SYCOSYS: An Integrated Framework for studying Interconnection Network Performance in Multiprocessor Systems

V. Puente, J.A. Gregorio and R. Beivide
Computer Architecture Group
University of Cantabria – Spain
{vpuente, jagm, mon}@atc.unican.es

Abstract – An environment has been developed which is capable of determining the impact that a multiprocessor interconnection subsystem causes on real application execution time. A general-purpose interconnection network simulator, called SICOSYS, able to capture essential aspects of the low-level implementation, has been integrated into two execution driven simulators for multiprocessors: RSIM and SimOS. The enhancement of both tools allows the analysis of new proposals for the interconnection subsystem of a cc-NUMA machine, from the VLSI level up to the real application level. Any new proposal can be translated to a specific message router architecture and by using a low-level implementation tool, the parameter delays of a detailed router model to be used by SICOSYS can be obtained. The pair RSIM-SICOSYS is adapted for simulating multiprocessor systems running preemptive parallel applications, such as intensive numerical ones, with hardly no operating system interference. However, for determining the impact that new proposals provoke on the execution time of parallel applications strongly interacting with the operating system, such as online transaction processing, the SimOS-SICOSYS environment is more adequate.

Keywords: Interconnection networks, multiprocessors, simulation tools, RSIM, SimOS, SICOSYS.

1. Introduction

The interconnection network is an essential element of multiprocessor systems and to determine its performance is critical. Although the number of works carried out in this direction have been numerous, there is a lack of completeness, mainly in two aspects. On the one hand, in a few occasions the impact of the low-level implementation is considered in any proposal. Thus, in spite of pioneering studies like Chien's work [3], it is usual to analize of new proposals without considering if the increase of complexity of its VLSI implementation will neutralize the supposed improvements. On the other hand, as opposed to what already happens with practically all other computer building blocks, the performance analysis of the interconnection network continues to be analyzed without paying too much attention to real working loads. However, it is evident that numerous proposals lose relevance when characteristics of traffic corresponding to real applications are considered.

The main reasons for the these deficiencies are both cost and complexity of their consideration. The simulation, at VLSI level, of an interconnection network of medium size (≈128 nodes) can take several days for obtaining some basic parameter. Also, to accurately simulate a few seconds of the traffic generated by a real parallel application can go beyond what could be considered as a reasonable design cycle time. Moreover, the situation is still worse if the special characteristics of structures as successful as Distributed Shared Memory machines, are considered. The coherence maintenance, normally by hardware, causes an uncertainty about the traffic applied to the network. An application will generate different traffic distribution depending on aspects such as data location, out-of-order execution, cache size, interference between processes, characteristics of the interconnection network, etc.

Obtaining real measures from this type of systems is not even feasible until very advanced stages of the design. The high cost prevents the construction of prototypes of multiprocessor systems, even with a reduced number of processors, without a guarantee of working. Analytical results are equally difficult to obtain. Although the parameters which take part in the performance of the interconnection network can be fixed, their stochastic relationships are complex, application dependent, and, therefore, it is difficult to obtain results without carrying out unacceptable approaches.

The only way to accurately determine the performance is simulation. However, the number and characteristics of the simulators are as diverse as research groups working on this subject. They can be composed of a few tens of code lines [10] (raw results at low cost), up to tens of thousands of lines of a VHDL simulator (higher precision at higher cost). For this reason, throughout the last five years, our research group has been developing a simulator called SICOSYS [14], which is able to incorporate the key parameters of the low-level implementation and provides results close to those from VHDL simulators, but at lower computational cost.

In this paper, the integration of this high-precision simulator in RSIM [9] and SimOS [12] is described. They are two of the most powerful public-domain simulation tools for multiprocessor systems. RSIM allows simulation of the characteristics of a superscalar processor, to the types of protocol used to maintain the data coherence. SimOS represents the following step in the simulation of a multiprocessor system, introducing the effect caused by the operating system. However, both simulators have the same drawback. Their capacity for modeling the interconnection network is not good enough. RSIM has a module, called NETSIM, to simulate the interconnection network of the system, but it presents serious limitations. For example, the topology is restricted to a mesh with constant transfer times. In SimOS the limitation is harder because a fixed latency for the interconnection network is always assumed.

