A Generic Architecture for On-Chip Packet-Switched Interconnections

Pierre Guerrier Alain Greiner

Université Pierre et Marie Curie
4, place Jussieu, F-75252 PARIS CEDEX 05
name.surname@lip6.fr

Abstract

This paper presents an architectural study of a scalable system-level interconnection template. We explain why the shared bus, which is today's dominant template, will not meet the performance requirements of tomorrow's systems. We present an alternative interconnection in the form of switching networks. This technology originates in parallel computing, but is also well suited for heterogeneous communication between embedded processors and addresses many of the deep submicron integration issues. We discuss the necessity and the ways to provide high-level services on top of the bare network packet protocol, such as dataflow and address-space communication services. Eventually we present our first results on the cost/performance assessment of an integrated switching network.

1. Introduction

In the year 1999, the first .18 μ fabs have moved into volume production. With such process technology, chip designers can create systems-on-a-chip (SoC) by incorporating several dozens of IP blocks, each in the 50-100 kGate range, together with large amounts of embedded DRAM. Those IPs can be CPU or DSP cores, video stream processors, and high-bandwidth I/O (like IEEE 1394, Gigabit Ethernet or DACs).

SoC designs pose a number of daunting methodology problems: how to specify the system ? how to map the specification onto a collection of available IPs ? how to evaluate design options ? These are mostly CAD issues, and some commercial solutions are appearing. However, they build on the implicit assumption that SoC will stick to the "*Central Bus*" architecture template, or slightly enhanced multi-bus variants.

Yet virtually every IP in the portfolio evoked above, has I/O requirements in the Gbit/s range: fast CPUs, fast network controllers, and multimedia processors. Figure 1 is a synopsis of the latency and throughput required by a few kinds of traffic. They may all be featured in a single consumer equipment, for instance an advanced multimedia PDA. If we want to exploit task-level parallelism between processing IPs, with concurrent reconfigurable communications, then aggregated interconnection throughputs like 50 Gbit/s are needed. This requirement will continue to grow as the number of IPs incorporated on chip will increase, as well as the individual performance of each IP (with faster processors, better video quality...).

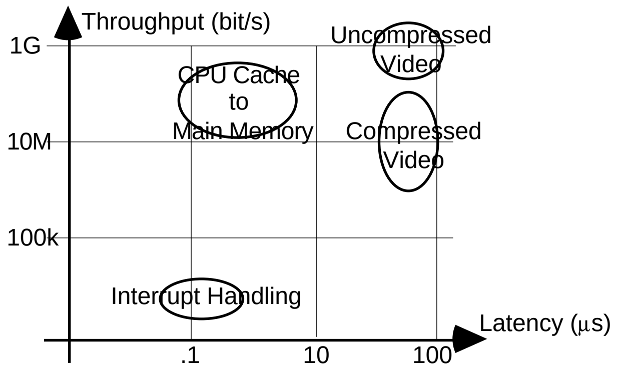

Figure 1: Requirements for some traffics

Bus-based architectures will not meet this requirement because a bus is *inherently non-scalable*. The bandwidth of a bus is shared by all attached devices, and it is simply not sufficient, firstly because the bus width cannot reasonably exceed a hundred bits, and secondly because the clocking frequency of global wiring becomes tightly constrained by the electrical properties of deep submicron processes [1]. New bus proposals are still being made, e.g. in the framework of [2], but in our opinion their only clear advantage is to standardize the IP interfaces. These architectures advocate multiple on-chip busses, requiring case-specific grouping of IPs and the design of transversal bridges, which does not make for a truly scalable and reusable interconnection.

This paper proposes another generic interconnection template that addresses the performance and scalability requirements of systems-on-chip, using integrated switching networks. We aim to prove that present manufacturing technology already enables the realization of a switching network with the wrappers to carry on both address-space and dataflow communications.

The paper is organized as follows: Section 2 provides the reader with a practical understanding of switching concepts. Then section 3 explains how they should be tuned for on-chip environment. Section 4 is a cost analysis of a tentative realization of such a network. Section 5 analyzes its performance through simulation benchmarks. Section 6 shows how hardwired protocols can make the network compatible with the synchronous dataflow I/O semantics as it is used in high performance video processors, or the address-space semantics of the VCI standard. Eventually section 7 will discuss the present shortcomings of the proposed architecture and possible solutions.

