On the Design of a High-Performance Adaptive Router for CC-NUMA Multiprocessors

Valentín Puente, José-Ángel Gregorio, *Member*, *IEEE Computer Society*,
Ramón Beivide, *Member*, *IEEE Computer Society*, and Cruz Izu

Abstract—This work presents the design and evaluation of an adaptive packet router aimed at supporting CC-NUMA traffic. We exploit a simple and efficient packet injection mechanism to avoid deadlock, which leads to a fully adaptive routing by employing only three virtual channels. In addition, we selectively use output buffers for implementing the most utilized virtual paths in order to reduce head-of-line blocking. The careful implementation of these features has resulted in a good trade off between network performance and hardware cost. The outcome of this research is a High-Performance Adaptive Router (HPAR), which adequately balances the needs of parallel applications: minimal network latency at low loads and high throughput at heavy loads. The paper includes an evaluation process in which HPAR is compared with other adaptive routers using FIFO input buffering, with or without additional virtual channels to reduce head-of-line blocking. This evaluation contemplates both the VLSI costs of each router and their performance under synthetic and real application workloads. To make the comparison fair, all the routers use the same efficient deadlock avoidance mechanism. In all the experiments, HPAR exhibited the best response among all the routers tested. The throughput gains ranged from 10 percent to 40 percent in respect to its most direct rival, which employs more hardware resources. Other results shown that HPAR achieves up to 83 percent of its theoretical maximum throughput under random traffic and up to 70 percent when running real applications. Moreover, the observed packet latencies were comparable to those exhibited by simpler routers. Therefore, HPAR can be considered as a suitable candidate to implement packet interchange in next generations of CC-NUMA multiprocessors.

Index Terms—Interconnection networks, adaptive routing, hardware router design, shared memory multiprocessors.

1 Introduction

Nowadays, the advances in microelectronic technology offer computer architects a lot of raw logic power, allowing the implementation of traditional off-chip modules on the processor die. However, when considering medium to large-scale parallel systems, it is evident that the off-chip interconnection network will become a crucial component, affecting the whole system performance. On the one hand, the performance of a parallel system will be seriously penalized if the network is not able to handle the increasing volume of information generated by the processing elements when executing intensive data interchange workloads. On the other hand, message latency will be as critical as maximum sustained throughput when executing latency-sensitive workloads. Both kinds of traffic patterns are present in common parallel applications. Moreover, it is very usual to encounter several execution phases in a single application managing different workload types.

This paper presents a detailed router architecture for parallel machines designed to optimize network throughput while maintaining a low node pass time, thus fulfilling the requirements of the next-generation multiprocessor systems. In this work, we focus on the CC-NUMA class of multiprocessors, which is one of the most popular architectures in the high-performance computing field due to its good scalability and easy programming. We consider a CC-NUMA machine having a 2D torus topology, which is a common selection for medium-to-large multiprocessor systems due to its good cost/performance ratio [1], [7]. In addition, we employ virtual cut-through (VCT) flow control. Although several commercial systems use wormhole flow control, they all provide large buffers with capacity for hundreds of flits. In fact, the Cray T3E also sets a maximum packet size, narrowing the barrier between the two flow control techniques. However, VCT simplifies switching among multiple virtual channels and it is less deadlock-prone. Thus, recent routers such as those used in the BlueGene/L supercomputer [2] and the Alpha 21364 microprocessor [3] incorporate VCT flow control.

It is well known that many parallel applications present specific communication patterns, in general far from uniform distributions. A deterministic router, while simple, will limit the maximum packet throughput due to its unbalanced use of network resources. Adaptive routing is preferable although it implies more complex routing logic. Moreover, this complexity translates to other key components such as the arbiter and the internal switching fabric. Consequently, for a new adaptive routing proposal to succeed, it needs to achieve a good cost/performance ratio. A simple deadlock avoidance mechanism for adaptive VCT routers known as "Bubble Routing" can lead to different high-performance router designs such as the one shown in [18], which will be the base router employed in this paper. Actually, the BlueGene/L supercomputer from

- *V. Puente, J.-Á. Gregorio, and R. Beivide are with the University of Cantabria, Avda. Los Castros s/n, Santander, 39005 Cantabria, Spain. E-mail: {vpuente, jagm, mon}@atc.unican.es.*
- *C. Izu is with the Department of Computer Science, The University of Adelaide, SA 5005 Australia. E-mail: cruz@cs.adelaide.edu.au.*

Manuscript received 20 Dec. 2001; revised 8 Nov. 2002; accepted 6 Jan. 2003. For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number 115604.

IBM employs an adaptive packet router based on this mechanism [2].

Once we provide sufficient buffer capacity and adaptive routing at a bounded cost, Head-of-Line Blocking (HLB) becomes the main limitation for achieving higher packet throughput. When the first packet in a FIFO queue blocks, any following packets do so as well. However, it is quite probable that one or more of those queued packets wish to reach a free output port. Thus, we will explore the use of buffer structures different to the standard input FIFO queues, which are the main source of HLB. Although many mechanisms have been proposed in the literature, their complexity usually results in higher base latencies that may counteract the gains in throughput. In this research, HLB is nearly eliminated by selectively using multiport output buffers to implement the most utilized queues of the router. This constitutes a clear application of Amdahl's Law. Other router alternatives avoiding HLB in a different way will be also considered in this paper for comparison purposes.

In short, the outcome of this paper is a High-Performance Adaptive Router (HPAR) that simultaneously uses packet adaptivity and an internal organization able to minimize the performance degradation due to HLB. We can find examples of routers that have employed adaptive routing [3], [20] and other examples providing some mechanism to avoid HLB [10], [23]. We will demonstrate in this work that adaptive routing with reduced HLB can be implemented in a realistic scenario at an affordable cost. To support the feasibility of our proposal, a detailed HPAR hardware implementation will be presented, evaluated, and compared against other router alternatives. The performance exhibited by the different solutions under study will be evaluated twofold: under synthetic traffic and executing real parallel application by means of different simulation environments.

The rest of this paper is organized as follows: Section 2 presents the basic motivations under this research and reviews some of the most related works. Section 3, which constitutes the kernel of this paper, presents the HPAR architecture basis, a suitable router organization, and an in-depth study of a particular hardware implementation. Section 4 presents a succinct but self-contained description of the router alternatives considered to compare with our proposal and the evaluation of their corresponding hardware costs. Section 5 presents a complete performance analysis of each router under synthetic and real workloads. Finally, Section 6 concludes the paper summarizing our main findings.

2 Motivations and Previous Works

The design of a high-performance network router is a trade off between the gains achieved by sophisticated routing and buffering mechanisms and their costs in terms of router's area and speed. In one extreme, we have the simplest designs which implement oblivious or Dimension Order Routing (DOR) and FIFO input queues, limiting network throughput to a fraction of its theoretical maximum [6]. Not more than 60 percent is achieved for random traffic, dropping below 30 percent for nonuniform patterns such as perfect-shuffle or bit-reversal permutations. On the other extreme, we have more sophisticated designs, such as the Chaos router [15], which achieve higher network throughput, but their complexity prevent them from being implemented in a real system. Nevertheless, when considering clock frequencies, the simple designs translate, in most cases, into higher absolute values of packets delivered per time unit as well as lower packet latencies. For these reasons, oblivious routers have been used in some commercial systems. Somewhere in between lies the appropriate router architecture under the current implementation technology. It should incorporate mechanisms that increase network throughput, provided that their gains offset the added implementation complexity. In this section, we describe the basic functions that must incorporate a high-performance packet router and their corresponding implementations that will determine its cost/performance ratio.

2.1 Deadlock Avoidance and Adaptive Routing

As we mentioned before, we focus in this research on *k-ary n-cube* networks, specifically on bidimensional tori in which packets are transferred under VCT flow control. These two features in conjunction with the selected routing mechanism will determine the nature of packet deadlocks in the system. It is perfectly known that the performance and complexity of any router is extremely sensitive to the methods employed to deal with potentially deadlocked packets.

An extension of virtual cut-through switching, known as Bubble Flow Control (BFC) [5], was successfully proven to avoid deadlock in deterministic tori with virtually no cost. A torus can be seen as a collection of unidimensional rings, which, under DOR, are visited in a specific order. BFC prevents the injection of a packet into any of these rings if it exhausts the ring's buffer space at the corresponding local router. Fulfilling this condition ensures the existence of a "bubble" (a free packet buffer) in any possible cycle of the network topology, then avoiding packet deadlock.

One of the best known methods to design deadlock-free adaptive networks is to add fully adaptive virtual channels to a given deadlock-free network, the latter constituting an escape subnet for any packet potentially deadlocked [9]. This subnet was chosen to be a BFC DOR virtual network in a previous adaptive router presented by the authors [18]. That router employed a second virtual channel per input link in which the packets were adaptively routed under VCT flow control. When using this switching technique, packets always try to travel through the fully adaptive virtual channels, changing dimensions only when blocked at a router. If all the adaptive paths for a packet are blocked, then the packet will use a BFC DOR escape virtual channel. Packets may change at any time from an escape channel to an adaptive one. In that work, we analyzed deterministic and adaptive versions of wormhole and BFC routers by designing the different alternatives at the VLSI level. It was demonstrated that the BFC adaptive routers outperform their wormhole counterparts both in packet latency and network throughput.

A switching technique based on this mechanism has recently been implemented in the BlueGene/L supercomputer [2]. Henceforth, we will use an evolution of a

BFC router, denoted as "Adaptive Bubble Router" [18], as the baseline for this new research and explore other design alternatives in order to improve its performance.

