegraphos I: FPGA-based Prototype

The Telegraphos I switch is a 6-layer PCB (4 signal and 2 power layers), of size $42 \times 30\ cm^2$, with 4000 pins, 1000 nets, and $0.2\ mm$ wide traces; it contains 108 Kbits of single-ported SRAM plus the approximate equivalent of 14,500 gates of random logic in 21 FPGA's (Xilinx series 3000). The cycle time is 75 ns; this relatively long cycle time is due to the slow FPGA technology – especially when the logic has to be split among multiple FPGA's. The resulting link throughput is 107 Mbits/s in each direction.

Figure 3 shows a floor plan of the Telegraphos I switch board; the squares represent FPGA's, the rectangles in the center are SRAM memories, and the rectangles at the board edges are TTL buffers. The pipelined shared data buffer is the core of the design, and is located in the center of the board; it consists of five FPGA's for the control and peripheral circuits, and of eight SRAM chips on their sides. Each outgoing link interface includes 3 FPGA's (forming an L-shape), 6 memories, and several TTL buffers, and is located at one of the switch corners. The incoming link and management FPGA's are located upper-left, and top-center on the board, respectively.

Table 1 also presents detailed cost estimates; discrete flip-flops are included in the gate counts, memories are given separately, and area figures refer to PCB area. As seen, the shared data buffer is the largest sub-block in terms of complexity, with 6,000 gates and 64 Kbits of memory. It was partitioned very carefully in order to fit in 5 almost fully utilized FPGA's, and the wiring in this area is very dense requiring 4 signal layers to route all the interconnections. The four outgoing link controllers account for a total of 6,600 gates and 32 Kbits of memory, and occupy half of the board area. The cost of VC flow control is the packet and credit status memories and circuits – the ready queues would be needed even without flow control. We see that *VC flow control costs just 12 %* of the board area, or 1,300 gates and 14 Kbits of SRAM.