2. Switching network basics

We call switching network, a set of switching elements connected together by full duplex point-to-point links. The reader is referred to [3] for a milestone presentation. The switching elements can operate in space (S), by establishing physical connections between their terminals. The typical S-switch is the crossbar. Other switches operate in time (T), using buffers to swap the order of timeslices on time-division multiplexed links. Historically, combinations of S and T switches have been used to build telephony networks.

Such networks implement a *circuit-switching technique*, where connections are established between two terminals by assigning them a set of time-slices on the network links. This set has to be determined by clever computations when the connection is requested, and subsequently remains constant during the entire connection. The main advantage is the *formal guarantee of bandwidth* resulting from the static establishment of the circuit. The PROPHID architecture template [4] has demonstrated a remarkable application of circuit-switching to on-chip communication. PROPHID uses a T-S-T switch (figure 2) to cascade dataflow video processings. It is also the only example to our knowledge of a non-bus SoC interconnection. (We do not consider prototyping platforms such as [5], which featured a single S-switch connecting dataflow computing resources, because those systems were not intended for production and could only be reconfigured at start-up time.)

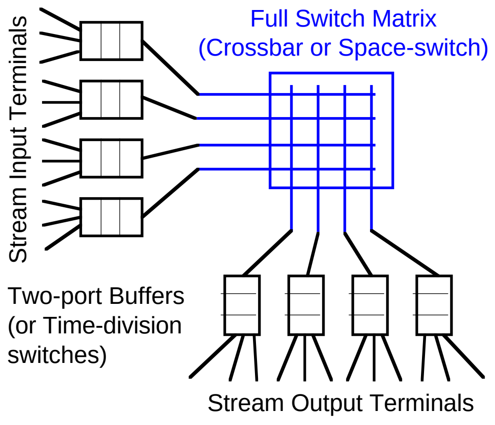

Figure 2: The PROPHID interconnection

However, the drawback of circuit-switching is the lack of reactivity against rapidly changing communications. For instance, the PROPHID interconnection cannot dynamically increase a bandwidth allocation to match bursts in an MPEG bitstream. It is even less suited to the random traffic between a CPU master and several slaves on a bus. For this reason, networks in parallel computers or LANs have used a different technique known as *packet-switching*.

A packet-switched network moves data from one of its terminals to another, in small formatted chunks called packets. These consist in a header identifying the destination of data, followed by a payload (the data itself), unambiguously terminated by a trailer. The switching elements of the network are called *routers* and operate in space. When it receives a packet, a router forwards it to one of his neighbors (chosen according to the header information). Packets repeatedly undergo this process until they reach their final destination. Since routing decisions are distributed over the routers, the network can remain very reactive even for very large sizes.

Despite the many routing hops, low latency can be maintained if routers forward the header of packets ASAP, without waiting for the trailer (figure 3). This technique is called *wormhole routing* and has been used extensively in high-performance parallel computer networks [6]. In a low-cost wormhole switch, buffers are small and packets typically span a handful of routers. If a packet requests a link which is already busy, and this packet cannot be entirely buffered in the router, the macropipeline formed by the other links and routers spanned by the packet will be stalled. This may result in many links being inefficiently kept busy. Such *cascaded contentions* practically prevents any packet-switched network from delivering its full theoretical bandwidth.

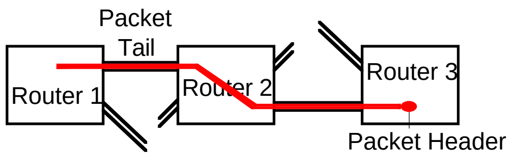

Figure 3: A wormhole packet switch

3. An on-chip switched network

This section is an overview of the global design options for a *Scalable, Programmable, Integrated Network* (SPIN) for packet-switched system-on-chip interconnections. The design decisions concern the nature of elementary links, the topology of the network, the packet structure, and the network access protocols.

