FACULDADE DE INFORMÁTICA
PUCRS - Brazil
http://www.inf.pucrs.br

Evaluation of Routing Algorithms on Mesh Based NoCs

Aline Vieira de Mello, Luciano Copello Ost,
Fernando Gehm Moraes, Ney Laert Vilar Calazans

TECHNICAL REPORT SERIES

Number 040

May, 2004

Contact:

calazans@inf.pucrs.br

http://www.inf.pucrs.br/~calazans

A. V. de Mello works at the PUCRS/Brazil as a Research Assistant since July, 2003. Ms. Mello receives a grant from the RICHA project, financed by the CNPq. He holds a B.Sc. Degree in Computer Science from PUCRS in Brazil. His main research interests are digital systems design and intra-chip networks.

L. C. Ost holds an M.Sc. degree obtained at the PPGCC/FACIN/PUCRS in 2004. He is Research Assistant at PUCRS/Brazil since April, 2004. His main research interests are digital systems design, fast prototyping, intra-chip networks, and standard interfaces.

F. G. Moraes works at the PUCRS/Brazil since 1996. He is a professor since August 2003. His main research topics are digital systems design and fast prototyping, digital systems physical synthesis CAD, telecommunication applications, hardware-software codesign. Dr. Moraes is a member of the Hardware Design Support Group (GAPH) at the PUCRS.

N. L. V. Calazans works at the PUCRS/Brazil since 1986. He is a professor since 1999. His main research topics are digital systems design, fast prototyping, and applications, reconfigurable systems, and embedded systems. Dr. Calazans is the head of the Hardware Design Support Group (GAPH) at PUCRS.

Evaluation of Routing Algorithms on Mesh Based NoCs

Aline Mello, Luciano C. Ost, Fernando G. Moraes, Ney L. Calazans

Pontifícia Universidade Católica do Rio Grande do Sul (FACIN-PUCRS),
Av. Ipiranga, 6681 - P 30/Bl 4 - 90619-900 - Porto Alegre – RS– BRASIL
{alinev, ost, moraes, calazans}@inf.pucrs.br

Abstract. The increasing complexity of integrated circuits drives the research of new on-chip interconnection architectures. Networks-on-chip (NoCs) are a candidate architecture to be used in future systems, due to its increased performance, reusability and scalability. A NoC is a set of interconnected switches, with IP cores connected to these switches. Four main components compose a switch: a *router*, to define a path between input and output switch ports; an *arbiter*, to grant access to a given port when multiple input requests arrive in parallel; *buffers*, to store intermediate data, and a *flow control module* to regulate the data transfer to the next switch. The goal of this work is to compare the performance of four routing algorithms for mesh based packet switching NoCs. Differently from the literature for generic networks, it is shown that deterministic algorithms can be superior to adaptive ones in NoCs.

1 Introduction

System on a chip (SoC) is the design methodology currently used by VLSI designers, based on extensive IP core reuse. Cores do not make up SoCs alone, they must include an interconnection architecture and interfaces to peripheral devices [1].

Usually, the *interconnection architecture* is based on dedicated wires or shared busses. *Dedicated wires* are effective only for systems with a small number of cores, since the number of wires in the system increases dramatically as the number of cores grows. Therefore, dedicated wires have poor reusability and flexibility. A *shared bus* is a set of wires common to multiple cores. This approach is more scalable and reusable, when compared to dedicated wires. However, busses allow only one communication transaction at a time. Thus, all cores share the same communication bandwidth in the system and scalability is limited to a few dozens IP cores [2]. Using separate busses interconnected by bridges or hierarchical bus architectures may reduce some of these constraints, since different busses may account for different bandwidth needs, protocols and also increase communication parallelism. Nonetheless, scalability remains a problem for hierarchical bus architectures.

A *network on chip* (NoC) appears as a probably better solution to implement future on-chip *interconnection architectures* [2]-[7]. In the most commonly found organization, a NoC is a set of interconnected switches, with IP cores connected to these switches. NoCs present better performance, bandwidth, and scalability than shared busses [3].