The integration of SICOSYS in both simulators provides the first public-domain framework able to analyze the impact that the technological implementation of the interconnection subsystem has on the execution time of the applications, running in cc-Numa machines, with a accuracy degree close to a VLSI tool. In this way, real parallel applications can be executed in multiprocessor machines using the interconnection network that we are trying to optimize. At the moment, RSIM-SICOSYS integration is complete and the resulting tool has been widely tested with innumerable practical cases. Nevertheless, since SimOS is a more recent and complex tool than RSIM, its integration with the simulator SICOSYS is still in a verification phase.

The rest of the paper is structured as follows. The next section presents the basic characteristics of SICOSYS and its comparison with other tools. Section 3 describes the most significant aspects of its integration into RSIM and SimOS. Section 4 presents an application example showing the potential advantages and drawbacks of using the framework. Finally, in Section 5, the main conclusions and future work are mentioned.

2. Simulator of Communication Systems (SICOSYS)

SICOSYS is a general-purpose interconnection network simulator that allows the modeling a wide variety of message routers in a precise way. Results are very close to those obtained by using hardware simulators but at lower computational cost. In order to make the tool easily comprehensible, extensible and reusable, the design of the tool is object-oriented and its implementation is in C++ language.
The implementation of the simulator is based on technology intimately related to the OO design: *the design patterns* [4]. In particular, approximately 110 classes, distributed in about 50,000 lines of code have been necessary. The portability is very high and practically it can be executed in any UNIX platform with a C++ standard compiler.

2.1 Structure

The implementation of the simulator has been oriented towards a suitable structuring and a high portability. It can be divided in the following parts:

- **SGML constructors**. Responsible for interpreting the input files and generating the objects required for the simulation. There is a constructor for each one of the three existing input files.
- **Components**. They represent the hardware that is going to be simulated. The functional components of the router, their characteristics and associate delays are modeled. In this way, a direct relationship between a specific hardware structure and the model that our simulator generates is established. The constructors simply generate a hierarchy of components related to each other.
- **Traffic patterns**. Module in charge of the injection of packets into the network. When it is desired to use the simulator in a stand-alone way, the main traffic patterns (random, matrix transpose, reversal bit, perfect shuffle, hot-spot,..) can be used. For injecting traffic generated by the execution of real applications, an "empty" pattern has been defined with the aim of working integrated with other tools.
- **Simulator**. This module carries out the simulation process. Periodically, it generates information on the simulation state and finalizes by generating a file with the simulation results. Among them, the two basic figures of merit of interconnection network performance: packet latency (average and maximum) and network throughput.

Although the simulator has a graphical interface for its stand-alone operation, is not useful for describing the integration process.

2.2 Specification files

The tool provides a solution to the generic problem of interconnection network simulation. The same environment can be used to experiment with very different architectural proposals. The specification of the type of experiment is carried out through description files in SGML language [4]. These files describe three aspects:

- *Router hardware architecture*. Description of the elements (memories, switches, multiplexers,..) that connected to each other form the router.
- *Interconnection network*. Specifications of the type of network (mesh, torus,..), dimensions and interconnection delays.
- *Simulation parameters*. Specification of parameters such as traffic pattern, applied load, message length, etc.

The modification of these input files allows to carry out different experiments without recompiling the source code. This provokes a certain loss of performance in the initialization phase, but as a counterpart, the tool presents a homogenous form to carry out experiments and an easy-to-use interface. In this way, not only the router is fixed but also the functional and temporary characteristics extracted from VLSI levels are transferred at a superior level of abstraction.