• The point-to-point link should be able to stand the throughput of a bus devoted to a single master. Therefore we decided on a parallel link built of two one-way 32-bit data paths. In contrast to a bus, there are no bi-directional wires. We use a credit-based flow control on the paths:

buffer overflows at the target end of a path are checked at the source, using a counter to track the amount of free buffer space. The receiver notifies the sender of every datum consumed, with a dedicated feedback wire. This inexpensive mechanism provides latency-independence: the links can be pipelined transparently to achieve the desired clock speed in a submicron process.

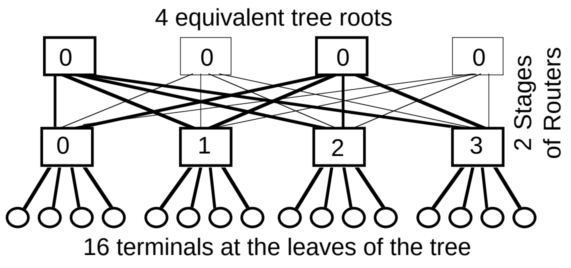

Figure 4: A fat-tree network

• The network topology impacts the complexity of the distributed routing decisions. Regular topologies like meshes or hypercubes make the decision functions simple and identical for every router, so the same macrocell can be reinstanciated to build the network. Among those, we preferred the Fat-Tree (figure 4) because Leiserson has proved formally that it is the most cost-efficient for VLSI realizations [7].

A fat-tree is a tree structure with routers on the nodes and terminals on the leaves, except that every node has replicated fathers. If there are as many fathers as children on all nodes, then there are as many roots as leaves, and the bisection throughput is preserved: the network is non-blocking. We chose a four-child tree because the 8x8 router seemed most convenient for VLSI implementation. The size of this network grows like $(n.\log n)/8$ with the number of terminals. Table 1 lists the costs of some network instances.

Attached	Resources	
Units	Routers	Links
8	2	12
16	8	32
32	16	96
64	48	192
128	96	448

Table 1: Scaling the SPIN fat-tree

• Packets consist of sequences of words of 32 bits. This width allows the header to fit in a single word. A byte in this word identifies the destination (this allows 256 terminals) and other bits are used for packet tagging and routing options. The routers are free to use any of the redundant paths in the fat-tree to route a packet. This feature is called adaptivity and reduces contention hot-spots. The packet payload may be of any size, and possibly infinite. Finally the trailer is a special word marked by a dedicated control line. It contains a checksum of the payload data.

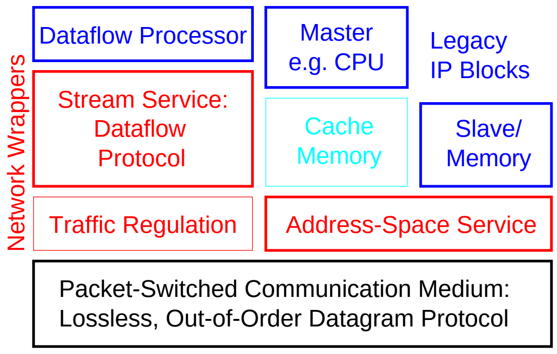

Figure 5: Hardwired protocol stack

• Packets natively implement the message passing communication model. Messages can be used to build protocols emulating other models like dataflow streams and address-space. We believe that the future system-on-chip will be heterogeneous, featuring at least these two communication mechanisms. They could be used together on some units, since they serve different purposes: for instance, application software configures and monitors dataflow processors through addressable registers. Hence, we specified a modular network access terminal, that can be shared by several wrappers, each providing a different service to the attached unit. The packets are tagged to associate them with services. Each service uses a private payload format for its packets. Figure 5 illustrates the parallel between this modular, *fully hardwired* approach, and a layered stack of APIs.

4. Router design and cost

The critical component of the network described in section 3 is the router macrocell. On one hand, it must be carefully optimized for area, because it is reinstanciated many times in a single system. On the other hand, the packet buffering strategy of individual routers strongly impacts the global performance of the network: the most conservative model where packets are queued in FIFOs at the router inputs is known to generate the highest contention [8]. But the smarter buffering schemes used in general purpose discrete routers require expensive hardware to clear the inputs by queuing packets on additional crossbar channels called "*output buffers*".

