University of Castilla-La Mancha

A publication of the

Department of Computer Science

A Low-cost Router Design for k-ary n-cube Networks

by

M. C. Carrión, G. Díaz and B. Caminero

Technical Report **#DIAB-02-02-30** October 2002

DEPARTAMENTO DE INFORMÁTICA
ESCUELA POLITÉCNICA SUPERIOR
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

A Low-cost Router Design for k-ary n-cube Networks

M. C. Carrión, G. Díaz and B. Caminero

Dept. of Computer Science, University of Castilla-La Mancha, 02071-Albacete, SPAIN
carmen@info-ab.uclm.es

Abstract. A challenge to improve performance in parallel computers comes by reducing the delays a message experiments when traveling across the router nodes of the interconnection network. Also, routers must be carefully designed to avoid anomalies such as deadlock and starvation. When trying to achieve these goals, hardware complexity must be taken into account too. In this work we present the design of the *ghost-packet* router for k-ary 2-cube networks. Two version are described, *Stop-ghost* and the *Give-way ghost*. These routers are aimed at providing good performance, as well as deadlock and starvation freedom, while still making a simple implementation feasible.

Index Terms - Parallel computers, k-ary n-cube networks, routing algorithm, flow control, VLSI router design

1 Introduction

Parallel computers have been traditionally categorized into non-shared distributed memory computers, based on message passing, or those based on shared memory [15] [20]. Nowadays, shared memory machines are often more and more distributed, and use some form of message passing, a crucial component regarding performance. Thus, routers have become a crucial building block in interconnection networks and they have to use efficiently all the resources in order to provide powerful communication services at all load conditions.

Many recent experimental and commercial parallel computers [7] [20] use direct networks for low latency, high bandwidth interprocessor communication. The typical direct networks are k-ary n-cube structures [6], which are cubes with dimension n and k nodes in each dimension. Rings, meshes and tori are included in this class of networks. The low dimension toroidal networks have good graph-distance parameters and a reasonably large bisection width which implies low latency and high maximum sustained throughput. For this reason, we have focused our attention on this kind of networks.

Finding a suitable routing algorithm is quite difficult due to the fact that there are a myriad of design factors which can impact the final performance [16]. Even though, the most extended tendencies suggest the use of an adaptive routing scheme, virtual channels and wormhole switching.

A survey of wormhole routing algorithms can be found in [17]. Other routing algorithms are specifically designed for cut-through flow control [13]. They need larger buffer capacity but are less sensitive to propagation delay. This makes them quite attractive [1]. Moreover, when a header flit is blocked, all the data flits in the packet are stored in the same router releasing the physical links. So, our proposed algorithm uses a flow control scheme similar to cut-through.

One of the first routers that used virtual channels to avoid deadlocks is the Torus Routing Chip (TRC) [8]. Each physical channel provides two virtual channels that prevent deadlock by converting possible channel dependency cycles into spirals [9]. Other router designs using virtual channels are the CrayT3E router [20], the SGI SPIDER [12] and the Intel Cavallino [2]. These router designs also use adaptive routing techniques that alleviates the link contention problems.

Nevertheless, the use of both virtual channels and adaptive routing algorithms increases hardware costs [5]. Some works on this topic [22] show that simpler router designs imply lower execution times for all the applications tested. Thus, our work focuses on the study of a low cost router design that avoids the necessity of virtual channels.

The main contribution of this paper is the presentation of a deadlock and starvation free algorithm for n-dimensional toroidal networks, the *ghost-packet algorithm*. We explore the hardware complexity of the router and determine the network performance under different traffic patterns. Comparisons with other well known algorithms are included [3] [8]. The experiments show that the performance of parallel applications running in multiprocessor systems is highly sensitive to the hardware cost when talking in terms of speed.

The rest of this paper is organized as follows. In the first section,we present the *ghost-packet algorithm*. The basic router architecture is described in section 3. Then, by combining the VHDL description and the available technology, we have obtained a gate-level model of the router components, which has provided us with the router cost in terms of time delay. In section 4, a comparison with several routing algorithms is presented. Finally, in section 5, we draw the most important conclusions from the obtained results.

2 The Ghost-Packet Algorithm

In this section we present a simple deadlock and starvation free algorithm for k-ary n-cube networks. The VLSI design of the router is included in the next section.