Figure 1 shows an example of the three input files. "Router.sgm" is a SGML description of all architectural characteristics of a determinist router for toroidal networks with a deadlock avoidance mechanism based on injection control. "Network.sgm" describes both the form in which routers are interconnected and the connection delays. Finally, "Simula.sgm" specifies the remaining parameters of the simulation.

```
<! □□□□□□□□□□□ Example of a SGML router description-->

<Router id="DOR2D-BU" inputs=4 outputs=4 bufferSize=64 bufferControl=CT
routingControl="DOR-BU">
<Injector id="INJ">
<Consumer id="CONS">

<Buffer id="BUF1" type="X+" dataDelay=2>
. . . . . . . . . . .
<Buffer id="BUF5" type="Node" dataDelay=2>

<Routing id="RTG1" type="X+" headerDelay=1 dataDelay=0>
. . . . . . . . . . .
<Routing id="RTG5" type="Node" headerDelay=1 dataDelay=0>

<Crossbar id="CROSSBAR" inputs="5" outputs="5" type="CT" headerDelay=2
dataDelay=1>
<Input id=1 type="X+">
<Input id=2 type="X-">
. . . . . . . . . . .
<Output id=5 type="Node">
</Crossbar>

<Connection id="C01" source="INJ" destination="BUF5">
<Connection id="C02" source="CROSSBAR.5" destination="CONS">
. . . . . . . . . . .
<Connection id="C12" source="RTG5" destination="CROSSBAR.5">

<Input id="1" type="X+" wrapper="BUF1">
. . . . . . . . . . .
<Input id="4" type="Y-" wrapper="BUF4">

<Output id="1" type="X+" wrapper="CROSSBAR.1">
. . . . . . . . . . .
<Output id="4" type="Y-" wrapper="CROSSBAR.4">

</Router>

<!□□□□□□□□□□□□□ A name of a network file -->
<TorusNetwork id="T88-DOR2D-BU"
  sizeX=8 sizeY=8 router="DOR2D-BU" delay=0>

<! □□□□□□□□□□□ Simulation file -->
<Simulation id="output_file">
<Network id="T88-DOR2D-BU">
<SimulationCycles id=100000>
<TrafficPattern id="MODAL" type=”RANDOM”>


<MessageLength id=1>
</Simulation>
```

Figure 1. Example of SGML input files for a simulation.

2.3 Comparative analysis

The main objective of SICOSYS is to reach the precision of a VLSI simulator but at a computational cost close to a functional simulator. In order to verify the degree of accomplishment of this objective, a comparison example with each type of simulator is presented.

SICOSYS versus VHDL simulator

As a test element, a completely adaptive virtual cut-through router (Figure 2) has been selected. Each one of the router channels has the following elements: a synchronizer block for asynchronous communication with neighboring routers; a packet storage with FIFO policy; a routing decision unit to route packets to a suitable output port; and a crossbar for the switching function. The deadlock avoidance mechanism is based on restriction of injection [11].

Starting with a RTL router description in VHDL and following a conventional design methodology, a logic-level implementation of the router has been obtained, employing the ECPD07 design kit (0.7 µm) [13].

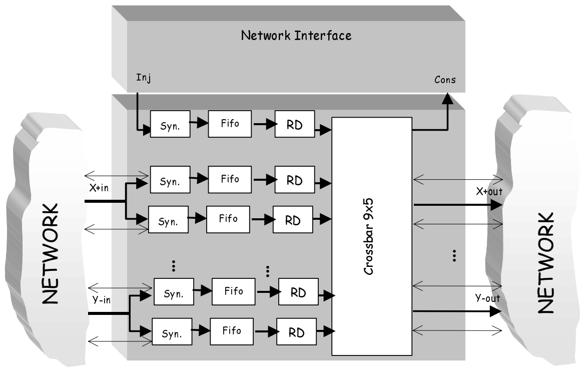
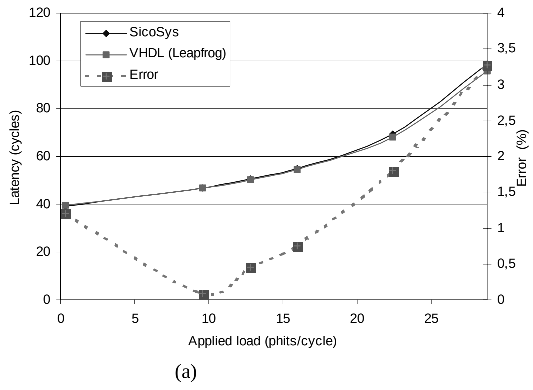


Figure 2. Example of the router used for comparison between SICOSYS and Leapfrog.