Switches are responsible for: (*i*) receiving incoming packets; (*ii*) storing packets; (*iii*) routing these packets to a given output port; (*iv*) sending packets to others switches. To accomplish these functions, four main components compose a switch: a *router*, to define a path between input and output switch ports (function *i*); *buffers* to store intermediate data (function *ii*); an *arbiter* to grant access to a given port when multiple input requests arrive in parallel (function *iii*); and a *flow control* module to regulate the data transfer to the next switch (function *iv*).

Packet switching is by far the most employed switching mechanism in NoCs, although circuit switching NoCs have already been proposed [7]. Packet switching requires the use of a *switching mode*, which defines how packets move through the switches [8]. The *wormhole* switching mode avoids the need for large buffer spaces, since a packet is transmitted between switches in smaller units, called *flits*. Only the header flit has routing information. Thus, the rest of the flits that compose a packet must follow the same path reserved for the header.

The objective of this paper is to evaluate different *routing algorithms* for NoCs employing wormhole packet switching, in mesh topologies. A comprehensive revision of related works highlighted the lack of analysis concerning routing algorithms, in the NoC context [6]. In this revision it is stated that some NoCs were already prototyped in FPGAs. Among these, lies the HERMES NoC used in the present work. In the long term, the Hermes NoC is intended to use in the context of coarse-grain reconfigurable architectures, with the NoC playing the part of a fixed interconnection fabric where reconfigurable complex modules are plugged on demand.

The rest of this paper is organized as follows. Section 2 presents basic concepts of routing algorithms. Section 3 describes different routing algorithms, emphasizing adaptive ones. The HERMES NoC [6], used to validate the routing algorithms, is briefly presented in Section 4. Section 5 compares the algorithms, considering relative performance under different load conditions and packet sizes. Finally, Section 6 presents some conclusions and directions for future work.

2 Routing Algorithm Basics

Routing algorithms define the path taken by a packet between source and target switches. They must prevent deadlock, livelock, and starvation [8][9] situations. *Deadlock* may be defined as a cyclic dependency among nodes requiring access to a set of resources, so that no forward progress can be made, no matter what sequence of events happens [6]. *Livelock* refers to packets circulating the network without ever making any progress towards their destination. *Starvation* happens when a packet in a buffer requests an output channel, being blocked because the output channel is always allocated to another packet.

Routing algorithms can be classified according to the three different criteria: (*i*) where the routing decisions are taken; (*ii*) how a path is defined, and (*iii*) the path length.

According to where *routing decisions* are taken, it is possible to classify the routing in *source* and *distributed* routing. In source routing, the whole path is decided at the source switch, while in distributed routing each switch receives a packet and defines the direction to send it. In source routing, the header of the packet has to carry all the routing information, increasing the packet size [9]. In distributed routing, the path can be chosen as a function of the network instantaneous traffic conditions. Distributed routing can also take into account faulty paths, resulting in fault tolerant algorithms.

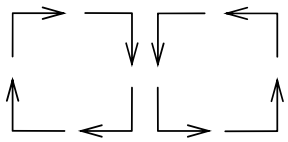
Depending how a path is defined, routing can be classified as *deterministic* or *adaptive*. In deterministic routing, the path is completely specified from the relative position of source and target addresses. In adaptive routing, the path is a function of the network instantaneous traffic [4]. Adaptive routing increases the number of possible paths usable by a packet to arrive to its destination. However, deadlock and livelock situations can happen in fully adaptive algorithms [8], which limit its usage.

Regarding the path length criterion, routing can be *minimal* or *nonminimal* [8][9]. Minimal routing algorithms guarantee shortest paths between source and target addresses. In *nonminimal* routing, the packet can follow any available path between source and target. Nonminimal routing offers great flexibility in terms of possible paths, but can lead to livelock situations and increase the latency to deliver the packet.

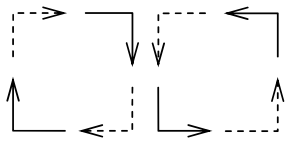
3 Routing Algorithms

Glass and Ni proposed wormhole routing algorithms for mesh-connected networks, which are deadlock and livelock free [10]. These were proposed in minimal and nonminimal partially adaptive versions.