2.1 Main Characteristics

The topology is a torus with bi-directional links and, although the algorithm works for n-dimensions, we assume a two dimensional network due to the fact that most current parallel processors are being built with low dimensional networks, and the obvious lay-out advantages of a two dimensional network for a single-substrate parallel computer routing component.

Some of the main characteristics of the *deterministic ghost-packet algorithm* are the following [4]:

1. Messages must be divided in data packets of a fixed length. Each data packet is composed of several phits [11].
2. A special kind of packet called the *ghost-packet* exists in each dimensional ring. These packets play an important role in order to guarantee that the network is deadlock free, as we will explain in the next subsection.
   We can define a *ghost-packet* as a network element that travels in the opposite direction to data packets. Its size is equal to a data packet, although it does not carry any information. *Ghost-packets* do not have any destination node. Rather, they remain in the network forever and are only visible from outside the dimensional ring.
3. Data packets are routed using a Dimensional Order Routing scheme (DOR) [8]. DOR has low hardware cost [5] and its minimal number of turns reduces the possible collision points for each data packet.
4. The header phit of the data packet carries the routing information to relay packets from their source to their destination node. Each time a data packet changes dimension, a header phit is split out.
5. Packet movement is governed by the ghost flow control function. This flow control scheme is quite similar to the virtual cut-through function because the header phit can only advance to a new buffer if there is enough storage capacity for the whole packet. Nevertheless, some other special restrictions must also be fulfilled. We will describe this flow control function in more detailin the next section.
6. No virtual channels are required to prevent the deadlock problem [9] [10] [18], which translates into low hardware cost.
7. The algorithm is starvation free because packets forwarded to the next router along the same dimension and turning packets have the same probability to access to the resources. The use of a fair arbitration policy, such that the advance of every packet is guaranteed within a finite period of time, is assumed.

2.2 The Ghost Flow Control

A flow control function, $F : Q * R * S \rightarrow True, False$, enables or disables the advance of a packet from the input queue, Q, to the input queue in the next router selected by the routing function, R, depending on the state, S, of the destination queue.

In the ghost flow control algorithm, a queue q has three possible states, $S(q) = Busy, Free, G\_busy$. If a buffer has capacity for one packet, the state of the queue is $Free$; in any other case the capacity will be either $Busy$ or G_Busy. The $Busy$ state is reached by a queue when all the resources are filled with data packets. In the G_Busy state, there are no free resources either, but one of the elements in the queue is the *ghost-packet*.

Two slightly different flow control proposals can be considered depending on the *ghost packet* behavior:

Stop-Ghost packets (S-Ghost) In this case, the *ghost-packet* moves backwards only when there is an exchange with a data packet advancing in the same dimension. If a data packet is requesting access to a *G_Busy* queue, the ghost will be moved backwards and the data packet will be forwarded to the next router along the same dimension. On the other hand, if the relative x-address is zero the data packet must be forwarded along the next y-dimension, but in this case no movement is allowed until the state of the destination queue becomes $Free$.

A formal description of the S-Ghost flow control algorithm for the data packets is shown in Equation 1.

$$F_{S-Ghost}(q_j, q_i, S(q_i)) = \begin{cases} True \;\; if \;\; S(q_j) = Free \text{ or} \\ \qquad\qquad S(q_j) = G_{busy} \; and \; dim(q_i) = dim(q_j) \\ False \; otherwise \end{cases} \tag{1}$$

Where q_j is the source queue, q_i the destination queue, and the function $dim(q)$ returns the dimension to which queue q belongs.

Give-way Ghost packets (GW-Ghost) In this case, the *ghost-packet* can move backwards in an exchange as in the previous case, but now there is another situation in which the *ghost-packet* can move backwards. Give-way ghost packet will try to move backwards whenever there is an external request to access to its buffer and the state of the buffer in the previous node in the same dimension is $Free$. Equation 2 shows a formal description of the GW-Ghost flow control for the data packets.

$$F_{GW-Ghost}(q_i, q_j, S(q_j), S(q_{j-1}) = \begin{cases} True \;\; if \;\; S(q_j) = Free \text{ or} \\ \qquad\qquad S(q_j) = G\_busy \;\; and \;\; S(q_{j-1}) = Free \\ False \; otherwise \end{cases} \tag{2}$$

In the GW-Ghost algorithm, a turning data packet can be forwarded to the next router even though the state of the output queue is *G_busy* due to the mobility of the ghost-packet. Both ghost flow control schemes avoid the deadlock problems on torus networks under DOR routing by preserving enough free resources for each cyclic dependence.