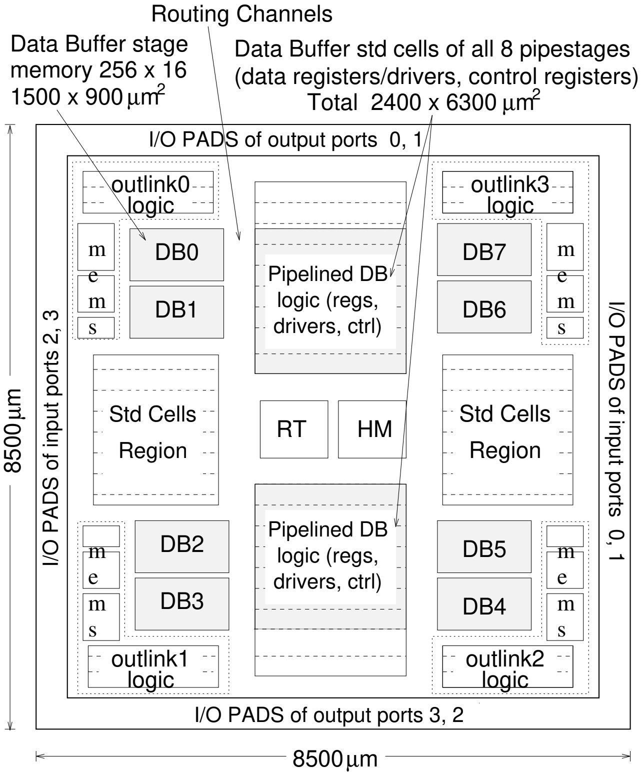

Figure 4: Floor plan of the Telegraphos II switch 0.7 µm CMOS standard-cell ASIC

Table 1: FPGA PCB and Standard-Cell ASIC Telegraphos Switch Prototypes

	Telegraphos I			Telegraphos II		
Purpose	Rapid Prototyping			First IC version		
Technology	FPGA Xilinx 3000, SRAM, TTL			ASIC (ES2 0.7 µm CMOS std cell)		
Size	$42 \times 30\ cm^2$			$8.5 \times 8.5\ mm^2$		
	80 chips (20 FPGA, 35 SRAM, 25 TTL)			120 pins		
Links	4 in × 4 out			4 in × 4 out		
Link Thruput (one dir)	8 bits / 75 ns = 107 Mbps			8 bits / 20 ns = 400 Mbps		
Logic Cycle Time	75 ns			40 ns		
Packet Size	9 bytes (1 hdr, 8 payload)			18 bytes (2 hdr, 16 payload)		
Buffer Size	1024 packets			256 packets		
Packet/Credit Links	separate			time-interleaved		
Cost Estimates	*(equiv. gates)*	*(Kbits SRAM)*	*(% area)*	*(gates)*	*(Kbits SRAM)*	*(% core area)*
Shared Data Buffer	6000	64	16	11100	34.5	51
Incoming Link Interfaces	4 × 275 = 1100	12	4	4 × 1225 = 4900	2.5	12
Outgoing Link Intf. (totl)	4 × 1650 = 6600	4 × 7.5 = 32	49	4 × 2100 = 8400	4 × 1.6 = 6.5	25
Packet/Credit Status	4 × 325 = 1300	4 × 3.5 = 14	12	4 × 900 = 3600	4 × .75 = 3	13
Ready Queue	4 × 1325 = 5300	4 × 4.5 = 18	37	4 × 1200 = 4800	4 × .875 = 3.5	12
Management, Misc	1000		31	1700		12
Total Switch	**14700 gates**	**108 Kb SRAM**	**100 %**	**26100 gates**	**43.5 Kb SRAM**	**100 %**

4.2 Telegraphos II: Standard-Cell Switch Chip

Telegraphos II is a single-chip implementation of the Telegraphos switch. It uses the 0.7 µm CMOS standard-cell process by European Silicon Structures Inc. (ES2). It is an 8.5×8.5 mm^2 chip in a 120-pin PGA package. As verified by *loaded worst-case* simulation, the clock cycle time is no more than 40 ns. The on-chip data paths are 16-bit wide, while the off-chip links are 8-bit wide, double-cycled, yielding a link throughput of 400 Mbits/s. The packet size is twice that of Telegraphos I, but the data buffer holds four times less packets. Packet and credit multiplexing on the same links (section 3.1) costs additional circuits in the incoming and outgoing link interfaces, and this is a main reason for the increase in gate counts from Telegraphos I to II in Table 1.

Figure 4 is the floor plan of Telegraphos II. At the center of the chip, RT is the routing table and HM is the (untranslated) header memory. The two standard cell regions at the left and right sides of the chip contain the input port logic, arbitration for the translation memory, and the management unit of the switch. The L-shaped areas marked by dotted lines at the four corners are the outgoing link controllers and the output registers and drivers. Similar to Telegraphos I, VC flow control is responsible for just 13 % of the (core) area. The shared packet buffer is identified by the shaded areas; it consists of 8 SRAM compiled megacells, and of peripheral circuits implemented using standard cells placed in the central rows. The total shared buffer area is to 32 mm^2, occupying half of the chip's core area. Thus, a more compact implementation is critical; this is the purpose of the Telegraphos III full-custom design.

4.3 Full-Custom Buffer and Switching Fabric

The pipelined memory shared buffer and associated switching fabric is such a regular structure that it greatly benefits from a full-custom layout. This was demonstrated with our *Telegraphos III* shared buffer prototype: we designed such a buffer in full-custom CMOS, we fully simulated it, and we submitted it for fabrication (June 1995) inside a test chip.