When passing between switches in a 2D mesh, a packet can follow four directions: East, West, North, and South. Eight distinct turns are possible in the path followed by a packet, as shown in Fig. 1(a). Algorithms with no restrictions are named *fully adaptive*, otherwise they are named *partially adaptive*. Fully adaptive routing algorithms are subject to deadlock conditions. As demonstrated in [10], if at least two turns are forbidden (dotted lines in Fig. 1(b)) it is possible to implement deadlock free algorithms. This a sufficient condition for achieving freedom of deadlock.



(a) All possible turns in a 2D mesh.



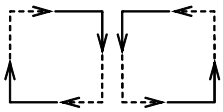
(b) A routing algorithm that limits the possible turns, can result in a deadlock free algorithm.

Fig. 1 –The *Turn Model* for adaptive routing [10].

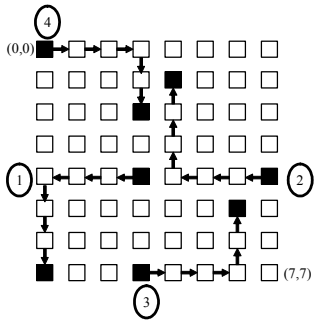
This Section presents four routing algorithms, one deterministic (XY) and three partially adaptive - *West first*, *North-last* and *Negative-first.* Only *minimal* [10] algorithms are considered in this work, to avoid increased latency to deliver packets and livelock situations.

Source and target coordinates are identified in the following discussion by the use of the notations (X_S, Y_S) and (X_T, Y_T), respectively.

The **XY** algorithm is deterministic. Flits are first routed in the X direction, until reaching the Y_T coordinate, and afterwards in the Y direction, as shown in Fig. 2(b). If some network hop is in use by another packet, the flit remains blocked in the switch until the path is released. As illustrated in Fig. 2(a) turns where the packet comes from the Y direction are forbidden (dotted lines).



(a) Continuous lines represent allowed turns. Only 4 turns are allowed.



(b) XY routing algorithm.

Fig. 2 – XY routing algorithm.

In the **West-First** algorithm if $X_T \leq X_S$, packets are routed deterministically, as in the XY algorithm, (Fig. 3, paths 1 and 2). If $X_T > X_S$ packets can be routed adaptively in East, North or South directions (Fig. 3, paths 3 and 4). The total time to deliver an individual packet can be reduced using adaptive algorithms, since the packet can, in some situations, make turns to escape from blocking conditions.

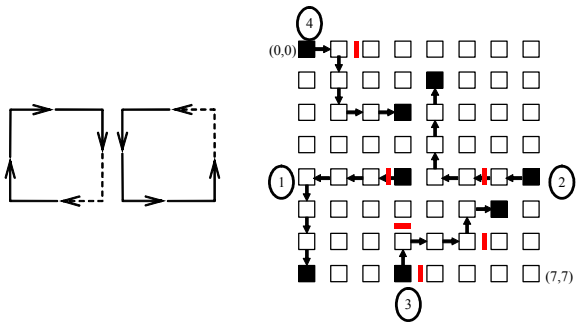


Fig. 3 - West-first routing algorithm. The prohibited turns are the two to the West.

In the **North-Last** algorithm if $Y_T \leq Y_S$ packets are routed deterministically (Fig. 4, paths 2 and 3). If $Y_T > Y_S$ packets can be routed adaptively in West, East, or South directions (Fig. 4, paths 1 and 4).

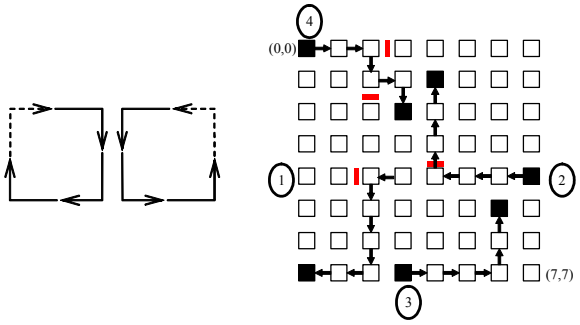


Fig. 4 - North-Last routing algorithm. The prohibited turns are the two when traveling North.