2.2 Avoiding Head-of-Line Blocking

Router performance depends not only on its functional characteristics, such as routing, flow control, and deadlock avoidance mechanisms, but also on its structural characteristics, such as the placement and organization of the internal buffers. A FIFO queue located at each input link is the most popular implementation as it requires single-port memories. Nevertheless, as mentioned before, this buffer organization exhibits Head of Line Blocking (HLB). A router suffering from HLB tends to exhibit poor performance. Under synthetic traffic loads with a random pattern for accessing the output ports, the router saturates at about 60 percent of its total capacity, thus wasting a significant fraction of its link bandwidth [11].

The solutions to this problem come from using non-FIFO input buffers and/or locating the storage space in a central queue or at the output links. Another common scheme is to use multiple input buffers, one for each output port, in the form of virtual channels [21]. The implementation of non-FIFO input queues implies the use of complex management hardware. For example, it is possible to use dynamic access queues (DAMQ) in order to service packets out of order as in [26]. Another possibility is to use a memory with multiple read ports to service one or more packets from the queue in a single cycle as in HIPIQS [22].

The use of a centralized queue shared between the input and output channel as in the IBM SP-2's router [23] entails a more efficient use of the buffer space, but requires multiple reads and writes per cycle. Placing the buffers at the output links still needs multiple writes to accept multiple incoming packets, but only one read per cycle. Although the efficiency in the use of the storage space is worse than in the centralized case, the silicon area needed diminishes considerably.

The viability of any of these buffer organizations will depend on the implementation complexity of the multiport memories associated to each design. A true multiport memory is extremely expensive and, usually, its application is limited to processor register files. Some proposals to provide high memory performance with lower cost include interleaved and wide memories. Such approaches have also been used in network routers such as the "Knockout Switch" [28] and the "Vulcan Switch" from IBM [24]. However, interleaved or wide memories have a serious drawback as it is necessary to wait for the reception of a wide word (in this context, a packet) before it can be written into memory. Therefore, its application would impose Store and Forward flow control or the use of an additional crossbar that allows the packets under low-load conditions to cut-through without using the memory [23].

The cost of the pipelined memory structure presented in [12] is similar to that of an interleaved memory, but, by exploiting the spatial locality of the accesses, it provides a fast and cheap VLSI implementation of multiport memories for packet routing purposes. The number of independent banks must be greater than or equal to the number of write ports. Writes and reads in such a memory are produced in a pipelined manner. The memory controller is quite simple because the write or read address in each one of the banks is that used by the previous bank in the preceding clock cycle.

In this research, we will investigate the use of this memory technology to implement some of our router's virtual channels as multiport output buffers, specifically the ones belonging to the adaptive virtual subnetwork. We will show how this technique greatly reduces HLB, leading to a very high packet routing performance. We will demonstrate the viability of our proposal by comparing it with a router having multiple virtual channels in the form of FIFO queues located at the input ports. This alternative represents our most directed contender in terms of both cost and performance.

3 High-Performance Adaptive Router (HPAR)

As mentioned before, our goal is to design a high-performance adaptive router for *k-ary 2-cube* networks, specially conceived to manage traffic generated by a standard CC-NUMA multiprocessor. The reactive traffic properties of CC-NUMA machines can give rise to application deadlock due to the limited capacity of the consumption queues at the network interface. The mechanism usually employed to avoid this kind of deadlock is to use two different virtual networks for request and reply traffic. This solution has been adopted in several systems such as the SGI Origin [16] and the Cray T3E [20]. In this work, we will also consider a router managing two adaptive virtual networks. The implementation of both networks is achieved by means of three virtual channels: one adaptive channel shared by request and reply packets and two separate DOR escape channels managed under a restricted injection policy (BFC) to avoid packet deadlock in both virtual networks. More evolved systems generating other traffic classes would need additional virtual networks. The router proposed in this work could consequently be adapted to cope with additional packet classes.

3.1 HPAR Architecture

A preliminary version of our HPAR, in which all its temporary storage has been associated to the output links, can be seen in Fig. 1. For simplicity, this figure only shows two of the four input/output transit modules. There is a single packet buffer per input channel in order to dissociate the internode flow control from the Routing Unit (RU). Each output link has three multiport buffers, one per virtual channel, plus a virtual channel controller. Adding the reply and request delivery queues, it would require 14 multiport memories in total. The number of writing ports would depend on our "bubble switching mechanism" [18], ranging from six for the +X/-X deterministic output queues to 11 for the +Y/-Y adaptive output queues. Note that this organization not only eliminates HLB, but also the need for arbitration as all incoming packets could be written in their output buffers at once. However, the complexity of the resulting structure makes this proposal unfeasible.

It is clear that we need to compromise our output buffer structure implementation in order to produce an affordable router. Fig. 2 shows that the population of the adaptive

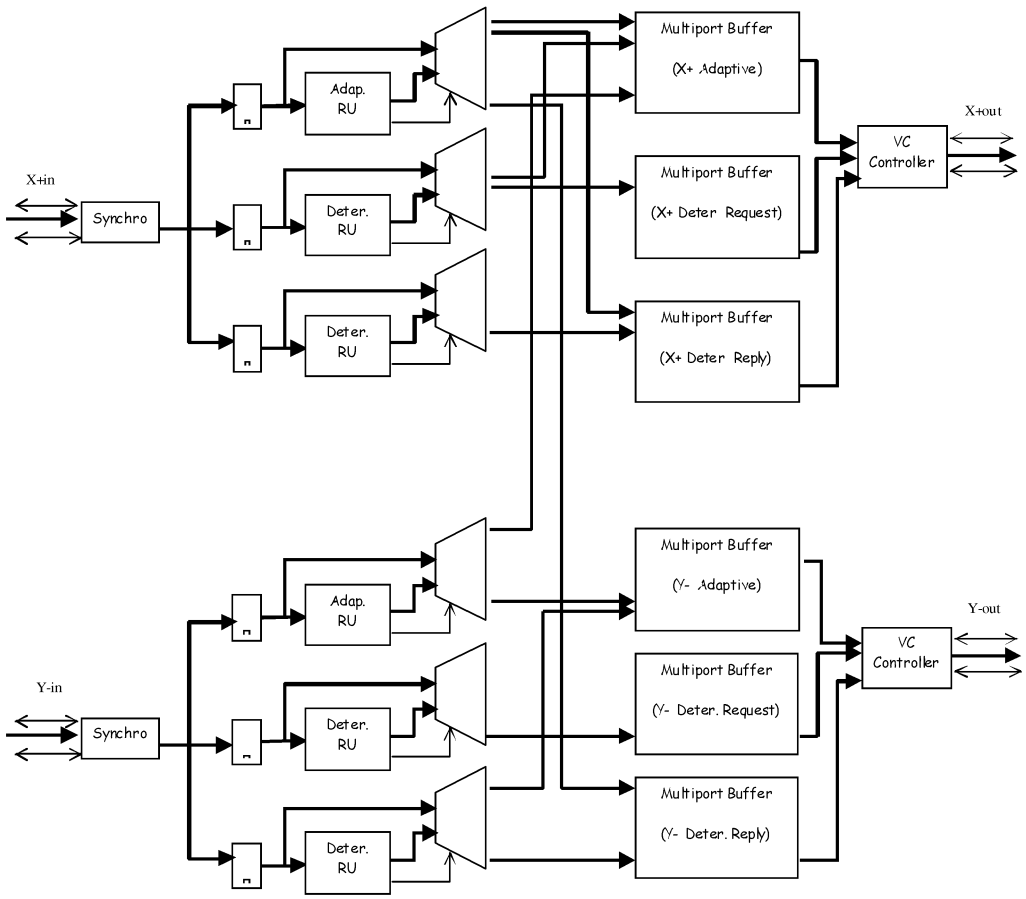


Fig. 1. Bubble Adaptive Router with output buffers (preliminary HPAR version).

queues in a 16 x 16 torus under heavy random traffic is three times higher than their deterministic counterparts when using a conventional input-buffer adaptive bubble router [18]. This happens because our switching mechanism selects adaptive paths whenever possible. Thus, at low loads, most traffic exclusively uses the adaptive paths and the need for the escape's paths is minimal. At saturation approaches, more packets will resort to escape paths. However, as BFC is a restricted injection mechanism, it prevents packets from filling up the escape queues. It is well known that by improving the most frequent case of router's operation we will obtain high performance at the lower cost Thus, we can limit our effort to reduce HLB by using only output buffers for the adaptive paths.

Taking this behavior into account, a more feasible version of HPAR is shown in Fig. 3. This approach uses output multiport buffers only for adaptive channels and input FIFO queues for escape channels. The number of multiport memories required for a bidimensional torus is five: four for the adaptive channels X+, X-, Y+, and Y- plus another one for the delivery channel. The connection between input and output modules requires a crossbar of 14 x 14, which is a manageable size to be implemented without input or output multiplexing. There exist 12 input/output crossbar

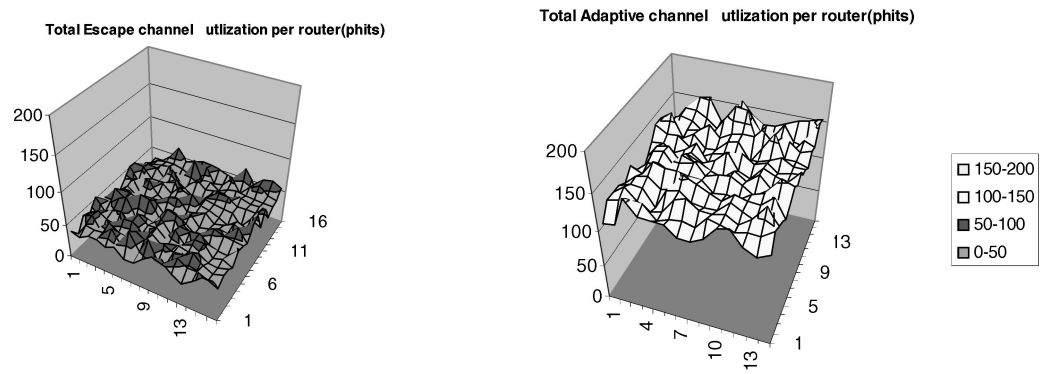


Fig. 2. Population count of escape and adaptive queues close to saturation.