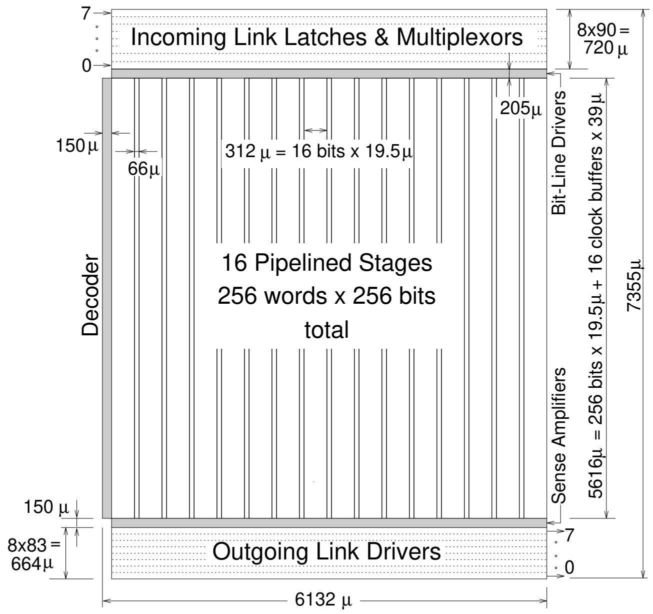

Figure 5: Floor plan of 16 Gbps, 64 Kbit pipelined buffer in 1 µm full-custom CMOS

This shared buffer has a packet size of 256 bits, a capacity of 256 packets, 8 incoming and 8 outgoing links. It uses the 1 µm CMOS technology (0.8 µm effective channel length) by ES2, with single poly and double metal layers. Figure 5 shows its floor plan. Its area is 45 mm^2, including crossbar and cut-through. Detailed SPICE simulation showed that, even in such a conservative technology, this 64 Kbit shared buffer for an 8×8 switch operates at 16 ns cycle time or 1 Gbps/link worst case (10 ns cycles or 1.6 Gbps/link typical). Worst case parameters are: 4.5 V supply, 125 °C junction temperature, slow transistors, and high parasitics.

In order to compare the semi-custom and the full-custom implementations of the pipelined memory shared buffer, we converted them both to a common size and technology; we chose the ATM cell size for the horizontal dimension, and the ES2 0.7 µm CMOS technology. For a 16×16 ATM switch, the memory is pipelined with 32 stages of 13 bits each and 1 stage of 8 bits. Table 2 shows the comparison results. As we see, full-custom layout yields a dramatic improvement of *one order of magnitude* in the area of the peripheral circuits, and 50 % improvement in the area of the core SRAM. The full-custom circuit is also *three times faster.*

Table 2: Full-Custom versus Semi-Custom Comparison: 256-cell Pipelined Memory for a 16 × 16 ATM Switch

Cell Size	53 Bytes = 424 bits		
Buffer Size	256 cells		
Technology	0.7 µm CMOS		
number of stages	32 (+1)		
Word width	13 (8)		
	Semi-Custom	Full-Custom	times better
Core Area	47 mm^2	31 mm^2	1.5
Periph. Ckt Area	128 mm^2	12 mm^2	10
Total Area	175 mm^2	43 mm^2	4
Cycle time	40 ns	14 ns	3
Aggregate Thruput	10 Gbps	30 Gbps	3

Conclusions

Virtual circuit (VC) flow control is essential for good performance of data communications in switch-based gigabit-per-second networks, and it is realizable at reasonable cost (10 to 20 % in area) and high speed in hardware. Shared buffering is a better switch architecture than input queueing; the pipelined memory is an efficient implementation of shared buffering, and it benefits greatly from full-custom layout. We showed three implementations (FPGA, semi-custom single-chip, full-custom buffer) of a fast fixed-size-packet switch that incorporates the above features, and we measured their complexity, which is in the order of 25 thousand gates and several tens of Kbits of SRAM.

Acknowledgements

Telegraphos was developed in the ESPRIT/HPCN project "SHIPS", and will be used by the ACTS project "ASICCOM", funded by the European Union (DG III and DG XIII). We deeply appreciate this financial support, without which Telegraphos would have not existed.