In the **Negative-First** algorithm packets are routed first in negative directions, i.e., to the North or to the West directions[1]. If ($X_T \leq X_S$ and $Y_T \geq Y_S$) or ($X_T \geq X_S$ and $Y_T \leq Y_S$) packets are deterministically routed, as illustrated in Fig. 5, path 1 (source address (3,4) and target address (0,7)) and path 3 (source address (3,7) and target address (6,5)). All other conditions allow some form of adaptive routing, as illustrated in paths 4 and 2.

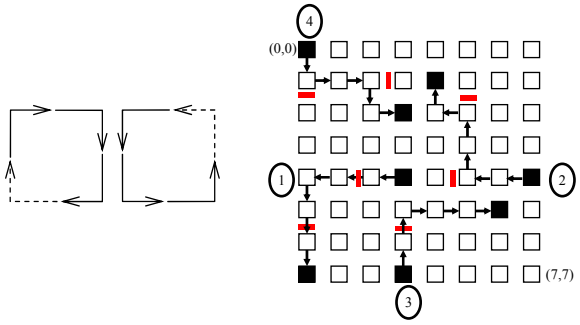


Fig. 5 - Negative-First routing algorithm. The prohibited turns are the two from a positive direction to a negative direction.

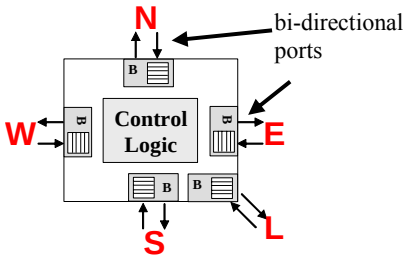
It is important to stress that, as stated before, the observations about the algorithms in this Section apply exclusively to minimal routing algorithms.

[1] The notation in this paper is slightly different from Ni´s [10], because of the different sense for the growing values on the Y axis.

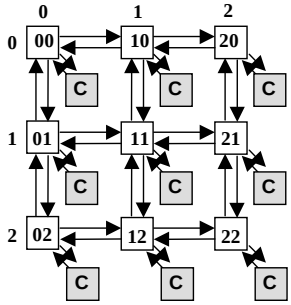
4 HERMES Network on Chip

HERMES is an infrastructure used to implement packet-switching NoCs for different topologies, flit sizes, buffer depths and routing algorithms [5][6]. It supports the implementation of the three lowers OSI-RM layers [11]: (*i*) physical - corresponding to the communication interface between switches, (*ii*) data link– referring to the handshake protocol built on top of the physical layer to deal with flow control and correctly sending and receiving data, and (*iii*) network - which implements the packet switching technique.

The main component of this infrastructure is the HERMES switch (Fig. 6a). This switch has a Control Logic module centralizing local arbitration and routing decisions, and five bi-directional ports: East, West, North, South, and Local. Each port has an input buffer for temporary storage of information. The Local port establishes a communication between the switch and its local IP core. Fig. 6b illustrates the structure of a 3*x*3 HERMES mesh network.



(a) HERMES switch architecture. B indicates input buffers.



(b) 3x3 Mesh topology. C marks IP cores, switch addresses indicate the XY position in network.

Fig. 6 – Hermes NoC.

Internally, two modules compose the Control Logic: *routing* and *arbitration*. The *routing* module implements one of the four previously presented algorithms. As a switch can simultaneously be requested to establish up to five connections, arbitration logic is used to grant access to an output port when one or more input ports simultaneously require a connection. A dynamic arbitration scheme is used.

A *2x2* HERMES NoC was successfully prototyped in a million-gate FPGA. It is estimated that a *5x5* HERMES NoC can be implemented in a 4-million gate device with a small 16-bit processor connected to the Local port of each switch. The reader should refer to [5] and [6] for a more extensive description of the HERMES infrastructure and NoCs built with it.