Next, using *leapfrog* (the VHDL simulator of Cadence [7]), a simulation of 64 nodes (each one using the RTL description) interconnected in a toroidal topology has been carried out. Latency and throughput results have been compared to those obtained by using SICOSYS, showing the goodness of our simulator, as can be seen in Figure 3. The latency error is below 4% and in the case of throughput, the error is even smaller. Meanwhile, the speed-up of SICOSYS can achieve up to 45 (see Figure 4). Similar results are obtained using other routers, with different traffic patterns and network configurations.



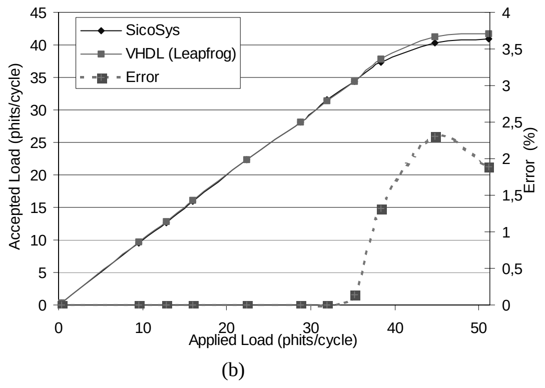


Figure 3. (a) Average latency and relative error of SICOSYS versus Leapfrog. (b) Throughput.

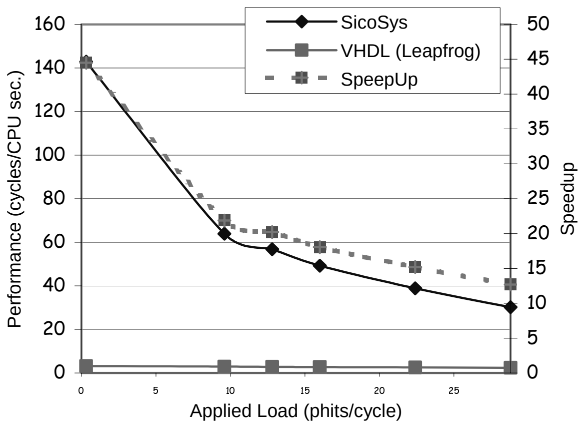


Figure 4. Performance of SICOSYS versus Leapfrog.

SICOSYS versus NETSIM

Originally developed as an independent tool, NETSIM is a module integrated in the execution-driven simulator RSIM [9]. It is able to simulate a direct interconnection network employing a specific C-library and three basic elements: buffers, multiplexers and demultiplexers. With the aim of comparing its performance with SICOSYS, several routers have been described in both simulators and their results compared employing synthetic loads.

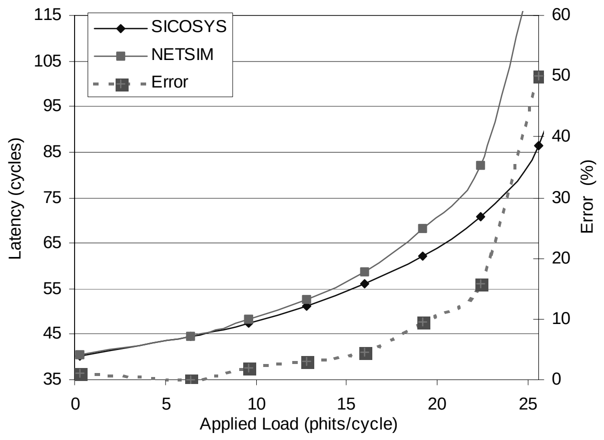


Figure 5. NETSIM –SICOSYS comparison (8x8 torus, deterministic routing and random traffic).

As an example, Figure 5 shows the results for one of the different cases analyzed. It represents the average message latency under uniform traffic. The network is an 8x8 torus, deterministic routing. Although the router simplicity is evident, the discrepancy between both types of simulators is up to 50%. And SICOSYS results remain closer to those obtained by the VHDL simulator.

These results demonstrate the strong discrepancies that can appear as a result of using functional simulators, even of the NETSIM characteristics. With such a degree of discrepancy it is not fair to try to analyze the impact of architectural proposals on the execution time of real applications.

3. Integrating SICOSYS into RSIM and SimOS

A tool like SICOSYS, able to capture the essential aspects of the low-level implementation, it is not sufficient to establish definitive results about the performance of an interconnection network. There is an enormous variability of results depending on the type of traffic used in the analysis and therefore, on the application that generates that traffic.