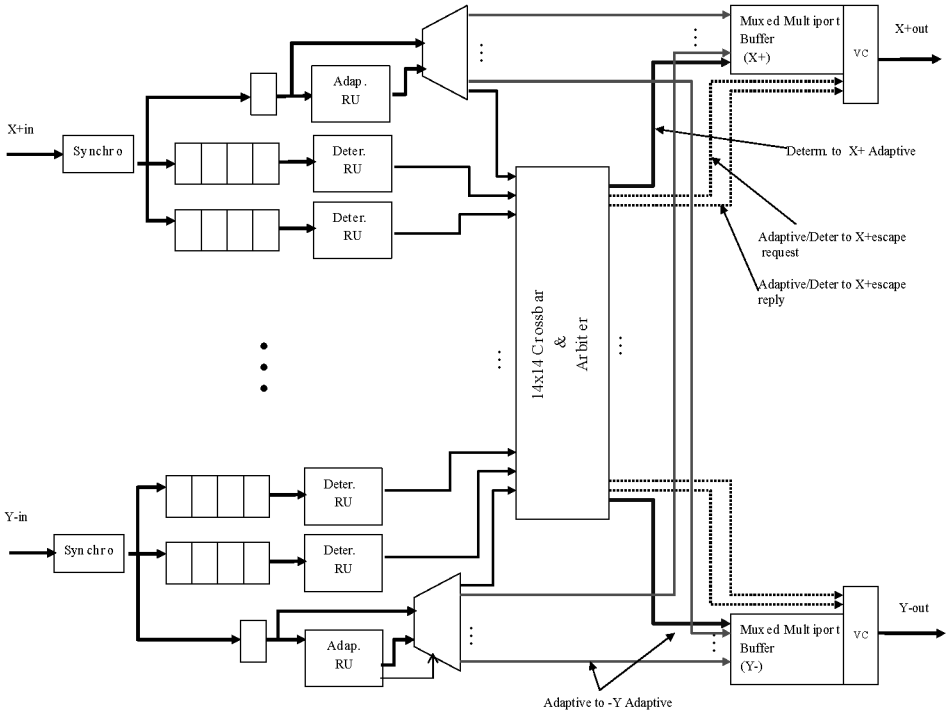


Fig. 3. High-Performance Adaptive Router (HPAR).

terminals to manage transit packets and two additional input/output crossbar terminals to manage the injection and consumption processes from/to two separate request and reply queues.

In addition, we reduce the number of write ports by allocating a single write port for each adaptive input channel (three, as packets use minimal paths) and another one for all other traffic routed toward that adaptive channel. Therefore, each adaptive input channel has its own path to forward packets to their selected adaptive outputs. Packets arriving at the adaptive channels that can continue advancing along them will reach the router output without passing through the crossbar. Thus, the most common packet transit case does not need arbitration. On the contrary, packets routed from any escape channel and new packets injected at that node moving toward an adaptive channel must compete to use a single shared write port. Similarly, all packets coming from an escape channel and requesting another escape channel must compete to acquire the corresponding crossbar's output. As escape and virtual channels share the physical link, that crossbar's outputs and the read port of the multiport adaptive buffer are connected to the corresponding virtual channel controller (VC). The delivery buffer also has four write ports corresponding to the four adaptive inputs, while packets arriving through escape channels reach directly the network interface via the escape delivery channels.

In an adaptive router, the selection function chooses the output channel from the set of profitable ones provided by the routing function. A dynamic selection function can balance network occupancy and, therefore, enhance its maximum throughput. From the different alternatives of dynamic selection, we have selected the MAX-CREDITS policy that gives preference to the less populated output channels [27]. The main problem associated with dynamic selection functions is the cost involved in knowing the traffic conditions. When the buffers are located at the input ports, it is necessary to collect occupancy figures from the neighboring nodes or to use communication protocols based on credits, which require additional hardware. Nevertheless, when the adaptive buffers are located at the output ports, such as in HPAR, the implementation of a dynamic selection function is straightforward.

3.2 HPAR Structure and Organization

In the HPAR architecture, each output link has an output buffer implemented by means of a multiport pipelined FIFO which is similar to the one presented in [12]. A basic example of this output buffer, using two write ports and two phits per packet, appears in Fig. 4. Two modules in this circuit can be distinguished. One of them, framed in a dotted box in this figure and denoted as Adaptive Multi-Input Block, implements the memory pipeline. The other module is composed of the memory banks themselves.

The total number of cycles required to pass through this memory under low load is two cycles instead of the three cycles needed in [12] because it is not necessary to manage several read ports. This favors latency-sensitive applications. Under higher traffic load, multiple packets arriving at the same cycle, two in this example, may want to write into the first memory bank. The Adaptive Multi-Input Block pipelines the corresponding writes: the second effective write will be delayed one clock cycle in order to adequately fill the memory pipeline. In no case are packets or phits lost. In addition, the pipelined FIFO is able to read and feed the output link at its maximum operation frequency. In this basic design, the minimum number of independent banks required for a correct memory operation is the maximum of

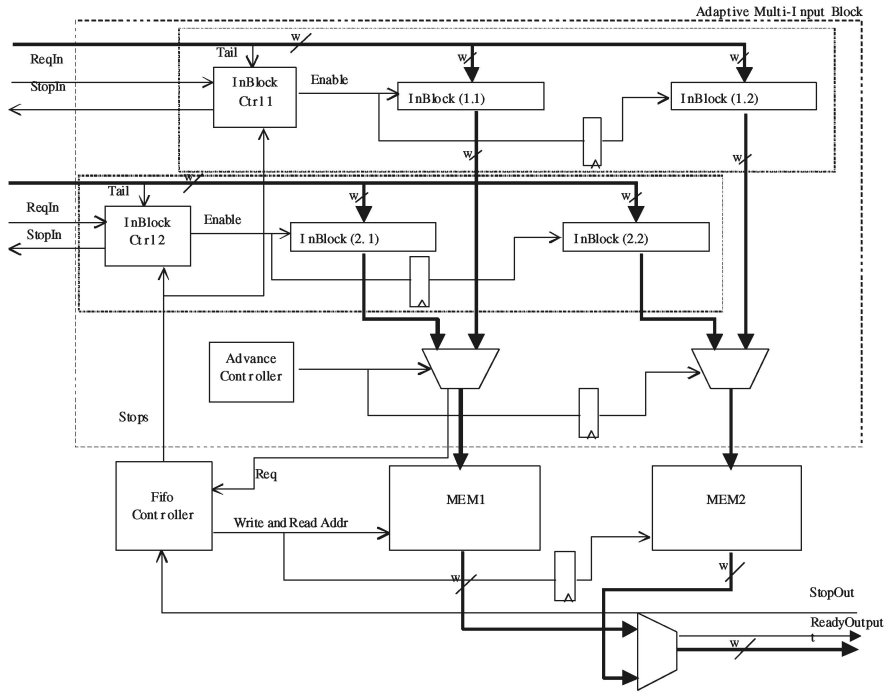


Fig. 4. Pipelined FIFO multiport memory (multiport buffer).

the two following quantities: the maximum packet size in words or phits and the number of write ports. In our design, one packet is stored in a single memory line. Therefore, the number of memory banks increases with packet length and so does it cost.

In order to limit cost, we have assembled two phits into one wider word, halving the number of banks. As the maximum packet length is 10 phits, which is dictated by the characteristics of our emulated CC-NUMA machine, each memory has five independent banks. It should be noted that, by the time the routing decision is taken, there are already two phits at the input FIFO, so this scheme does not increase router latency. Obviously, the header phit is advanced to the corresponding routing unit without waiting to receive the second phit of the word. To support this feature, HPAR has asymmetric FIFO memories at the input links which use a read bus twice as wide as the write bus. Consequently, phit serialization will be needed at the multiport memory output. A diagram of this solution can be seen in Fig. 5.

As we have seen above, the multiport pipelined output buffers are designed for a fixed packet size, set by the number of banks (or a multiple of it). However, CC-NUMA machines exhibit a bimodal message length distribution, having both short command messages and larger data messages. As both the request and reply traffic share the same adaptive channels with a short-to-long message ratio close to one, this would lead to significant memory fragmentation. This would not only use the buffer space inefficiently, but could cause lower link utilization as well. To solve this problem, we add one more memory bank for short packets with its additional memory controller, as

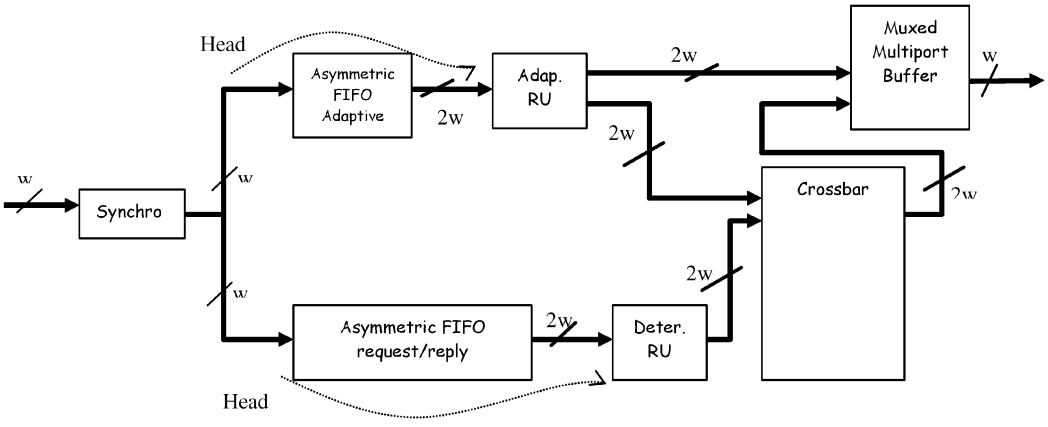


Fig. 5. Assembling phits to decrease the multiport FIFO memory complexity.