However, careful analysis of our topology shows that contention will mostly result from packets flowing down the tree towards child links, because alternate paths are only available for the father links, and also because those packets span more routers on average. We based our design (figure 6) on this property and the experience of the successful realization of the discrete router RCube™ [9]: Small (4-word) input buffers are necessary for hiding the delay of the control logic and the link latency. In case of output contention, child-bound packets can use two output buffers of 18 words each. They reduce cascaded contention by providing a longer side track for halted packets. It is more hardware-efficient than simply making all input buffers longer, because in most practical conditions only a couple of inputs will be subject to contention. The chosen size handles most efficiently packets shorter than 18 words, with payloads as large as typical bus bursts. The crossbar grows to 10×10 because of the output buffers, but it is not full because all routes are not possible.

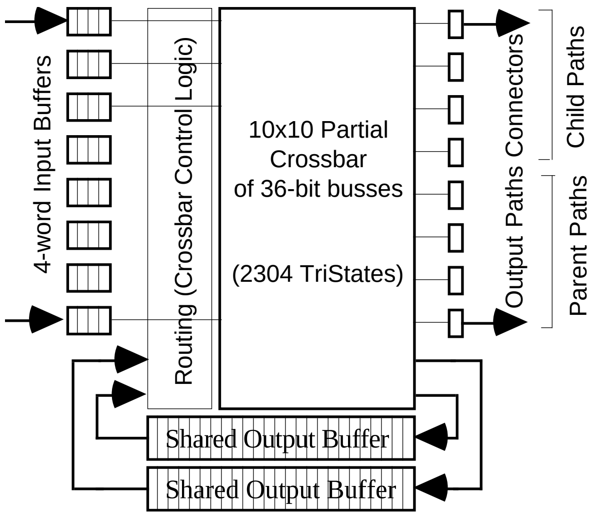

Figure 6: Router datapaths synopsis

We used the symbolic layout standard cell library of the ALLIANCE system to synthesize the router. The regular parts (buffers and crossbar) were placed manually using metal layers 4, 5 and 6 to interconnect the switch matrix. Figure 7 shows this placing. The area is 1×0.8 mm^2 in a .25 μ process. The control logic is synthesized, placed and routed by automatic tools in the empty space, using metal layers 2 and 3. From this geometry and from table 1, we can forecast the network costs for future systems, which we summarized in table 2.

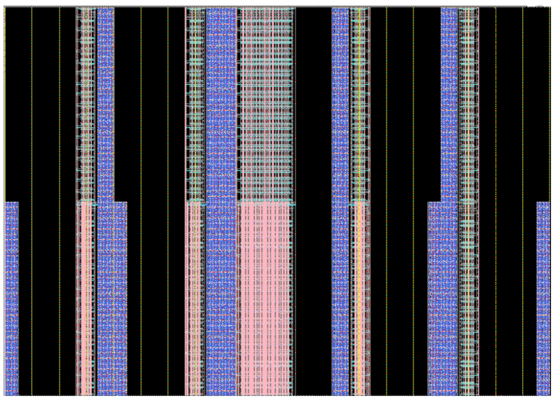

Figure 7: Router floorplan

The control logic can be pipelined to match virtually any speed the switching matrix may reach. Therefore the timing is constrained by the delay of the wires in the matrix. Electrical simulation taking into account crosstalk capacitances suggests a cycle time below 5 ns for a .25 μ CMOS process. Table 2 assumes this delay will scale in accordance with the SIA roadmap projections. A test silicon of the router macrocell is scheduled for the end of this year, which will enable accurate performance and power measurements.

Process	Attached Units	Peak Bandwidth	Network Area
.25 μ, 6ML	32	205 Gbit/s	< 13 mm^2
.18 μ	64	568 Gbit/s	< 20 mm^2
.13 μ	128	1.82 Tbit/s	< 20 mm^2

Table 2: Projected network costs

Eventually, we paid careful attention to the network testability issues, as no off-the-shelf test method can handle a system comprising dozens of crossbars and hundreds of FIFOs. We use a graph property of the fat-tree shown on figure 8: the graph is *eulerian*, thus a common predefined connection scheme can be applied to all routers to create paths covering all links and buffers in the network. For the test mode, these paths are daisy-chained and fed with a pattern generator. All the global interconnect is tested by this method. Stalls and bubbles are included in the stream to stimulate the FIFOs and test them without any scan path. In addition, an input-output loop-back test is applied to every switching matrix individually.