Telegraphos is a collective effort, with many contributors. The authors wish to acknowledge in particular E. Markatos, S. Kapidakis, P. Constanta-Fanouraki, and Ch. Nikolaou for valuable contributions to the Telegraphos design; G. Kalokerinos, Ch. Xanthaki, A. Dollas, and G. Papadourakis for implementing the rest of the Telegraphos I system prototype; G. Kornaros, Ch. Xanthaki, and N. Houstis for porting the Telegraphos I switch design to the Telegraphos II ASIC technology; and Ch. Kozirakis and M. Grammatikakis for contributing to the switch simulation. Also, the comments of D. Serpanos and A. Dollas were valuable. We thank all of them.

References

[AOST93] T. Anderson, S. Owicki, J. Saxe, C. Thacker: "High-Speed Switch Scheduling for Local-Area Networks", *ACM Trans. on Computer Systems,* vol. 11, no. 4, November 1993, pp. 319-352.

[AnCP95] T. Anderson, D. Culler, D. Patterson, and the NOW team: "A Case for NOW (Networks of Workstations)", *IEEE Micro,* vol. 15, no. 1, February 1995, pp. 54-64.

[Arno89] E. Arnould, F. Bitz, E. Cooper, H.T. Kung, R. Sansom, P. Steenkiste: "The Design of Nectar: A Network Backplane for Heterogeneous Multicomputers", *Proceedings of ASPLOS-III Conference,* ACM SIGARCH vol. 17, no. 2, April 1989, pp. 205-216.

[Bode95] N. Boden, D. Cohen, R. Felderman, A. Kulawik, C. Seitz, J. Seizovic, W-K. Su: "Myrinet: A Gigabit-per-Second Local Area Network", *IEEE Micro,* vol. 15, no. 1, February 1995, pp. 29-36.

[Bork90] S. Borkar e.a.: "Supporting Systolic and Memory Communication in iWarp", *Proc. of the 17th Int. Symp. on Computer Architecture,* ACM SIGARCH vol. 18, no. 2, June 1990, pp. 70-81.

[CHKM93] R. Cypher, A. Ho, S. Konstantinidou, P. Messina: "Architectural requirements of Parallel Scientific Applications with Explicit Communication", *Proc. of the 20th Int. Symp. on Computer Architecture,* ACM SIGARCH vol. 21, no. 2, May 1993, pp. 2-13.

[DaSe87] W. Dally, C. Seitz: "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks", *IEEE Trans. on Computers,* vol. C-36, no. 5, May 1987, pp. 547-553.

[Dally90] W. Dally: "Virtual-Channel Flow Control", *Proc. of the 17th Int. Symp. on Computer Architecture,* ACM SIGARCH vol. 18, no. 2, May 1990, pp. 60-68.

[Dec94] "GIGAswitch/ATM Networking Switch", *Digital Equipment Co.,* Nov. 1994, URL: file://ftp.digital.com/pub/Digital/info/infosheet/EC-F3924-42.ps

[GeKl80] M. Gerla, L. Kleinrock: "Flow Control: A Comparative Survey", *IEEE Trans. on Communications,* vol. 28, no. 4, April 1980, pp. 553-574.

[Gies78] A. Giessler, e.a.: "Free Buffer Allocation – An Investigation by Simulation", *Comput. Networks,* vol. 2, 1978, pp. 191-208.

[HaGa86] E. Hahne, R. Gallager: "Round Robin Scheduling for Fair Flow Control in Data Communication Networks", *Proc. IEEE Int. Conf. on Commun.,* June 1986, pp. 103-107.

[Hahn91] E. Hahne: "Round Robin Scheduling for Max-Min Fairness in Data Networks", *IEEE Journal on Sel. Areas in Communications,* vol. 9, no. 7, September 1991, pp. 1024-1039.

[HlKa88] M. Hluchyj, M. Karol: "Queueing in High-Performance Packet Switching", *IEEE Journal on Sel. Areas in Communications,* vol. 6, no. 9, December 1988, pp. 1587-1597.

[HoLa93] M. Homewood, M. McLaren: "Meiko CS-2 Interconnect Elan - Elite Design", *Meiko Ltd.,* 650 Aztec West, Bristol, UK, May 1993.