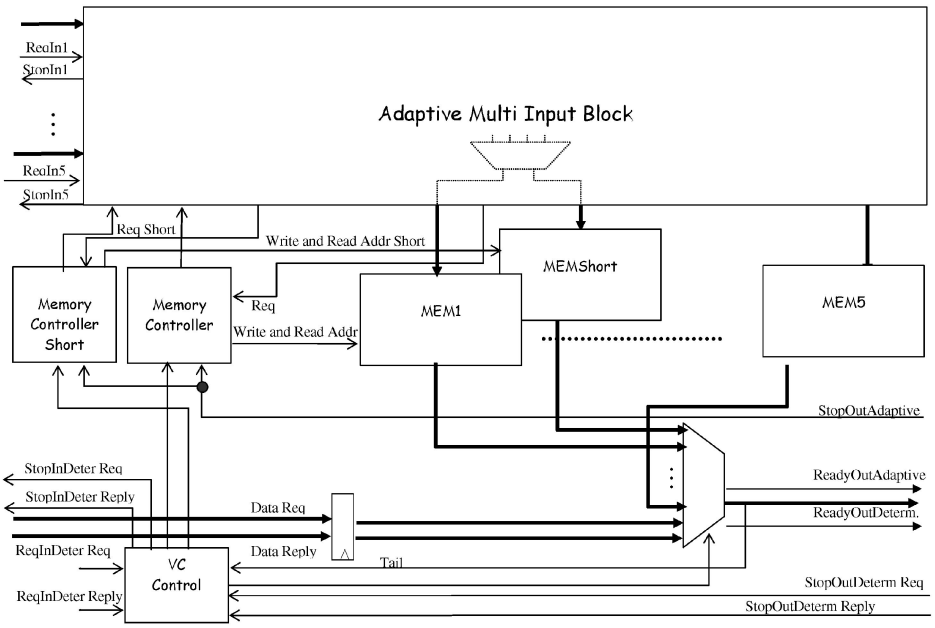


Fig. 6. Pipelined Multiport FIFO including virtual channel controller.

reflected in Fig. 6. The Adaptive Multi-Input Block also provides the path to write into that additional bank. The outputs from the six independent banks will be multiplexed into the same physical link.

Finally, we need to arbitrate the access to the physical link between the adaptive channel (the outputs from the multiport memories) and the two deterministic channels. This arbitration adds only one simple multiplexing stage to the output path. Furthermore, as the multiport pipelined memories already have a multiplexing stage among the memory banks, we can incorporate the two escape channels into this same multiplexer which is driven by the VC controller. Although this multiplexer is on the critical path of the output stage, most decisions can be taken in advance; while one packet uses the output, the next outgoing packet can be chosen. If the output port is free and two or more packet headers arrive simultaneously, the VC controller will give way to the adaptive channel. Thus, packets from escape channels will always have to spend a cycle checking the port availability.

3.3 HPAR Implementation and Hardware Costs

In order to estimate the hardware costs, our router has been described at register-transfer level in VHDL. Starting from this representation, a logic description of each router's component has been obtained using the synthesis tools from the EDA Synopsys 1999.10 suite [25]. The design has been implemented in 0.25 μm technology using five metal layers from the UMC foundry. Employing these tools at the logic implementation level, we have extracted the router's costs, both in delay time and silicon area. It must be highlighted that we are interested in the study of the HPAR behavior at the architectural level and in the comparison of its features with those of other router designs. Going down toward the physical level would unnecessarily increase the complexity of this analysis with a very limited effect on the conclusions that can be extracted from the logic implementation level.

As we only want to compare different router alternatives, we will assume no channel pipelining. In this way, the most important contribution to packet latency corresponds to the router pass time. This constitutes the worst-case scenario for complex router designs, such as HPAR, because the impact of the router delays dominates over the link delays. In our case, the maximum wire frequency will determine the lowest cycle time for any router design. If any module of the router was not able to reach this frequency, it would be necessary to split it, increasing the number of stages of the corresponding pipeline. The greater the complexity of the router, the more pipeline stages and, therefore, the higher the latency.

We will assume a channel width of 32 bits and a frequency of operation of 333 MHz, providing a maximum bandwidth of 1.3 GB/sec per direction. This number adequately fits among the practical values exhibited by current multiprocessors. Typical values range from 14 bits per link at 375 MHz in the Cray T3E to 20 bits per link at 200 MHz in the SGI Origin. In fact, the aggregate bandwidth of HPAR will be 14.6 GB/sec. This is close to the value claimed for systems like the Alpha 21364 [3], which also has 32-bit channels and router bandwidth between 10 and 15 GB/sec.

The first step to carry out the synthesis process of HPAR is to describe in VHDL each component at the register-transfer level. The router pipeline has been established by separating each module in different stages with clearly different functionalities. These stages are: synchronization, temporary storage, request, and arbitration. Therefore, the router will initially have a latency of five cycles. Results from the first synthesis step will determine if the critical path of each module fulfills the frequency limit imposed by the system. The resulting pipeline structure that achieves

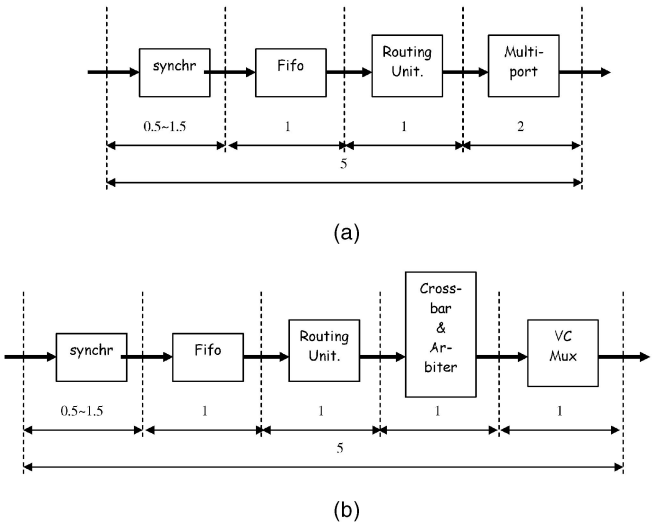


Fig. 7. Pipeline structure of HPAR: (a) Adaptive to adaptive switching. (b) Deterministic to deterministic and adaptive to deterministic switching.

these temporal requirements is shown in Fig. 7. As we can see, the pipeline for the most common case, adaptive to adaptive channel switching, employs five cycles. The deterministic to deterministic channel switching and the adaptive to deterministic switching also employ five cycles. Finally, the deterministic to adaptive transit, not represented in the figure, consumes six cycles.

Modules facilitated by the Synopsys DesignWare components library have been employed to synthesize all the router elements (FIFOs, static memories, and controllers). In this way, we could choose among an abundant list of fast implementations based on the requirements of each module. The hardware costs of the temporary storage depend both on its size and the given packet length. Given a maximum packet length of 40 bytes, the memory sizes that achieve a suitable performance are those shown in Table 1. The total buffer size of our proposal is about 2 KB, which is perfectly acceptable for 0.25 μm technology. Larger buffer sizes do not significantly improve router performance, but it may even increase the router pass time. Moreover, the cycle time of a FIFO module is a function of its depth. In our router, as mentioned above, asymmetric FIFOs with double width and half depth were used to reduce the number of banks of the output multiport memories.

TABLE 1
Buffer Sizes for HPAR Expressed in Phits (Bytes)

	Buffer Sizes phits (bytes)
Fifo Injection Reply	**40(160) x 1**
Fifo Injection Request	**8(32)x 1**
Fifo Deterministic Reply	**40(160) x 4**
Fifo Deterministic Request	**8(32)x 4**
Adaptive Input Buffer	**10(40)x 4**
Adaptive Multiport Memory	**40 (160)x 4**
Consumption Multiport Memory	**40 (160)x 1**
Total	**480 (1920)**

TABLE 2
Main Synthesis Results for HPAR

	Critical Path (NS)	*Area (MM^2)*
Synchro. U.	1.08	0.011(x4)
Fifo Adaptive	2.91	0.132(x4)
Fifo Determ. Rep.	2.98	0.412 (x5)
Fifo Determ. Req.	2.84	0.106(x5)
CrossBar	2.93	0.446 (x1)
Routing U. Adap.	2.75	0.039 (x4)
Routing U. Determ.	2.60	0.0336 (x10)
Multiport Memory	2.97	1.031(x5)
TOTAL	**3**	**9.27**

Continuing with the synthesis process, a logic-level implementation for each module of the router was obtained. The main characteristics, in terms of time and silicon area, for this implementation are shown in Table 2.

4 Alternative Router Designs and Comparative Hardware Costs

In order to assess the cost/performance ratio exhibited by HPAR, we must compare these figures not only with our baseline router but also with the other alternative design to reduce HLB. This section describes these two additional routers and their estimated hardware costs.

4.1 Bubble Adaptive Router

The base router employed in this paper is an evolution of our original bubble adaptive router. This router can be seen as the one achieving the highest performance among the range of adaptive routers having FIFO queues located at the input ports [18]. By comparing HPAR with this router, we can see the gains due to reducing HLB.

The Bubble Adaptive router, or BADA for short, has three virtual channels per link, as show in Fig. 8. In the same way as with HPAR, BADA has a shared input FIFO to adaptively manage "request" and "reply" CC-NUMA traffic and two separate escape queues managed under DOR to assure deadlock-free communications. Synchronization units (Synchro), Routing units (RUs), and the corresponding crossbar unit complete the router structure. For the sake of clarity, the figure only shows two of the four input modules. Besides, only the circuit data-path is shown.