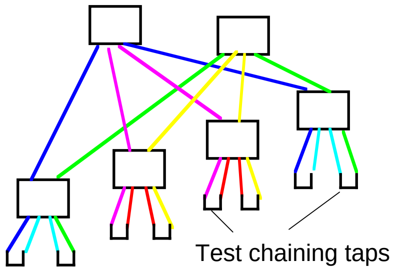

Figure 8: Global test paths

5. Performance evaluation

A fast cycle-true, bit-true simulation model of the SPIN router was written in C language for the CASS system-level simulation tool [10]. This model allowed us to test large networks (up to 256 terminals) under a range of benchmark loads. The most common benchmark in use in the networking literature is the *uniform random distribution* of packet destinations. The results of this test are a *pessimistic estimate* of the practical performance of a network, because the load does not exhibit any locality that the tree clustering could exploit. Since all paths of the network are equally loaded, it also reduces the advantages of router adaptivity. Despite these artifacts, we tested the network under a random load to compare it against past realizations, while still getting a worst-case performance prediction.

Figure 9 summarizes the key data for a 32-terminal network (16 routers) with 20-word packets spread uniformly. The simulations were run for one million clock cycles. This is enough to provide ±1% accurate population counts. The *offered load* is the average proportion of cycles at which data is injected into unbounded buffers connected to each input terminal. These buffers become saturated when the network cannot absorb the offered load. Figure 9 shows how the packet latency grows as the network is loaded, until it reaches saturation. Note that this latency includes the time spent in the injection buffers before entering the network. The time actually spent *inside* the network is always lower.

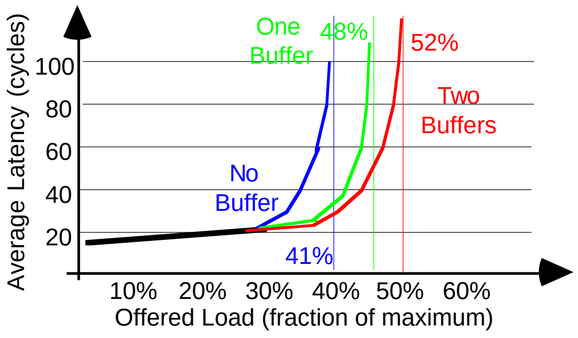

Figure 9: Load benchmarks

The curves show the performance impact of the two shared output buffers. These simulations permitted to optimize the cost/performance of the realization presented above. The results are satisfactory by (discrete) networking standards, although somewhat lower than state-of-the-art. This is remarkable given the very small depth of the input buffers, and it demonstrates the relevance of our topology-specific buffering policy. Simulations with larger networks have shown little performance degradation. We conclude that a network clocked at 200 Mhz would easily deliver 2 Gbit/s per terminal, and still scale nicely to aggregated throughputs up to 100 Gbit/s for 32 terminals.

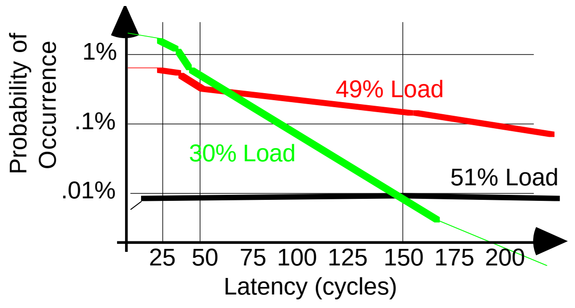

Figure 10: Latency distributions

Nevertheless, figure 10 shows an important drawback of the switched network: all latencies tend to occur, with an exponentially decreasing probability. This means that nearly all packets will be delivered in a small time. However, there *will be* some random hick-ups, all the more frequent as the network is heavily loaded. These simulations are for a 64-terminal network (48 routers), with a uniform load, where most packets have the maximum number of routing hops (i.e. 5 hops).