5 Routing Algorithms Comparison

5.1 Experimental Setup

HERMES NoCs are described in VHDL, while traffic generation and analysis are written in C language. Co-simulation experiments employ the ModelSim simulator and the FLI library [12], which allows VHDL and C to communicate.

A set of parameters was fixed for use in the simulations. Most simulations involve a 5 x 5 network mesh topology. All simulations assume a traffic generator is connected to each switch, each sending 40 packets. This represents simulations with data about a total of 1,000 packets each. All simulated NoCs use 16-bits flit size and 8-positions switch input buffers.

To evaluate the routing algorithms (XY, *West-first*, *North-last*, *Negative-first*) the following parameters vary: (*i*) packet size: 10, 100, 1000 and 10000 flits; (*ii*) generated traffic: given a routing algorithm, a packet size, and a traffic load, 3 distinct traffics are generated by

randomly varying the source-destination pairs; (*iii*) traffic load: 30%, 50%, 70%, and 100%.

The *load* offered by a given simulated traffic is defined as the percentage of the channel bandwidth used by each communication initiator [7]. Fig. 7 illustrates two different traffic loads, 100% and 50%. A 100% load arrives when all cores are continually sending data, without interruption between successive packets. In real situations, the system load is much smaller. This can be compared to data reported in [7], where the PI-bus architecture is shown to work well with load values below 4% and the SPIN NoC with load values below 28%. The Hermes NoC, due to its mesh topology, does support heavier traffic loads.

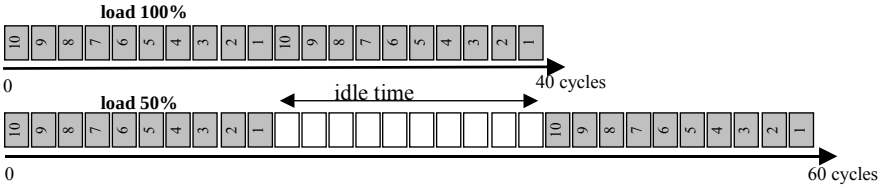


Fig. 7 –Traffic load interpretation. In HERMES NoCs, a flit takes 2 clocks cycles to be transmitted. Thus, each 10-flit packet takes a minimum of 20 clock cycles to be sent.

Traffic used for simulation has an average distance between source and target switches corresponding to half of the maximum distance in the NoC. In our experiment, for a 5x5 network, the longest distance is 8 hops. This guarantees equilibrium between short and long paths, which is important to adequately evaluating routing.

The above described parameter variations led to a total execution of 192 distinct simulations.

Co-simulation CPU time, in a Sun Blade 2000 900 MHz/1GB RAM, is in average 5, 28, 294 and 2587 seconds for packets sizes equal to 10, 100, 1000 and 10000 flits respectively.

5.2 Results and Discussion

Three issues are discussed in this Section: relative routing algorithm performance, the influence of packet size and the influence of the network load. Fig. 8 illustrates the performance, in clock cycles, for each routing algorithm, considering different packet sizes and traffic loads.

Concerning the relative performance of the algorithms, results indicate that, in terms of total clock cycles to deliver all packets, deterministic XY routing is faster than the other three partially adaptive algorithms. Partially adaptive algorithms can potentially speed up the time to deliver individual packets, but globally the results point out to poorer performance than the XY algorithm. Glass and Ni [10], suggest that reducing the number of turns that a message takes may reduce blocking and hence improve performance. This can be justified because adaptive routing has a trend to concentrate the traffic in the center of the network, increasing in this way the number of blocked paths. The *North-last* algorithm presents a small advantage over the XY algorithm for 30% traffic and small packets (10 and 100 flits). This situation leads to a reduced number of blocked paths and the availability of idle time between packets. As the XY algorithm can not explore different paths, even when they are available, adaptive algorithms have a potential advantage in this case.