4.2 Bubble Adaptive Router with Multiple Virtual Lanes

The second alternative is a variation of the BADA router, called BADAVL (BADA with Virtual Lanes), in which the

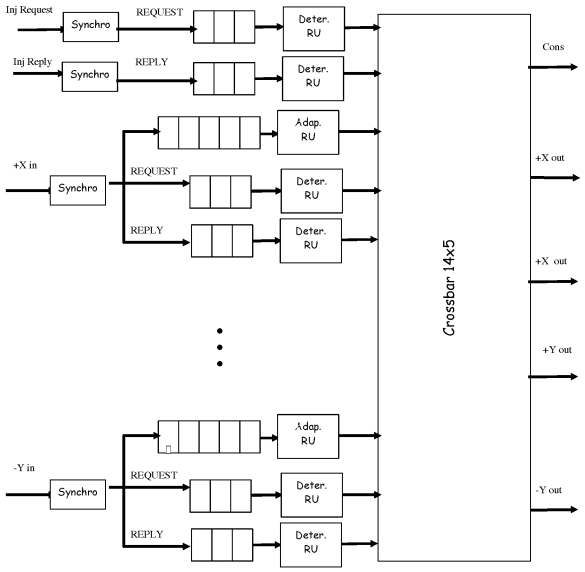


Fig. 8. Structure of a Bubble Adaptive Router (BADA).

adaptive paths have been split into four virtual lanes.[1] The use of virtual lanes located at the input ports to avoid HLB is a standard practice in the industry [8]. When a packet blocks at any given virtual channel, it does not prevent packets at the other virtual lanes from advancing. Each virtual lane has room to store one packet and any packet traveling through an adaptive path can use any of the four lanes, so the buffer space will be efficiently used. This design alternative will show the effects of using standard HLB avoidance strategies when combined with adaptive routing.

When using BADA and HPAR, the adaptive channels share "reply" and "request" traffic and this introduces buffer fragmentation. This effect would be magnified when using BADAVL. If the adaptive channels were shared between reply and request virtual networks, the storage utilization of each lane would fall because its individual buffer can be exhausted with only one short message. Thus, to avoid this negative effect, two independent virtual networks have been implemented: one for request and another for reply traffic. Fig. 9 describes the basic building blocks of this router. Note that the addition of virtual lanes does not require extra signaling in the communication protocol between neighboring routers.

Obviously, as we add virtual lanes the number of inputs to the crossbar increases. Consequently, the crossbar control logic has more complexity. The approach taken here is to multiplex these different lanes at the crossbar input [20], keeping the cycle time and silicon area within manageable limits. A Routing Unit (RU) per input link selects the output ports for each incoming packet and arbitrates amongst them to reach the crossbar. Since the flow control employed is VCT, packet-level multiplexing will be used. It should be noticed that this design provides a simple buffer structure at the cost of increasing the complexity of the switching fabric.

1. Note that the term "lanes" is used to describe a set of virtual channels which are indistinctly used by the selection function, in contrast with the more generic term "virtual channels" in which the routing function may select one channel or another depending on different network conditions.

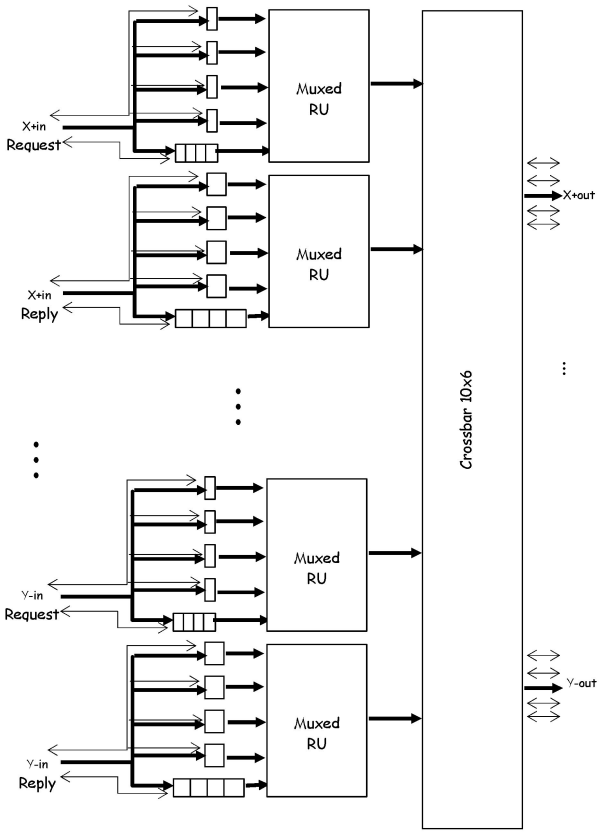


Fig. 9. Structure of the BADAVL router with independent request and reply virtual networks to avoid fragmentation.

4.3 BADA Routers Implementation and VLSI Costs

To carry out a fair comparison of our proposal with respect to its two counterparts, a specific implementation of both routers has been developed following the same methodology employed for HPAR. For a fair comparison, we have fixed the same router cycle for all the alternatives under study. The pipeline structures for both BADA routers are shown in Figs. 10 and 11.

It must be noted that the Multiplexed Routing Unit of the BADAVL router includes a pipeline stage more than the simpler BADA router. BADAVL needs an extra round-robin-based arbitration stage to decide which channel sends its request to the crossbar. The implementation, in a single clock cycle, of the round-robin scheme together with the corresponding arbitration process is not possible under the technology used.

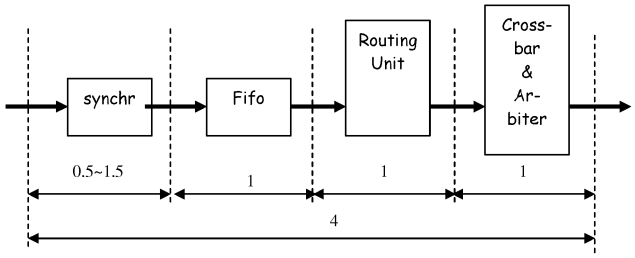


Fig. 10. Pipeline structure of BADA router.

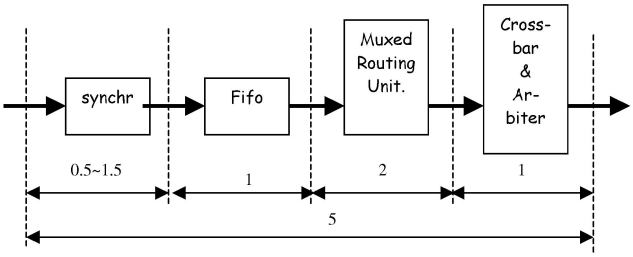


Fig. 11. Pipeline structure of BADAVL router.

Again, in order to make the comparison fair, all the routers will have the same storage capacity. A summary of the buffer sizes and their distributions in the alternative router designs is shown in Table 3.

Finally, the measures obtained from the VLSI synthesis process of both routers can be seen in Table 4. Further details related to these alternative router implementations can be found in [18].

The main drawback for the routers with HLB avoidance mechanisms is the increase in their base latency. Both, BADAVL and HPAR require, at least, five stages to pass through them. It must be highlighted that, in some switching cases, HPAR employs more cycles than BADAVL.

This section ends with comparing the hardware costs of the three routers considered in this research. Obviously, the BADA router with FIFO input queues is the cheapest one in terms of the required silicon area. In absolute terms, the increase in area for the routers with HLB avoidance mechanisms is above 50 percent. Nevertheless, our proposal requires 5 percent less area than the BADAVL router. To put this number in context, the Alpha 21264A occupies 225 mm^2 implementing 15.2 million transistors with 0.25μm technology [13]. This design rule is the same as the one we used in our design process. Thus, the HPAR would represent only around 4 percent of the processor occupied silicon area, compared to 2.6 percent for the simplest BADA router. Whether we integrate the router in the processor chip or not, the additional cost is quite low; more so when considering the performance gains it entails, as we will see next.

TABLE 3
Buffer Sizes for Each Router in Phits (Bytes)

Buffer Capacity phits (bytes)	Routers	
	BADA	BADAVL
Fifo Injection Reply	40(160)x1	40(160)x1
Fifo Injection Request	32(128)x1	32(128)x1
Fifo Deterministic Reply	40(160)x4	40(160)x4
Fifo Deterministic Request	32(128)x4	24(96)x4
Fifo Adaptive Channel	40(160)x4	4*10(160)x4
Fifo Adaptive Channel Req	--	4*2(32)x4
Total	520 (2080)	520 (2080)

TABLE 4
Main Synthesis Results for BADA and BADAVL Routers

	BADA		BADAVL	
	Critical Path (NS)	Area (MM^2)	Critical Path (NS)	Area (MM^2)
Synchro	1.08	0.011(x 4)	1.08	0.011(x 4)
Fifo Adapt	2.96	0.381 (x 4)	2.82	0.500 (x 4)
Fifo Adapt Req	--	--	2.75	0.085 (x 4)
Fifo Determ Rep	2.96	0.381 (x 5)	2.96	0.381 (x 5)
Fifo Determ Req	2.84	0.332 (x 5)	2.84	0.228 (x 5)
CrossBar	2.96	0.248 (x 1)	2.95	0.169 (x 1)
RU Adap	2.60	0.0336 (x 4)	2.90	0.0351 (x 10)
RU Determ	2.60	0.0336 (x 10)	--	--
TOTAL	**3**	**5.85**	**3**	**9.70**

5 Performance Analysis

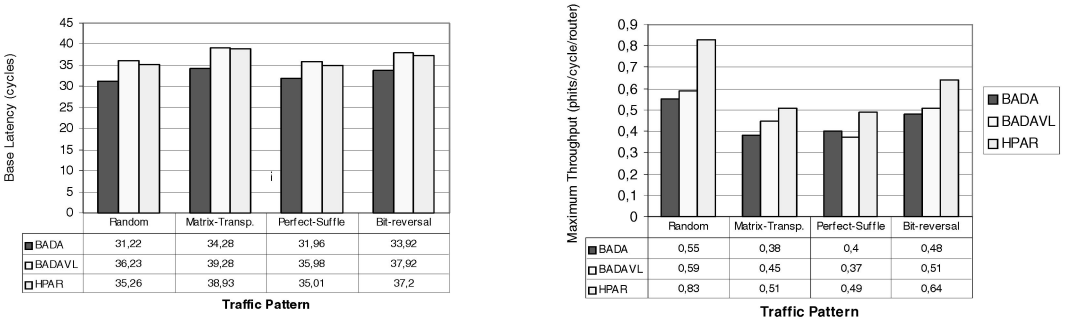
This section presents a detailed performance analysis of the three routers described above in order to establish the potential advantages of HPAR when it is used in a CC-NUMA multiprocessor. This analysis takes into consideration the hardware costs associated to each router established in the previous sections.