6. The communication protocols

This section presents results on the hardware implementation (i.e. wrappers) of the protocols enabling dataflow or address communications through the packet-switched network. The goal is to emulate these models using the message passing model. In addition, the two bandwidth-effective features of SPIN, adaptivity and output buffering, have the unpleasant property of swapping packets. Protocols must also enforce strict in-order delivery for dataflow and for some address-space transactions.

Regarding stream communications, we imagined a protocol based on our experience with sender-based protocols for adaptive networks [11]. The key is an *unambiguous chronological tagging* of every packet transmitted in a stream. Upon reception, packets are stored in the wrapper, which may deliver them in order to the dataflow processor. Buffer storage is reserved for the missing packets until they arrive. If a packet is very late, the processor stalls, the entire buffer becomes busy, and other incoming packets have to be rejected in the network, which is used as a delay line. An end-to-end credit-based traffic regulation bounds the amount of outstanding stream data and thus prevents rejection from causing a catastrophic network congestion. Both statistical modeling and cycle-true simulations show that this spurious traffic is negligible given the latency distribution of SPIN. The total bandwidth overhead introduced by the protocol is 31% of the stream throughput.

We synthesized a VHDL description of this hardware protocol stack, supporting four concurrent streams per terminal. The basic network access layer, necessary to plug in any service wrapper, test the network and check data integrity, represents about 0.15 mm^2 in a .25 μ process. The stream traffic regulation layer is less than 0.1 mm^2. The stream protocol itself is only 0.1 mm^2, plus a dual-port SRAM of 1 to 4 kbits (0.1 to 0.3 mm^2 with our symbolic SRAM generator). In any configuration, the full network interface is smaller than 0.6 mm^2 in this process, making it affordable for systems comprising up to a dozen stream processors, each with gigabit/s throughput.

Regarding address-space communications, we are defining a protocol for wrappers matching the "Virtual Component Interface" address-oriented standard [2]. The basic principle is to translate the VCI request/response packets into SPIN packets. However for optimal performance, the initiator must take advantage of the VCI *split transactions*: because the delay of return-trips through the switched network is large, several transactions must be overlapped, as shown on figure 11, to use the full link bandwidth, e.g. for cache to memory traffic. Memory throughput can then be scaled by distributing it over several network terminals standing concurrent accesses, like in *Distributed Shared Memory* multi-computers.

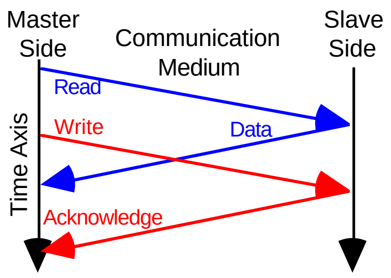

Figure 11: The split transaction model

Addressing the memory over a switching network raises a software compatibility concern, for those features that relied upon the snooping of a central bus, like semaphore synchronization and cache invalidation. DSM multi-computers have shown that a comprehensive support for cache consistency is possible but too complex for an embedded system [12]. Fortunately it is not needed because these systems are rather heterogeneous, asymmetric multiprocessors, with explicit invalidations and simple synchronization primitives. These can be supported more easily by simple protocols. We do not presently have a cost measurement of the VCI wrappers for SPIN. However, we believe they will be smaller than the dataflow wrappers, because they do not require any data buffering.

7. Present limitations

A well-known criticism against our approach, is the complexity of switching network concepts. The design space for networks is different and larger than for busses, with new caveats and new refinements. Although all silicon engineers have seamlessly used commodity networks (through telephones, NFS or Internet...), they are unfamiliar and ill-at-ease with key network aspects like true concurrence or statistical behavior prediction. This problem cannot be solved by improvements to the proposed architecture, although protocols and CAD tools can help hiding its internal intricacies. The true solution is the education of the designer community to the broader perspectives opened by deep submicron systems [13].

8. Conclusions

We have shown a new architecture template for system-level interconnection, based on switching networks. We assessed the cost and the performance of this template through joint functional modeling and physical implementation of key parts of the architecture. Our results demonstrate that it matches the throughput, latency and area requirements of future systems-on-chip.