Thus, an application written for example using the message passing interface library MPI, generates a very different traffic pattern depending on whether it is executed in a PC cluster or in a cc-Numa machine. This does not mean that synthetic loads must be given up, but forces us to have tools able to directly provide traffic coming from the execution of real applications, because a proposal can lose or gain importance depending on such a traffic. For that reason, the following step has been to integrate SICOSYS within both execution driven simulators, RSIM and SimOS, obtaining a complete and trustworthy simulation environment.

3.1 RSIM-SICOSYS Integration

RSIM is an execution driven simulator, basically designed to study the behavior of shared memory architectures and to emulate current processors. The architecture of the memory hierarchy that RSIM is able to simulate, is very similar to the employed in the DASH multiprocessor [8], except each node of the system has only one processor.

RSIM directly interprets the application codes compiled and linked for SPARC processors. The support for the development of parallel applications is based on a specific library that incorporates the SystemV calls and an implementation of macros PARCMACS [1]. Distribution of the shared variables has to be made explicitly in the application code by using functions provided by the library.

Given the characteristics of both simulators, RSIM and SICOSYS, it is clear that their integration will give rise to a very complete simulation tool. The simplest way to carry out such an integration is to replace the module NETSIM by the simulator SICOSYS. The main problem was the simulation control. SICOSYS is a time-driven simulator but RSIM is essentially an event-driven simulator handled by the library YACSIM [6]. The solution adopted was to control the simulation driver by SICOSYS. Figure 6 shows the structure of the complete environment.

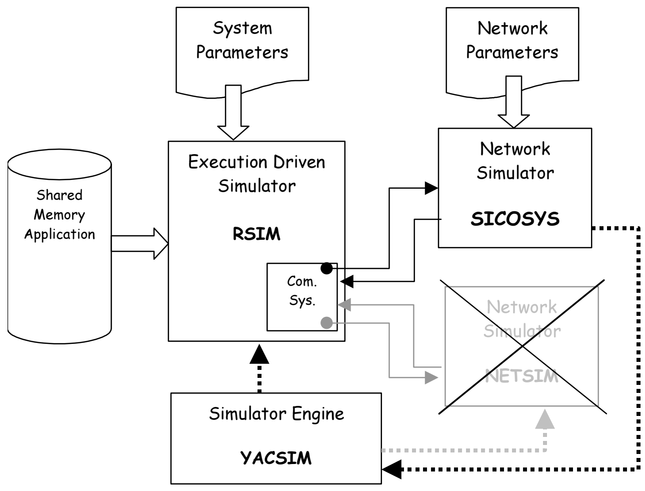


Figure 6. RSIM-SICOSYS integration.

The resulting tool is able to feed the accurate simulator SICOSYS with traffic directly generated by the execution of real applications. The interrelation between advanced processor characteristics and the technology used in the interconnection network implementation can be analyzed. For example, the effect on the application execution time caused by both the out-of-order execution of the processor and the type of message management of the router can be jointly analyzed. Thus, in a cc-Numa machine, the network performance affects the management of the cache lines of the processor and, in turn, the number of requests the processor sends back to the network. Therefore, for analyzing this type of systems the use of a precise simulation environment is imperative, specially in the last phases of the design.

Precision is usually paid her by increasing resources and simulation time. Nevertheless, the resulting tool RSIM-SICOSYS slows down less than 3 times, respect to RSIM. Moreover, a biprocessor version able to reduce this performance fall to 2 times has also been developed. As an example, Table 1 shows the simulation time of an FFT application of the suite SPLASH, running on a 64-processor system, with the default size problem (64K complex doubles). Simulations have been carried out in a SGI Power Challenge with MIPS R10000 processors at 200 Mhz, on IRIX6.2.

Application	RSIM-NETSIM (uni-processor)	RSIM-SICOSYS (uni-processor)	RSIM-SICOSYS (Bi-processor)
FFT 64 Processors	3767	9413	7126

Table 1. Simulation time (in seconds) of the different tools.