First, we will compare the performance exhibited by three alternative interconnection networks using the different routers under a range of synthetic workloads. Second, we will compare the three networks in the context of a state-of-the-art CC-NUMA multiprocessor running real workloads.

5.1 Performance Analysis under Synthetic Traffic

The simulation environment employed in this study is based on SICOSYS (SImulator of COmmunication SYStems) [19]. This simulator allows us to take into consideration most of the VLSI implementation details with high precision, but with much lower computational requirements than hardware-level simulators. The maximum error observed with respect to a standard hardware simulator is around 2 percent, providing, in all cases, pessimistic estimations [19].

Although we begin our analysis using synthetic loads, in order to adequately model the traffic of a CC-NUMA machine, we have considered a bimodal distribution of packet lengths with a fixed length of 2 phits for short messages and 10 phits for the long ones. The real ratio of short/long messages depends on the network and application characteristics. To simplify this preliminary study, we don't take into account coherency messages; hence, for each request (short message), the machine answers with a reply (long message). Therefore, the probability of generation of the two classes of messages is set to 0.5. With respect to the destination pattern, we have considered random as well as three widely used permutations: transposed matrix, bit-reversal, and perfect-shuffle. It is well known that this type of traffic only approximately models the complex behavior



(a) (b)

Fig. 12. (a) Base latencies (0.05 percent applied load with respect to the network bisection). (b) Maximum throughput.

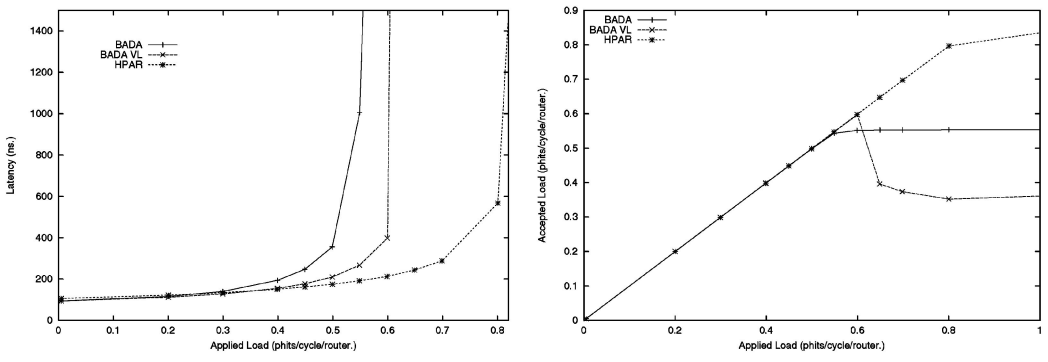
of the applications. Nevertheless, the network response can be manageably observed with network loads ranging from zero to its saturation point. This provides an insight into network performance in the two extreme cases: latency sensitive applications that generate low loads and throughput-limited applications that put high traffic pressure on the interconnection networks.

As we previously mentioned, the topology is set to a 64 2D-torus network. Fig. 12 shows base latencies and maximum throughputs for the different routers and Fig. 13 shows the network behavior as a function of the load under random traffic. The base latency differences are small and reflect the values of pass-through delays for each router. On the contrary, substantial differences can be observed in their maximum throughputs. Of the two proposals to reduce HLB, HPAR clearly outperforms BADAVL, in spite of having fewer area requirements. Besides, the throughput achieved by HPAR managing random traffic is nearly twice the value exhibited by the conventional BADA router. This peak performance is very close to the ideal value in which each network node can consume one phit per cycle. In the same situation, the BADAVL router exhibits only a slight throughput increasing.

Although adding virtual lanes reduces HLB and increases peak throughput (see BADAVL versus BADA), network performance degrades above its saturation point. The multiplexing stage before the crossbar forces the deterministic channel to compete with the other four adaptive channels to access the crossbar and long waits result from it. After the load reaches its peak, more and more packets will resort to their escape routes, which will only increase network congestion even more. This deficiency could be alleviated using a more complex arbitration policy than round-robin or implementing a not-multiplexed crossbar, both solutions having much higher cost.

To complete this evaluation, we have also considered the impact of varying the ratio of short to long messages. As the differences in base latency are negligible, we will only compare the values of the maximum achievable throughput for each network under the previous traffic patterns, which are shown in Fig. 14. It can be seen that HPAR performance does not degrade too much, even for large short/long ratios, which is proof of its good response under high fragmentation scenarios. The use of multiport memories and the added improvements to favor the most frequent switching case are responsible for this behavior. For low-to-medium short/long message ratios, BADAVL also exhibits a constant behavior, but, when the amount of short messages increases, its performance quickly falls. In some cases, this performance degradation is close to 15 percent. The improvements in performance in the medium ratio values are due to better buffer utilization. In this case, the use of each virtual lane is more balanced.

Finally, it can be seen that the performance exhibited by the BADA router is lower in all these experiments. In some



(a) (b)

Fig. 13. Latency and throughput evolution for an 8 x 8 torus under random traffic.

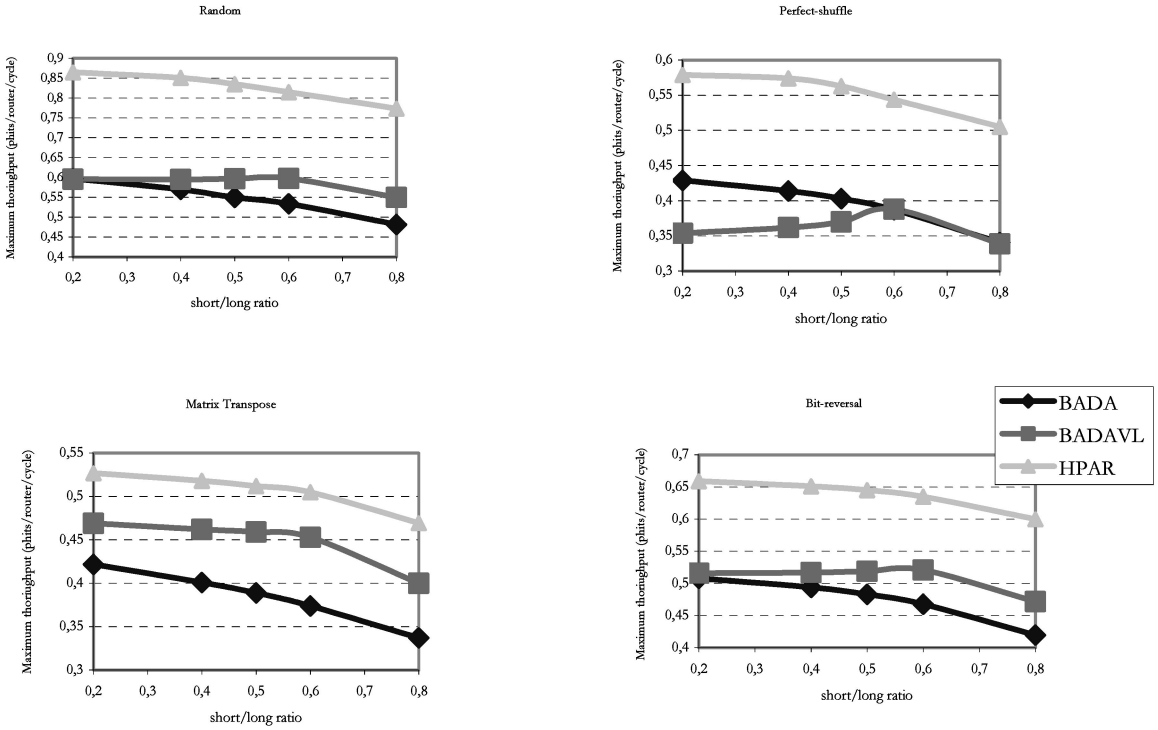


Fig. 14. Maximum achievable throughput with different ratios for short/long messages and different traffic patterns.

scenarios, the effect of varying short/long messages ratios shows degradations up to 20 percent. This is a consequence of the increase in contention in the router. When the average packet length decreases, the arbitration processes inside the router are more frequent and, consequently, the congestion increases. HPAR is less sensitive to this effect as packets using adaptive paths require no arbitration. In a CC-NUMA multiprocessor, some applications exhibit high short/long ratios; thus, it is crucial to adequately support short message traffic.

5.2 Performance Analysis under Real Workloads

In order to test the networks under a more realistic scenario, an execution-driven simulation process has been carried out. The integration of our network simulator, SICOSYS, into the RSIM simulation environment [17] provides a powerful tool to emulate a complete state-of-the-art CC-NUMA machine.

We have initially set the simulation configuration parameters among the different levels of the memory hierarchy as in [4]. In that paper, 1 GHz processors implemented in 0.18 μm technology were used, establishing, in consequence, all the different memory access times. However, our routers have been implemented using 0.25 μm technology, so we scale the processor frequency down to 666 MHz. This is, in fact, the value reached by processors developed with the same channel length, such as the Alpha 21264A [13]. The access latencies to the different levels of the memory hierarchy have been modified proportionally. In short, we could model our multiprocessor system under any given technology by adequately tuning the configuration parameters of both RSIM and SICOSYS simulators.