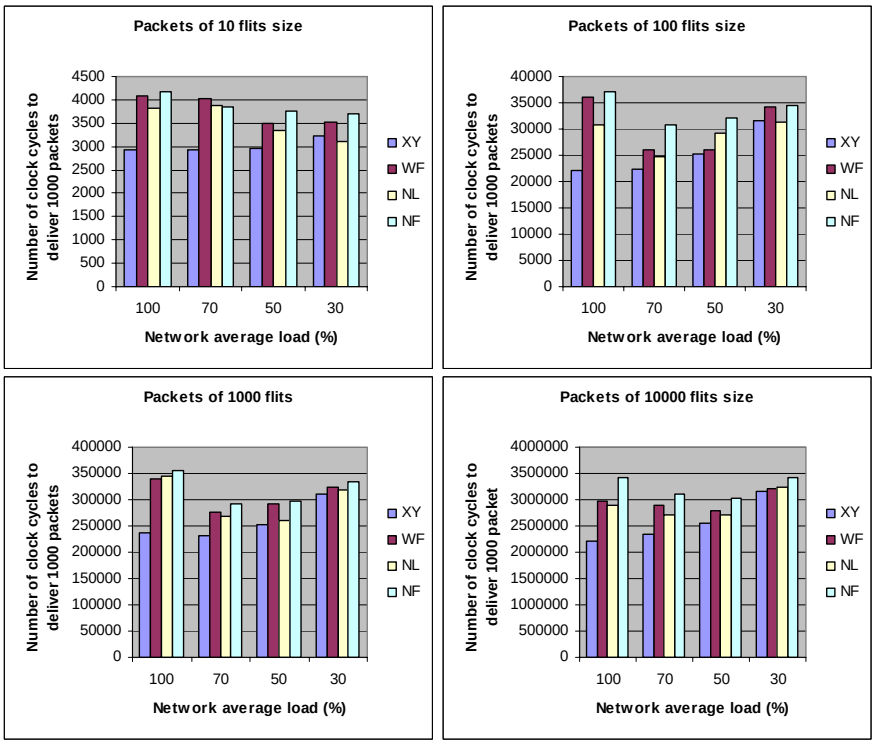


Fig. 8 - Comparison of routing algorithms for different packets size and network loads: (a) 10-flit packets, (b) 100-flit packets, (c) 1000-flit packets, (d) 10000-flit packets. Each bar in a graph represents the average value over three simulations.

These results were obtained for a relatively small NoC (*5x5*), which can mask the possible advantages of partially adaptive algorithms. To support these claims for larger NoCs, Table 1 compares the performance of the algorithms for a *10x10* NoC. Again, for larger NoCs the XY algorithm is faster than the other ones.

Table 1- Performance of routing algorithms for a *10x10* NoC, packet size equals to 1,000 flits, and traffic load equals to 70%.

Routing Algorithm	Number of clock cycles to deliver 1,000 packets
XY	129694
West First	195161
North Last	187148
Negative First	201797

Next, it is important to analyze the trade-off between packet size and the *average time* to deliver these packets. In Fig. 8, the average time to deliver 1 packet (XY algorithm, load 70%) for packets with 10, 100, 1000 and 10000 flits is 132, 460, 79352 and 1054707 clock cycles, respectively. Dividing this time by the packet size it is possible to obtain the *average time to deliver 1 flit*. This time corresponds to 13.2, 4.6, 79.4 and 105.47 clock cycles, respectively.

Small packets have the advantage to induce a small number of blocked paths, with a corresponding reduction in packet delivery times. Larger packets present the opposite behavior. However, small packets impose larger overheads for segmentation and reassembly in the IP core wrappers. Also, arbitration/routing is executed more frequently in this case. As a conclusion, medium size packets (up to 500 flits) represent a good trade-off between packet size and the *time* required to deliver these packets with the XY routing algorithm. To corroborate this assumption, Table 2 presents a comparison of the total time to deliver a fixed amount of data, 100000 flits, varying the packet size. As expected, medium size packets

present superior performance.

Table 2 – Total time to deliver 100,000 flits in a 5x5 NoC, XY algorithm, load 70%.