3 Router Design

We have described the behavior of the internal router components for a bidirectional 2D torus network by using the hardware description language VHDL. In this section we describe the functional blocks that compose a *ghost-packet* router.

Although the communication between routers is asynchronous, internal router components are clocked synchronously. The goal of using asynchronous communication is to avoid clock skew problems. Each unidirectional link has a 16-bit datapath and 5-bits dedicated to control signaling. An unidirectional physical channel is detailed in Figure 1-A). The *Request* and *Ack* signals control the transmission of the phits. The *Request* signal inform next router of the arrival of a new data phit. The *Ack signal* works at the packet level so that communication is less sensitive to wire delay. The *tail* signal is used to inform the neighbor router about the end of a packet transmission. And finally, the *Here_G* and the *Free_G* signals control the movements of the *ghost-packet* along the same dimension.

3.1 Basic Router Blocks

The organization of the *ghost-packet* router is shown in figure 1-B). To have a global idea of the hardware structure of the router, let us briefly describe the functional modules of the circuit.

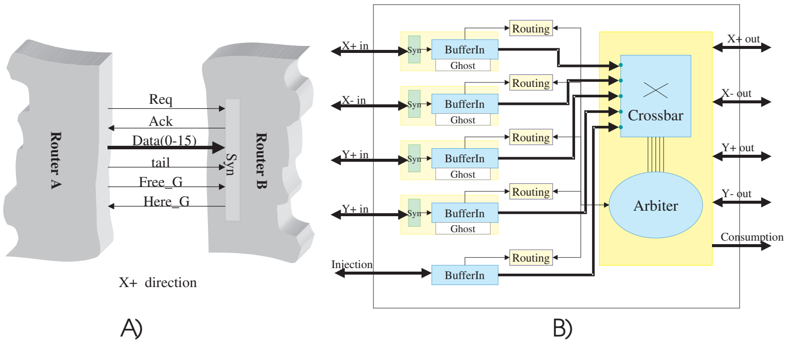

Fig. 1. Router structure for a k-ary 2-cube network

1. **The Syn block:** Its purpose is to synchronize the arriving flits with the internal clock. The protocol between routers has been chosen to avoid the necessity of an acknowledgement signal for each flit sent. Thus, the input port needs storage capacity for some flits in order to guarantee that none of the possible arriving flits, even those in flight on the channel's wires, are missed.
2. **The BufferIn block:** It uses a FIFO policy and has capacity for $m*L$ phits, where m is any integer and L is the length of the packets. In our experiments, $L = 10$ phits. This module has been designed using hybrid SRAM/register buffers. Hence, big storage capacity is easily achieved without increasing the latency of the module.
3. **The Ghost block:** It controls the external signals that implement the ghost flow control.
4. **The Routing block:** It takes the head flit and decides the profitable output port, according to the dimensional order routing algorithm. The relative address is also updated by this block, either decreasing by one the value in

the head flit or splitting out the whole first head flit. The complexity of this module is determined by the number of outputs and the length of the word to be compared and decreased.

5. **An Arbiter block:** It decides in a round-robin fashion about the assignment of an output port. The delay of this component depends on the number of input ports, $2n+1$ in our case.
6. **The Crossbar block:** It handles the switching functionality of the router, with the aid of the arbiter.

3.2 Synthesis Level: Area and Timing Results

In order to get the clock period, we have carried out the design of the router components using the hardware description language VHDL [21]. The cost in terms of delay of the different modules has been obtained by using the CAD tool *Synopsys v3.4a* [14]. Both the VHDL description code and the synthesis process of the modules have been done with the main goal of minimizing the clock period.

Table 1 shows the temporal delay of each block, when mapping the VHDL designs into 0.35μ technology from MIETEC foundry under typical working conditions.

MODULE	TIME	MODULE	TIME
Syn	1.54	Routing	1.88
S-Ghost	1.73	GW-Ghost	1.90
Arbiter & Crossbar	2.14	BufferIn (8L)	2.90
BufferIn (4L)	2.38	BufferIn (2L)	2.09

Table 1. Time expressed in nanoseconds

4 Experimental Results

In this section we evaluate and compare the *ghost-packet* algorithm performance with other proposals. It should be noted that the results presented later include also the hardware cost of the router.