3.2 SimOS-SICOSYS Integration

We have also integrated SICOSYS with the more complete machine simulator, SimOS, because of several reasons. RSIM-SICOSYS is an environment adapted for the simulation of multiprocessor systems running preemptive parallel applications, such as intensive numerical ones. The number of processors, assigned by the operating system, remains constant during some interval, or until their conclusion, without hardly interference. However, there are other important applications, like the online transaction processing (OLTP), that interact with the operating system in a very intensive way. The kernel component can be up to 25% of the total time (user and kernel) [2]. Therefore, the committed error carrying out simulations of this type of applications without considering the operating system interference, can be inadmissible. Even numerical applications can suffer certain operating system interventions (for example, from the virtual memory manager) [5].

One of the few free distribution simulators able to emulate the behavior of the entire system, including an operating system, is SimOS [12]. This tool is able to simulate the hardware components of the multiprocessor system (processors, MMU, caches, disks, console,..) with enough detail for running real software. Although SimOS is also able to emulate the interconnection subsystem behavior, in this specific aspect, its deficiencies are important. Thus, while the impact on the application execution time of almost any processor modification can be analyzed, the message delivery time for the network is considered constant. Obviously, considering a network without contention can give rise to an important discrepancy in results. For example, Figure 7 shows the difference in the execution time of RADIX (a SPLAH application) employing the raw interconnection network model from SimOS against an accurate model from SICOSYS. The application has been compiled using the *sprocs* implementation of PARMACS and the emulated operating system is a modified version of SGI IRIX6.4.

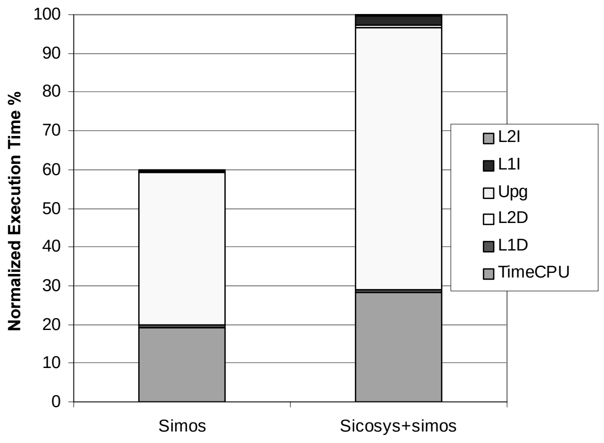


Figure 7. Effect of the accurate interconnection model of the SimOS-SICOSYS tool on the RADIX execution time.

Both simulators give rise to an environment with a great potential. From the point of view of the analysis of the interconnection subsystem, it opens the possibility to reflect the impact of any parameter on the execution time of real applications running under a commercial operating system control.

4. Application example

With the aim of showing how the developed environments can be used, next we present an application example using RSIM-SICOSYS (completely described in [11]). Figure 8.a shows the pipeline structure of a router (BDOR) for direct networks with a deadlock avoidance mechanism based on injection control. Packets reach their destination using dimension order routing (DOR) and the router's buffers are located at the input links. The question is to determine the impact of locating the buffers at the output links and routing packets in an adaptive fashion.

The new router (HPAR) can be divided into a number of stages like those shown in Figure 8.b. The greater complexity of the new router gives rise to an increase of the number of cycles a packet will need for crossing it. Nevertheless, due to the adaptivity and the elimination of a great part of the head-of-line blocking it seems feasible that the new router will improve the previous one.