We acknowledged the need for legacy communication protocols to enable straightforward reuse of existing IPs in industry designs. Therefore we have devised a layered protocol architecture, to provide a set of industry-standard communication mechanisms on top of a switching network. Results of such a mechanism for dataflow communication have been presented, including detailed silicon area evaluation. VCI-compliant wrappers for address-space communications are being actively investigated.

Bus Pros & Cons			Network Pros & Cons
Every unit attached adds parasitic capacitance, therefore electrical performance degrades with growth.	-	+	Only point-to-point one-way wires are used, for all network sizes.
Bus timing is difficult in a deep submicron process.	-	+	Network wires can be pipelined because the network protocol is globally asynchronous.
Bus testability is problematic and slow.	-	+	Dedicated BIST is fast and complete.
Bus arbiter delay grows with the number of masters. The arbiter is also instance-specific.	-	+	Routing decisions are distributed and the same router is reinstanciated, for all network sizes.
Bandwidth is limited and shared by all units attached.	-	+	Aggregated bandwidth scales with the network size.
Bus latency is zero once arbiter has granted control.	+	-	Internal network contention causes a small latency.
The silicon cost of a bus is near zero.	+	-	The network has a significant silicon area.
Any bus is almost directly compatible with most available IPs, including software running on CPUs.	+	-	Bus-oriented IPs need smart wrappers. Software needs clean synchronization in multiprocessor systems.
The concepts are simple and well understood.	+	-	System designers need *reeducation* for new concepts.

Table 3: The bus-versus-network arguments

Bibliography

[1] K. Keutzer, "*Chip Level Assembly (and not Integration of Synthesis and Physical) is the Key to DSM Design*", Proceedings of the ACM/IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems (Tau'99), Monterey, CA, USA, March 1999.

[2] Virtual Socket Interface Alliance, "*On-Chip Bus Attributes*" and "*Virtual Component Interface - Draft Specification, v. 2.0.4*", `http://www.vsia.com`, September 1999 (document access may be limited to members only).

[3] C. Clos, "*A Study of Nonblocking Switching Networks*", Bell System Technical Journal, vol. 32, no. 2, pp. 406-424, 1953.

[4] J. Leijten et al., "*Stream Communication between Real-Time Tasks in a High-Performance Multiprocessor*", Proceedings of the 1998 DATE Conference, Paris, France, March 1998.

[5] D. C. Chen and J. M. Rabaey, "*A Reconfigurable Multiprocessor IC for Rapid Prototyping of Algorithmic-Specific High-Speed DSP Data Paths*", IEEE Journal of Solid-State Circuits, 27, no. 12, pp. 1895-1904, December 1992.

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

[7] C. Leiserson, "*Fat-Trees: Universal Networks for Hardware-Efficient Supercomputing*", IEEE Transactions on Computers, vol. C-34, no. 10, pp. 892-901, October 1985

[8] M. Karol et al., "*Input versus Output Queueing on a Space-Division Packet Switch*", IEEE Transactions on Communications, pp. 1347-1356, December 1987.

[9] B. Zerrouk et al., "*RCube : A Gigabit Serial Link Low Latency Adaptive Router*", Records of the IEEE Hot Interconnects IVth Symposium, Palo Alto, CA, USA, August 1996.

[10] F. Pétrot et al., "*Cycle-Precise Core Based Hardware/Software System Simulation with Predictable Event Propagation*", IEEE Computer Society Press, Proceedings of the 23rd Euromicro Conference, Budapest, Hungary, pp. 182-187, September 1997.

[11] F. Wajsbürt et al., "*An Integrated PCI Component for IEEE 1355*", Proceedings of the 1997 EMMSEC Conference and Exhibition, Florence, Italy, November 1997.

[12] J. Hennessy, D. Patterson, "*Computer Architecture, A Quantitative Approach - 2nd Edition*", Morgan Kaufmann Publishers, San Francisco, CA, USA, 1996.

[13] H. de Man, "*Education for the Deep Submicron Age: Business As Usual ?*", Proceedings of the 34th Design Automation Conference, Anaheim, CA, USA, March 1997.