4.1 Methods

We use a register transfer level simulator which has been written in C++ to measure the network performance. The low-level characteristics of the routers have been included into the simulator. In fact, as routers are synchronous the simulator is cycle-driven. The asynchronous communication among routers has been emulated by considering an average delay of one cycle.

A run consists of a single combination of network, load, communication pattern, and packet size. A run terminates either after a minimum number of packets have traveled through the network and the latency converges or when the network goes into saturation (more than 50,000 cycles).

We use different communication patterns: random and synthetic loads. In the first case, all destinations are equally probable. Synthetic loads try to summarize the characteristics of some patterns founds in scientific computations. Among these destination patterns we can mentioned the perfect-shuffle, transpose and bit-reversal distribution. Regarding the traffic generation rate, we have used a uniform and random distribution over time for all the run simulations.

In the results presented below, we show the performance of a 8×8 torus network. The packet size was chosen to represent the transfer of a small cache line. One of the metrics used to measure network performance is *maximum throughput*, which is defined as the maximum load which does not cause saturation. The other figure of merit is the average packet latency at different loads. The load is presented in terms of the expected number of phits injected per node per cycle.

4.2 Comparing to other Algorithm proposals

To validate the *ghost-packet algorithm* performance, we have compared it to the Dally&Seitz [8] and Bubble [3] algorithms.

The Dally&Seitz [8] algorithm is used by the Torus Routing Chip (TRC) which uses DOR and the wormhole flow control technique. The TRC uses two virtual channels to avoid deadlock problems.

The Bubble algorithm [3] uses DOR and a special flow control algorithm based on injection restriction. This flow control scheme is similar to virtual cut-through. It only differs in that packet injection into a ring is only allowed if there is free space for at least two packets in the set of queues belonging to the dimension requested by the incoming packet. The deterministic bubble algorithm is deadlock free but it can suffer from starvation.

The hardware components of a deterministic bubble router are similar to the ones of the *ghost-packet* router, but the TRC requires an extra module, namely the virtual channel controller for multiplexing between the virtual channels of a physical channel. Therefore, a packet going from an input port to an output port without contention will need 4 cycles and 5 cycles in the Bubble and TRC router, respectively. The Crossbar&Arbiter module determines the clock period of the TRC router, whereas for the deterministic Bubble router, the buffer module is in the critical path so clock period depends on input buffer capacity. More information about the structure of the TRC and the Bubble router design can be found in [19].

Figure 2 shows the results obtained for random and transpose pattern traffic. The input buffer size is fixed to $2L$.

These experiments show that both *ghost-packet* algorithms (S-Ghost and GW-Ghost) achieve better performance results than TRC for all the loads. The *ghost-packet* algorithm reduces the network latency at zero load by 30% , while the increase in the maximum sustained throughput is 20%. Regarding the bubble

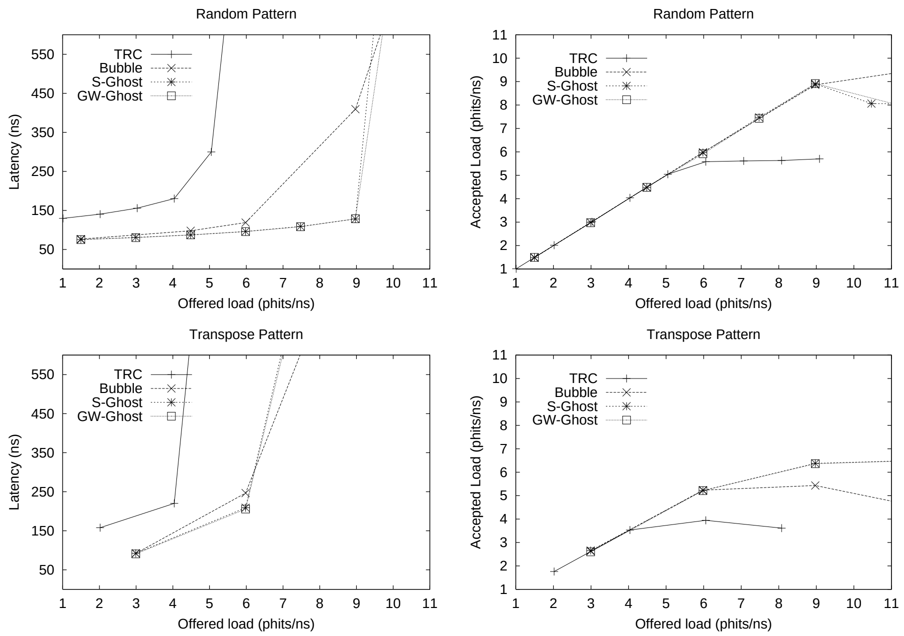

Fig. 2. Network performance under different Random and Transpose traffic patterns.