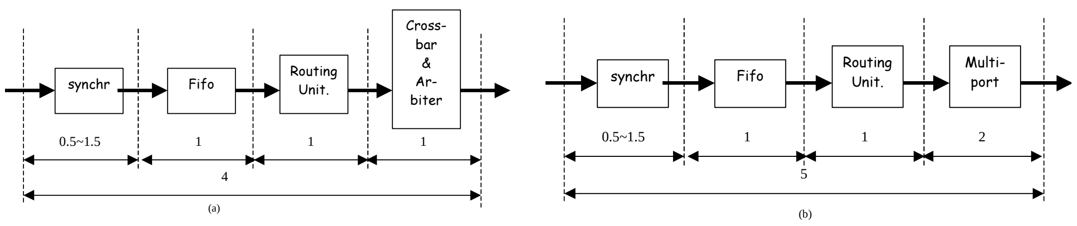
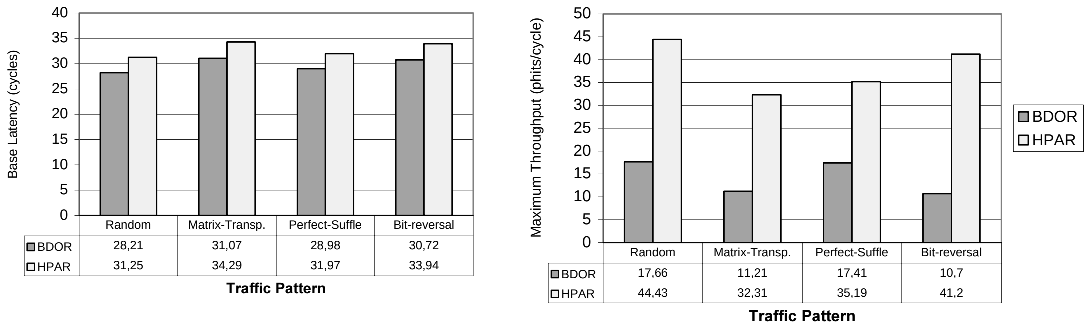


Figure 8. Pipeline Structures of the BDOR router (a), and the HPAR router (b).

Preliminary results can be easily obtained by using synthetic loads. Employing the usual traffic patterns of this type of analysis, it can be quickly proved that the increase of a cycle of the router structure translates to a greater base latency (time spent for a packet to arrive at destination, in load absence). Nevertheless, adaptivity and changing the buffer's location improve the new router performance up to 300% when it is integrated in a 64-node network (see Figure 9).



	Random	Matrix-Transp.	Perfect-Suffle	Bit-reversal
▨ BDOR	28,21	31,07	28,98	30,72
□ HPAR	31,25	34,29	31,97	33,94

	Random	Matrix-Transp.	Perfect-Suffle	Bit-reversal
▨ BDOR	17,66	11,21	17,41	10,7
□ HPAR	44,43	32,31	35,19	41,2

Figure 9. Results of latency and maximum throughput of both routers using synthetic loads for a 64-node network.

The question that always arises in these cases is what part of this gain will be transferred to the real applications. Only making use of simulation environments like the one developed in this work can give the correct answer at a reasonable cost. Thus, if the operating system does not interfere, the impact of router modifications on any real application can be analyzed using RSIM-SICOSYS.

As an example, Figure 10 shows the results for the Radix application. Although this application is highly communication demanding, the performance improvement of the proposal is lower than that forecast by SICOSYS using synthetic loads. The explanation comes from the data that RSIM-SICOSYS provides. In Figure 10.b the load applied to the network throughout the execution time of the application has been represented. There are phases of high and low communication demand. In the high demand phases, the HPAR router takes advantage of its higher maximum throughput and the application ends these phases quickly. However, during low demand phases the lower base latency of the simpler router (BDOR) allows the application to spent less time and the new proposal loses part of its advantage. The final result is that the new proposal improves the original router remarkably, but significantly less than the value predicted by using only synthetic loads.

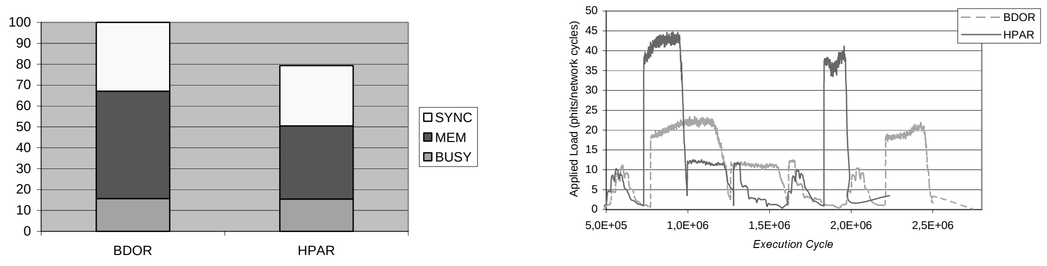


Figure 10 (a) Normalized execution time for Radix with 64 processors and 512 K integer keys. (b) Network performance analysis for Radix execution.