	Time do deliver 100,000 flits (in clock cycles)	Observation
10,000 packets with 10 flits	27025	Time wasted with frequent arbitration/routing
1,000 packets with 100 flits	22474	
100 packets with 1,000 flits	28288	Larger packets induce more blocked paths

Network load is another important issue to address. It is possible to conclude from Fig. 8 that heavily loaded networks penalize adaptive algorithms. The explanation is again the increased blocking of paths in the middle of the network. As the traffic load decreases, the spacing between packets reduces blocking which favors adaptive routing. However, when the load reduces beyond a certain point, the network starts to become underutilized.

Finally, a coarse comparison between NoC and shared busses performance is possible. In Fig. 8 is possible to observe that flits are transmitted in average in less than 0.25 clock cycles, a performance impossible to obtain with single busses. If real bus architectures are considered, NoCs are expected to present at least one order of magnitude of gain in performance over busses.

6 Conclusion and Future Work

This paper answered two questions relevant to the design of NoCs. The first one is the choice of routing algorithm, where XY routing appears as the best one in most situations, for medium to large NoCs. The second question is the determination of the best compromise between packet size and total time to deliver the total load. Medium sized packets are the best choice, due to the network buffering capacity. Small packets underutilize the network and increase segmentation and reassembly overhead, while large packets lead to network congestion and buffer saturation.

As future works, it is possible to say that deeper traffic analyses are needed, using e. g. real traffic load distributions. Besides it is also important to consider the usage of NoCs with virtual channels, which modify blocking conditions and thus change traffic characteristics. Combining the results provided here and extensions to deal with traffic in virtual channels NoCs, it is possible to safely address the problem of designing NoCs with controlled quality of service (QoS).

References

[1] Martin, G.; Chang, H. System on Chip Design. In: 9th International Symposium on Integrated Circuits, Devices & Systems (ISIC'01), Tutorial 2, 2001.

[2] Kumar, S.; et al. A Network on Chip Architecture and Design Methodology. In: IEEE Computer Society Annual Symposium on VLSI. (ISVLSI'02), Apr. 2002, pp. 105-112.

[3] Benini, L.; De Micheli, G. Networks on chips: a new SoC paradigm. IEEE Computer, v. 35(1), Jan. 2002, pp. 70-78.

[4] Bolotin, E.; Cidon, I; Ginosar, R.; Kolodny, A. QNoC: QoS architecture and design process for Network on Chip. The Journal of Systems Architecture, Special Issue on Networks on Chip, 2004 (accepted for publication)..

[5] Moraes, F.; Mello, A.; Möller, L.; Ost, L.; Calazans, N. A Low Area Overhead Packet-switched Network on Chip: Architecture and Prototyping. In: IFIP Very Large Scale Integration (VLSI-SOC), 2003, pp 318-323.

[6] Moraes, F. G.; Calazans, N.; Mello, A.; Möller, L.; Ost, L. HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip. Integration VLSI Journal, 2003 (accepted for publication).

[7] Andriahantenaina, A.; et al. SPIN: a Scalable, Packet Switched, On-Chip Micro-network. In. Design Automation and Test in Europe Conference (DATE'2003), 2003, pp. 70-73.

[8] Liang, J.; Swaminathan, S.; Tessier, R. aSOC: A Scalable, Single-Chip communications Architecture. In: IEEE International Conference on Parallel Architectures and Compilation Techniques, Oct. 2000, pp. 37-46.

[9] Ni, L. M.; McKinley, P. K. A Survey of Wormhole Routing Techniques in Direct Networks. IEEE Computer Magazine, v.26(2), February, 1993, pp. 62-76.

[10] Mohapatra, P. Wormhole routing techniques for directly connected multicomputer systems. ACM Computing Surveys, 30(3), Sep. 1998, pp 374-410.

[11] Glass, C.; Ni, L. The Turn Model for Adaptive Routing. Journal of the Association for Computing Machinery, v. 41(5), Sep. 1994, pp. 874-902.

[12] Day, J.; Zimmermman, H. The OSI reference model. Proceedings of the IEEE, 71(12), Dec. 1983, pp. 1334-1340.

[13] Model Technology. ModelSim Foreign Language Interface. Version 5.8a, 2004.