algorithm, we can observe that network latency at zero load is same to the one obtained by the *ghost-packet* one. The maximum sustained throughput is quite similar to that of the proposed algorithms. Moreover, if the input buffer capacity is increased, the bubble algorithm performs slightly better at high loads, see Figure 3. We can appreciate how the accepted load of the *ghost-packet* decreases at high loads. This problem can be alleviated by not allowing the injection of so many packets into the network after the saturation point. In fact, it could be completely avoided by configuring the network in such a way that more than one *ghost-packet* is confined per ring. Several simulation runs have confirmed this issue. For instance, the latency and the accepted load of the S-Ghost algorithm are shown in Figure 4 for both the minimum and the maximum number of allowed ghosts per ring. In this case, the storage capacity of the input buffers is $8L$, so the minimum and maximum values are 1 and 7, respectively.

5 Conclusions

A deadlock and starvation free algorithm for k-ary n-cube networks with DOR routing function has been presented. The main idea of the proposed algorithm is the *ghost-packet* mobility. A *ghost-packet* is one that does not contain any information, but avoids to exhaust the resources of a cyclic dependency. The VHDL design of the router has also been carried out. We have evaluated the design

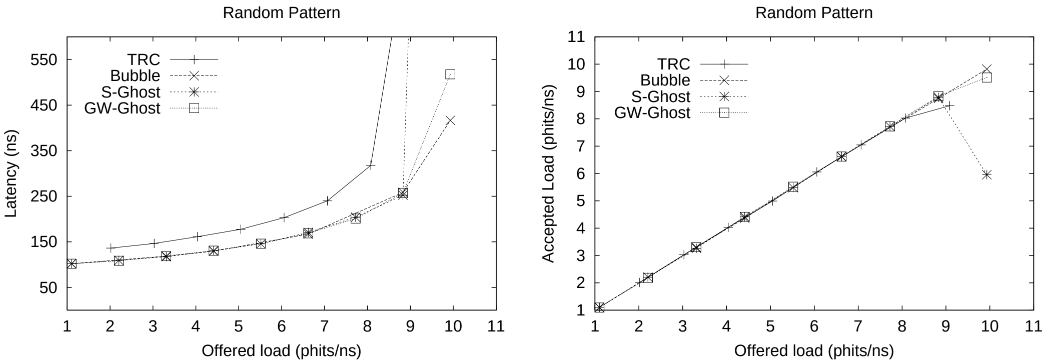

Fig. 3. Input buffers with capacity for $8 \times L$. Random traffic.

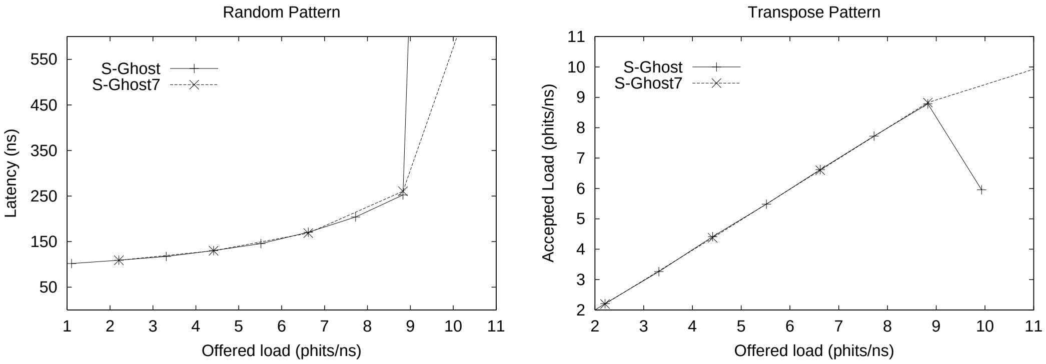

Fig. 4. Average Packet latency. Random traffic. BufferIn=$8 \times L$.

by combining the technology results with simulation tools. Several comparisons have been done on a 2D torus topology. The *ghost-packet* algorithm presents good performance due to the lower hardware cost of the router. Improvements of around 25% are achieved with respect to the TRC router, which uses virtual channels for deadlock avoidance. The results obtained are quite similar to those of the deterministic bubble algorithm, but, as opposite to the bubble algorithm, the *ghost-packet* algorithm does not suffer from starvation.