5. Conclusions and future work

As with other basic-building blocks of computers, to analyze the interconnection network it is essential to have sufficiently precise tools. In the initial stages of the design, synthetic loads or execution traces can be employed for analyzing new proposals. However, in the latter stages real loads must be used. Moreover, current superscalar processors with characteristics such as their out-of-order execution, oblige the use of execution-driven simulators. In fact, for analyzing the goodness of new proposals, not only the low-level implementation aspects have to be considered but in addition, the use of real application loads is essential. Important discrepancies can arise if inaccurate simulation tools are used.

The developed simulation environment, with the integration of SICOSYS into both RSIM and SimOS simulators, allows the transfer of the impact that technological implementations of the routers have on the execution time of applications. By means of the two powerful tools, RSIM-SICOSYS and SimOS-SICOSYS, it is possible to simulate the behavior of a medium size cc-NUMA machine, with an elevated level of detail (interconnection network included).

We think that this is the first public domain environment able to analyze the impact that a technological implementation of the interconnection subsystem has on the execution time of the applications with a degree of approximation close to VLSI domain.

At present, the effort is being directed fundamentally in three directions. In the first place, to improve the most stable environment, RSIM-SICOSYS. Secondly, to carry out exhaustive tests on the behavior of SimOS-SICOSYS with non-numerical working loads. And finally, trying to reduce the execution time of this new tool because the overhead of the operating system simulation is elevated.

Acknowledgements

We would like to thank Robert Kunz for his help during the installation and test of SimOS. We are also grateful to SGI for facilitating the use of the IRIX kernel.

6. References

[1] E. Artiaga, N. Navarro, X. Martorell and Y. Becerra, "Implementing PARMACS Macros for Shared-Memory Multiprocessor Environments", Dpto. of Computer Architecture (DAC-UPC), Report UPC-DAC-1997-07.

[2] L.A. Barroso, K. Gharachorloo, A. Nowatzyk and B. Verghese, "Impact of Chip-Level Integration on Performance of OLTP Workloads", HPCA-6, January 2000.

[3] A. Chien, "A cost and Speed Model for k-ary n-cube Wormhole Router", Hot Interconnects, August 1993.

[4] E. Gamma et al. "Elements of Reusable Object-Oriented Software", Addison-Wesley Professional Computing Series, 1995.

[5] J. Gibson, R. Kunz, D. Ofelt, M. Horowitz, J. Hennessy, and M. Heinrich. "FLASH vs. (Simulated) FLASH: Closing the Simulation Loop". Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp. 49-58, November 2000.

[6] J.R. Jump, "YACSIM Reference Manual", Rice University, Electrical and Computer Engineering Department, March 1993.

[7] Cadence, "Advanced Computer-Aided Engineering Tools", Verilog XL Reference Manual, 1999.

[8] D. Lenoski, J. Laudon, K. Gharachorloo, W. Weber, A. Gupta, J. Hennessy, M. Horowitz and M.S. Lam, "The Stanford Dash Multiprocessor", IEEE Computer, Vol 25, no.3, pp. 63-79, March 1992.

[9] 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, October 1997.

[10] M.J. Pertel, "A Simple Simulator for Multicomputer Routing Networks", Caltech Computer Science TR, March 1992.

[11] V. Puente, C. Izu, R. Beivide, J.A. Gregorio, F. Vallejo and J.M. Prellezo, "The Adaptive Bubble Router", Journal on Parallel and Distributed Computing. To be published in August, 2001

[12] M. Rosenblum, E. Bugnion, S.A. Herrod, and S. Devine, "Using the SimOS machine simulator to study complex computer systems", ACM Transactions on Modeling and Computer Simulation, 7(1):78:103, Jan. 1997.

[13] ES2/ATMEL ECPD07 Design Kit Reference Manual(0.7 μm).

[14] J.M. Prellezo, V. Puente, J.A. Gregorio y R. Beivide, "SICOSYS: Un Simulador de Redes de Interconexión para Computadores Paralelos", (in Spanish) Actas de las IX Jornadas de Paralelismo. San Sebastian - September de 1998.

[15] S. Woo, M. Ohara, E. Torrie, J.P. Singh, and A. Gupta. “The SPLASH-2 programs: Characterization and methodological considerations”, Proceedings of the 22nd International Symposium on Computer Architecture, June 1995.