Due to the limitations imposed by the complexity of the execution-driven simulated system, 16KB L1 cache and 64KB L2 cache have been used. The benchmarks employed have been tuned according to these cache sizes. In both cases, the cache line size is 32 bytes. The basic command messages traveling through the network have been fixed at 8 bytes. Therefore, as mentioned before, the command-message and the data-message are, respectively, 2 and 10 phits long.

To carry out a realistic evaluation, we fed our simulation platform with three applications selected from the SPLASH-2 suite: Radix, FFT, and LU. These three applications were selected because they have significant communication demands and each one represents a different case of network load. Radix puts high pressure in terms of volume of information to be handled by the network, while exhibiting a practically uniform communication pattern in many phases of its execution. FFT, however, applies medium pressure on the network, but the communication pattern has low spatial locality. Finally, LU applies lower pressure on the network, but it gives rise to hot spots in localized zones of the system. These three examples will allow us to explore the effectiveness of each router under different conditions.

The problem size for FFT is 64K double complexes. This is the default problem size established in [29]. Due to the high demand for computational resources, the problem size for LU has been reduced from its default size of 512×512 to 256×256. The problem size for Radix has also been reduced from one million integer keys to a half-million using a radix of a half-million. For the emulated system size, these changes do not affect the accuracy of the results. The capacity for the different levels of the memory hierarchy was chosen in such a way that the results obtained are significant for the selected problem sizes and

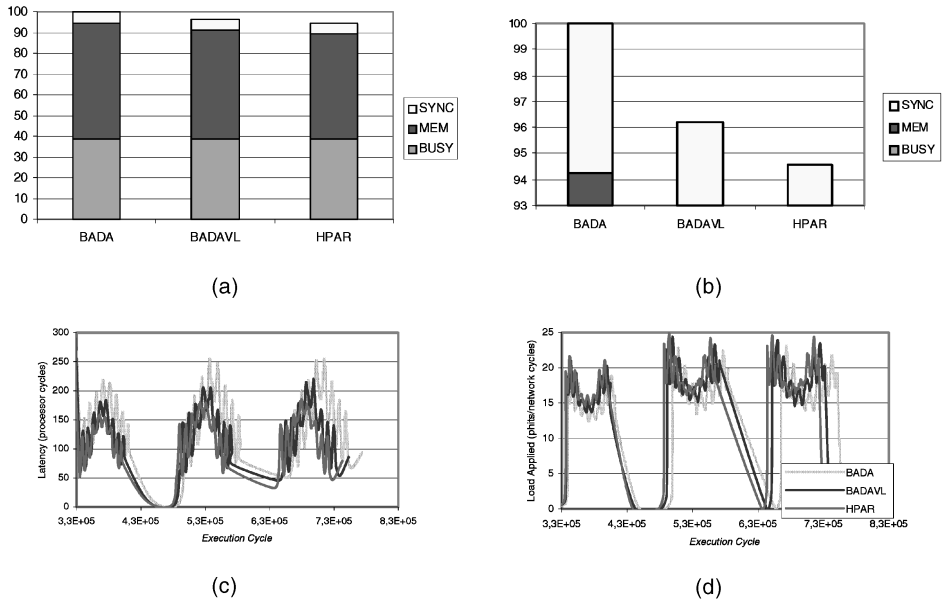


Fig. 15. (a) Normalized execution time for FFT with 64 processors (8 x 8 Torus) managing 64K double complexes. (b) Zoom. (c) Average latency evolution. (d) Average throughput evolution.

for the dimensions of the global system. Other SPLASH-2 applications were not considered because they do not add any valuable information. In some cases, they exhibit similar characteristics and, in others, the interconnection network has no significant impact on its performance.

Figs. 15, 16, and 17 show the system behavior when running the three applications over three networks using BADA, BADAVL, and HPAR routers. Each figure includes the normalized execution time (and a close-up of it) and the network behavior in latency and throughput as it evolves during the program execution. The average latency of the remote accesses is measured in processor cycles and the throughput is expressed in phits per network cycle. In this way, we can see the network performance impact on the remote access latency and the network utilization level throughout the application execution. Note that the theoretical limit for the network load is 64 phits/network cycle (one phit per node per cycle). Only the most interesting cases are shown.

Remember that the CC-NUMA testbed is identical for each experiment except for the packet routers, all of them implementing adaptive routing and having the same clock cycle. As Fig. 15d shows, the network load fluctuates from phases of heavy load to others of very low load. At low loads, all routers behave similarly because the differences on latency are minimal. The main differences are due to the ability of HPAR and, in some degree of BADAVL, to sustain higher peak throughput. This is why HPAR completes the heavy load phases ahead of its contenders. This is also true when running Radix, as shown in Fig. 16d, in which the application loads range from high to medium loads. In this case, latency values differ from one network to other, as shown in Fig. 16c. HPAR exhibits lower latency at high and medium loads, but BADA has lower values at low loads.

There is a clear correlation between the peak throughput reached under RADIX (70 percent for HPAR, 64 percent for BADAVL, and 60 percent for BADA of the theoretical maximum) and the behavior observed under synthetic random traffic. The peak throughput observed in FFT (around 40 percent) shows a similarity to the values observed under synthetic permutations. The differences among the three networks' peak throughput when running real applications are not as considerable as under synthetic loads. Note that CC-NUMA traffic is reactive in the sense that, as network congestion slows down the pending replies, it also reduces the number of incoming requests. So, maximum load levels oscillate around the network peak value.

In short, the incorporation of HLB avoidance mechanisms improves network throughput. In the highest load phases, the network can manage heavier traffic. Hence, the duration of these phases is smaller and, therefore, their execution times are shorter (see Fig. 16d for example). In fact, the throughput enhancements of both routers with HLB reduction compensate for their slight increases in base latency, as can be clearly seen in the Radix benchmark. HPAR reduces the execution time in respect to BADA by up to 10 percent. Besides, it always outperforms the BADAVL router in spite of using 5 percent less silicon area. Note that both routers share the same clock frequency and even, in some cases, HPAR employs more stages to pass a packet through it. It must also be noted that the whole CC-NUMA system is a complex architecture and its performance not only depends on the interconnection subsystem. Further tuning of other subsystems would increase the significance of the network performance in the total execution time.

The previous scenario represents a single kind of system workload, i.e., numerical programs. Nevertheless, there is a larger range of applications for this class of popular CC-NUMA multiprocessors. An important set of these applications, such as OLTPs and DSSs, can put more pressure on the interconnection network than the numerical ones due to their scarce data locality [4].

6 Conclusions

In this paper, the design and evaluation of a high-performance adaptive router (HPAR) suitable for next

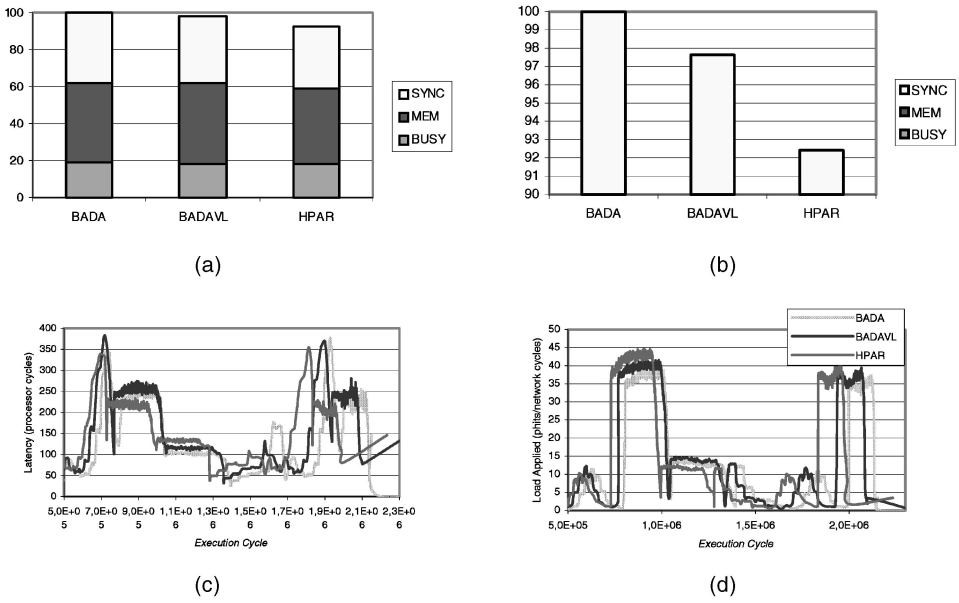


Fig. 16. (a) Normalized execution time for Radix with 64 processors (8 x 8 Torus) for 512K integer keys and 512K Radix. (b) Zoom. (c) Average latency evolution. (d) Average throughput evolution.

generations of multiprocessor systems have been carried out. The router has been optimized for *k-ary 2-cube* interconnection networks specifically designed for CC-NUMA machines. As this low-dimensional router has low area requirements, it could be easily integrated within the processor chip.

This new architecture is the result of gathering several functional and technological optimizations together to obtain a competitive adaptive router design. The use of Bubble Flow Control as a deadlock avoidance mechanism provides fully adaptive routing using just one channel for each virtual network plus an adaptive shared channel. By selectively using output buffers to manage the most frequent switching cases and by implementing them as pipelined multiport memories, we have obtained an efficient architecture that highly reduces the effect of head-of-line blocking. The presence of output buffering has allowed us to use an optimized channel selection function that improves the load balance at almost no cost. Moreover, the most common accesses to these output buffers do not need arbitration.