References

1. A. Agarwal, *Limits on interconnection network performance*, IEEE Transactions on Parallel and Distributed Systems **2** (1991), no. 4, 398–412.
2. J. Carbonaro and F. Verhoorn, *Cavallino: The teraflop router and nic*, Hot Interconnects Symposium IV (1996).

3. C. Carrión, R. Beivide, J.A. Gregorio, and F. Vallejo, *A flow control mechanism to avoid message deadlock in k-ary n-cube networks*, International Conference on High Performance Computing (1997), 322–329.
4. C. Carrión, C. Izu, J.A. Gregorio, F. Vallejo, and R. Beivide, *Ghost packets: A deadlock-free solution for k-ary n-cube networks*, 6th Euromicro Workshop on Parallel and Distributed Processing (PDCS-97) (1998), 22–32.
5. A. Chien, *A cost and speed model for k-ary n-cube wormhole routers*, Proceedings of Hot Interconnections'93, August 1993.
6. W. J. Dally, *Performance analysis of k-ary n-cube interconnection networks*, IEEE Transactions on Computers **39** (1990), no. 6, 775–785.
7. W.J. Dally and L. R. Dennison, *Architecture and implementation of the reliable router*, Proc. of Hot Interconnets II,Stanford University, Palo Alto (1994).
8. W.J. Dally and C.L. Seitz, *The Torus Routing Chip*, Distributed Computing **1** (1986), no. 3, 187–196.
9. ———, *Deadlock-free message routing in multiprocessor interconnection networks*, IEEE Transactions on Computers **C-36** (1987), no. 5, 547–553.
10. J. Duato, *Necessary and sufficient condition for deadlock-free routing in cut-through and store-and-forward networks*, IEEE Transactions on Parallel and Distributed Systems **7** (1996), no. 8, 841–854.
11. J. Duato, S. Yalamanchill, and L. Ni, *Interconnection networks*, IEEE Computer Society, Washington, 1997.
12. M. Galles, *Scalable pipelined interconnect for distributed endpoint routing: The SGI SPIDER Chip*, Proceedings of Hot Interconnects Symposium IV (1996), 141–146.
13. P. Kermani and L. Kleinrock, *Virtual cut-through: A new computer communication switching technique*, Computer Networks **3** (1979), 267–286.
14. P. Kurup and T. Abbasi, *Logic synthesis using synopsys*, Kluwer Academic Publishers, Norwell, 1995.
15. J. Laudon and D. Lenoski, *The SGI Origin: A ccNUMA highly scalable server*, Proceedings of the 24rd Annual International Symposium on Computer Architecture, ACM SIGARCH and IEEE Computer Society TCCA, June 1997, pp. 241–251.
16. K. J. Liszka, J. K. Antonio, and H. J. Siegel, *Problems with comparing interconnection networks. is an Alligator better than an Armadillo?*, IEEE Concurrency, Parallel, Distributed and Mobile Computing **5** (1997), 18–28.
17. L. M. Ni and P. K. McKinley, *A survey of wormhole routing techniques in direct networks*, IEEE Computer Magazine **26** (1993), no. 2, 62–76.
18. T. M. Pinkston and S. Warnakulasuriya, *On deadlocks in interconnection networks*, Proceedings of the 24rd Annual International Symposium on Computer Architecture, ACM SIGARCH and IEEE Computer Society TCCA, June 1997, pp. 38–49.
19. V. Puente and J.A. Gregorio et al, *Adaptive bubble router: a design to balance latency and throughput in networks for parallel computers*, IEEE International Conference on Parallel Computing (1999).
20. S. L. Scott and G. M. Thorson, *The Cray T3E Network: Adaptative routing in high performance 3D Torus*, Proceedings of Hot Interconnections'93, August 1996, pp. 147–155.
21. IEEE Standard VHDL, *IEEE standard VHDL language reference manual*, std 1076 ed., IEEE, 1993.
22. A. S. Vaidya, A. Sivasubramaniam, and C. R. Das, *Performance benefits of virtual channels and adaptative routing: An application-driven study*, International Conference on Supercomputing, July 1997, pp. 140–147.