[Inmo93] "IMS C104 Packet Routing Switch – Preliminary Datasheet", *INMOS Ltd,* 1000 Aztec West, Bristol, BS12 4SQ. UK, June 1993.

[Kate87] M. Katevenis: "Fast Switching and Fair Control of Congested Flow in Broad-Band Networks", *IEEE Journal on Sel. Areas in Communications,* vol. 5, no. 8, October 1987, pp. 1315-1326.

[Kate94] M. Katevenis: "Telegraphos: High-Speed Communication Architecture for Parallel and Distributed Computer Systems", *Technical Report FORTH-ICS/TR-123,* ICS, FORTH, Heraklio, Crete, GR, May 1994, 38 pages, URL: file://ftp.ics.forth.gr/tech-reports/1994/94.TR123.Telegraphos.ps.Z

[KaSC91] M. Katevenis, S. Sidiropoulos, C. Courcoubetis: "Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip", *IEEE Journal on Sel. Areas in Communications,* vol. 9, no. 8, October 1991, pp. 1265-1279.

[KaVE95] M. Katevenis, P. Vatsolaki, A. Efthymiou: "Pipelined Memory Shared Buffer for VLSI Switches", *Proceedings of the ACM SIGCOMM '95 Conference,* Cambridge, Ma., USA, August 1995, URL: file://ftp.ics.forth.gr/tech-reports/1995/1995.SIGCOMM95.PipeMemoryShBuf.ps.gz

[Kung92] H. T. Kung: "Gigabit Local Area Networks: A Systems Perspective", *IEEE Communications Magazine,* vol. 30, no. 4, April 1992, pp. 79-89.

[KuBC94] H.T. Kung, T. Blackwell, A. Chapman: "Credit-Based Flow Control for ATM Networks: Credit Update Protocol, Adaptive Credit Allocation, and Statistical Multiplexing", *Proceedings of the ACM SIGCOMM '94 Conference,* London, UK, 31 Aug. - 2 Sep. 1994, pp. 101-114.

[LaSe94] R. Lamaire, D. Serpanos: "Two-Dimensional Round-Robin Schedulers for Packet Switches with Multiple Input Queues", *IEEE/ACM Trans. on Networking,* vol. 2, no. 5, October 1994, pp. 471-482.

[NoWD93] M. Noakes, D. Wallach, W. Dally: "The J-Machine Multicomputer: An Architectural Evaluation", *Proc. of the 20th Int. Symp. on Computer Architecture,* ACM SIGARCH vol. 21, no. 2, May 1993, pp. 224-235.

[OzSV95] C. Ozveren, R. Simcoe, G. Varghese: "Reliable and Efficient Hop-by-Hop Flow Control", *IEEE Journal on Sel. Areas in Communications,* vol. 13, no. 4, May 1995, pp. 642-650.

[Rind77] J. Rinde: "Routing and Control in a Centrally Directed Network", *Proc. Nat. Comput. Conf.,* Dallas, Tx, USA, June 1977.

[SiDa79] J. Simon, A. Danet: "Controle des Ressources et Principes du Routage dans le reseau TRANSPAC", *Proc. Int. Symp. Comput. Networks, Versailles, FR, February 1979.*

[Souz94] R. Souza, P. Krishnakumar, C. Ozveren, R. Simcoe, B. Spinney, R. Thomas, R. Walsh: "GIGAswitch System: A High-Performance Packet-Switching Platform", *Digital Technical Journal, DEC,* vol. 6, no. 1, Winter 1994, pp. 9-22.

[TaFr88] Y. Tamir, G. Frazier: "High-Performance Multi-Queue Buffers for VLSI Communication Switches", *Proc. of the 15th Int. Symp. on Computer Architecture,* ACM SIGARCH vol. 16, no. 2, May 1988, pp. 343-354.

[TaCh93] Y. Tamir, H-C. Chi: "Symmetric Crossbar Arbiters for VLSI Communication Switches", *IEEE Trans. on Parallel & Distributed Systems,* vol. 4, no. 1, January 1993, pp. 13-27.