Our HPAR has been evaluated and compared with other alternatives, starting at their hardware costs. The module's delays obtained from a VLSI synthesis process have been incorporated into a detailed network simulator able to deal with standard synthetic traffic patterns. The experiments for an 8×8 torus showed that the use of adaptive output buffering results in throughput gains ranging from 20 to 50 percent when compared to the simplest input buffer implementation. This improvement comes only with a minor increment in base latency. Our router also outperforms the alternative solution for avoiding HLB based on splitting the input buffer into multiple virtual lanes both in throughput (ranging from 14 to 40 percent) and in base latency. Besides, an execution-driven simulator able to faithfully emulate CC-NUMA multiprocessors has been employed to compare the impact of using different routers when running parallel applications. This testbed has allowed us to show how our proposal can reduce the execution time of several applications belonging to the SPLASH-2 suite. In this scenario, our router outperforms its most direct rival based on a higher number of virtual channels. These gains have been achieved even with lower area requirements.

From a technological point of view, HPAR occupies an area of approximately 10 mm^2 using a 0.25 μm design rules. If it were integrated within a state-of-the-art microprocessor using the same technology, it would increase the chip area by not more than 5 percent. All in

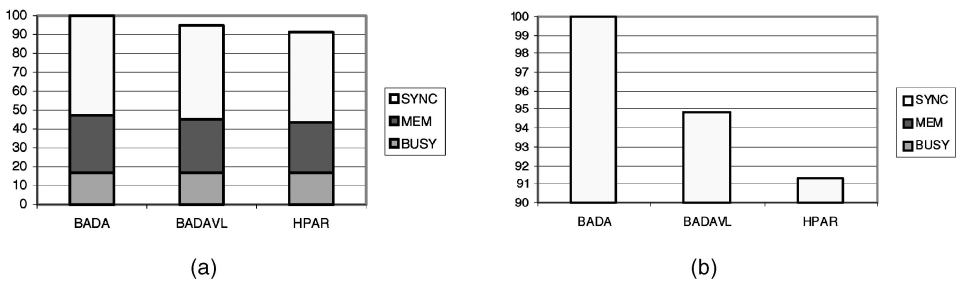


Fig. 17. (a) Normalized execution time for LU with 64 processors (8 x 8 Torus) for a 256 x 256 Matrix. (b) Zoom.

all, HPAR is an excellent candidate for integration into the processor chips that will configure the next generations of CC-NUMA multiprocessors.

ACKNOWLEDGMENTS

This work has been supported by the Spanish CICYT project TIC2001-0591-C02-01 and by the Spanish Ministry of Education under grant PR2002-0043.

REFERENCES

[1] A. Agarwal, "Limits on Interconnection Network Performance," *IEEE Trans. Parallel and Distributed Systems,* vol. 2, pp. 398-412, Oct. 1991.
[2] N.R. Adiga et al., "An Overview of the BlueGene/L Supercomputer," *Proc. Supercomputing 2002 Conf.*, Nov. 2002.
[3] P. Bannon, "Alpha 21364: A Scalable Single-Chip SMP," technical report, Compaq Computer Corp., http://www.digital.com/alphaoem/microprocessorforum.htm, 1998.
[4] L.A. Barroso, K. Gharachorloo, A. Nowatzyk, and B. Verghese, "Impact of Chip-Level Integration on Performance of OLTP Workloads," *Proc. High Performance Computer Architecture Conf., HPCA-6,* pp. 3-14, Jan. 2000.
[5] C. Carrion, R. Beivide, J.A. Gregorio, and F. Vallejo, "A Flow Control Mechanism to Prevent Message Deadlock in k-Ary n-Cube Networks," *Proc. Int'l Conf. High Performance Computing, HiPC '97,* pp. 50-58, Dec. 1997.
[6] W.J. Dally and C. Seitz, "The Torus Routing Chip," *Distributed Computing,* vol. 1, no. 3, pp. 187-196, Oct. 1986.
[7] W.J. Dally, "Performance Analysis of k-Ary n-Cube Interconnection Networks," *IEEE Trans. Computers,* vol. 39, no. 6, pp. 775-785, June 1990.
[8] W.J. Dally, "Virtual-Channel Flow Control," *IEEE Trans. Parallel and Distributed Systems,* vol. 3, no. 2, pp. 194-205, Mar. 1992.
[9] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks," *IEEE Trans. Parallel and Distributed Systems,* vol. 7, no. 8, pp. 841-854, Aug. 1996.
[10] M. Galles, "Spider, A High-Speed Network Interconnect," *IEEE Micro,* vol. 17, no. 1, pp. 34-39, Jan./Feb. 1997.
[11] M.J. Karol, M.G. Hluchyj, and S.P. Morgan, "Input versus Output Queuing on Space Division Packet Switch," *IEEE Trans. Comm.,* vol. 35, no. 12, pp. 1347-1356, Dec. 1987.
[12] M. Katevenis, P. Vatsolaki, and A. Efthymiou, "Pipelined Memory Shared Buffer for VLSI Switches," *Computer Comm. Rev.,* vol. 25, no. 4, pp. 39-48, Oct. 1995.
[13] R. Kessler, "The Alpha 21264 Microprocessor," *IEEE Micro,* vol. 19, no. 2, pp. 24-36, Mar./Apr. 1999.
[14] P. Kermani and L. Kleinrock, "Virtual Cut-Through: A New Computer Communication Switching Technique," *Computer Networks,* vol. 3, no. 4, pp. 267-286, Sept. 1979.
[15] S. Konstantinidou and L. Snyder, "The Chaos Router: A Practical Application of Randomization in Network Routing," *Proc. ACM Symp. Parallel Algorithms and Architectures,* pp. 21-30, July 1990.
[16] D. Lenoski and J. Laudon, "The SGI Origin: A ccNUMA Highly Scalable Server," *Proc. Symp. Computer Architecture,* pp. 241-251, June 1997.
[17] V.S. Pai, P. Ranganathan, and S.V. Adve, "RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors," *IEEE TCCA Newsletter,* vol. 35, no. 11, pp. 37-48, Oct. 1997.
[18] V. Puente, C. Izu, J.A. Gregorio, R. Beivide, and F. Vallejo, "The Adaptive Bubble Router," *J. Parallel and Distributed Computing,* vol. 61, no. 9, pp. 1180-1208, Sept. 2001.
[19] V. Puente, J.A. Gregorio, and R. Beivide, "SICOSYS: An Integrated Framework for Studying Interconnection Network in Multiprocessor Systems," *Proc. IEEE 10th Euromicro Workshop Parallel and Distributed Processing,* pp. 360-368, Jan. 2002.
[20] S.L. Scott and G. Thorson, "The Cray T3E Networks: Adaptive Routing in a High Performance 3D Torus," *Proc. Hot Interconnects IV,* Aug. 1996.
[21] D. Shah, P. Giaccone, and B. Prabhakar, "An Efficient Randomized Algorithm for Input-Queued Switch Scheduling," *Proc. XI Hot Interconnects Conf.,* Aug. 2001.
[22] R. Sivaram, C.B. Stunkel, and D.K. Panda, "HIPQS: A High-Performance Switch Architecture Using Input Queuing," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 3, pp. 275-289, Mar. 2002.
[23] C.B. Stunkel, D.G. Shea, and B. Abali, "The SP-2 High Performance Switch," *IBM System J.,* vol. 34, no. 2, pp. 185-204, May 1995.
[24] C.B. Stunkel, D.G. Shea, B. Abali, M.M. Denneau, P.H. Hochschild, D.J. Joseph, B.J. Nathanson, M. Tsao, and P.R. Varker, "Architecture and Implementation of Vulcan," *Proc. Int'l Parallel Processing Symp.,* pp. 268-274, Apr. 1994.
[25] Synopsys Online Documentation, v1999.10, 1999.
[26] Y. Tamir and G.L. Frazier, "Dynamically-Allocated Multiqueue Buffers for VLSI Communication Switches," *IEEE Trans. Computers,* vol. 41, no. 2, pp. 725-737, June 1992.
[27] A.S. Vaidya, A. Sivasubramaniam, and C.R. Das, "LAPSES: A Recipe for High-Performance Adaptive Router Design," *Proc. Int'l Symp. High-Performance Computer Architecture,* pp. 236-243, Jan. 1999.
[28] Y. Yeh, M. Hluchyj, and A. Acampora, "The Knockout Switch: A Simple, Modular Architecture for High-Performance Packet Switching," *IEEE J. Selected Areas in Comm.,* vol. 5, no. 8, pp. 1274-1287, Oct. 1987.
[29] S. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations," *Proc. Int'l Symp. Computer Architecture,* pp. 24-36, June 1995.

Valentín Puente received the MS and PhD degrees in physics from University of Cantabria, Spain, in 1995 and 2000, respectively. He is currently a lecturer at the University of Cantabria. His research interests are in VLSI design and interconnection network, with emphasis in performance optimization and fault-tolerant mechanism.

José-Ángel Gregorio received the MS and PhD degrees in physics (electronics) from the University of Cantabria, Spain, in 1978 and 1983, respectively. He is currently a professor of computer architecture in the Department of Electronics and Computers at the same university. His research interests include parallel and distributed computers, interconnection networks, and performance evaluation of computers and communication systems. He is a member of the IEEE Computer Society.

Ramón Beivide received the BSc degree in computer science from the Universidad Autónoma de Barcelona in 1981 and the PhD degree in computer engineering from the Universidad Politécnica de Catalunya (UPC) in 1985. He has been an assistant professor at the Universidad Politécnica de Catalunya and at the Universidad del País Vasco, both in Spain. In 1991, he joined the School of Telecommunication Engineering at the Universidad de Cantabria, Spain, where he is currently a professor. His research interests include parallel computers, interconnection systems, performance evaluation, and graph theory. Dr. Beivide has published more than 100 full-reviewed technical papers and he has served as referee, editor, and program chair of different international magazines and conferences. He is a member of the IEEE Computer Society.

Cruz Izu's biography and photograph are not available.