Chapter 3

Performance Analysis

The previous chapter has described the different modifications that SICOSYS has overcome in order to become a useful tool for the future. This chapter will quantify the improvement levels obtained by such modifications.

3.1 Optimization of Internal Structure

The internal structure of SICOSYS has three major changes, the management of messages by pointers, the improvement in the queue implementation and the use of a message pool. To visualize the effect of the different changes, four snapshots of the code have been used to simulate a wide selection of situations. The four snapshots represent the state of SICOSYS through its development as follows.

a) is the original simulator with no changes at all.

b) is a simulator which manages the messages with pointers, though it uses `new` and `delete` for their allocation and deallocation.

c) it includes the new queue in which the final element does not have to be searched for.

d) is the final simulator with the message pool that reduces the cost of the allocation and deallocation of the messages.

These four simulators fill be fed with a wide range of simulations, spanning typical simulation scenarios. Three torus networks with sizes 4, 64 and 1024 nodes have been used. The message injection is done with a random pattern with normalized load of 0.01, 0.2, 0.5 and 0.9. Each of these environments is simulated for 50000 and 200000 cycles.

The simulation time of each modified simulator (b,c,d) is compared to that of the original (a) and put in graphs. Figure 3.1 shows the speedup of the simulator with a deterministic bubble router.

The graphs show the improvement of the simulation time obtained by each modification in different situations. In general, it can be observed that with low load the most significant improvement is the manipulation of messages with pointers (b), this is because the messages spend most of their lifetime traversing the network and do not wait a long time at injection queues. However, at higher loads messages get held for long times in the injection queues, causing these to grow unlimitedly. Therefore, especially at very high loads, it is the new queue (c) that speeds up the simulations most. However, when bigger networks are simulated the major improvement is provided by the message manipulation (b).

The modifications in the internal structure have lead to cut down simulation times at least in 80% (1.25 times faster) when low load cases are simulated, and down to 1.4% (70 times faster) when small networks are simulated for very long. It must be noted that the accuracy of the simulator remains untouched, as only the simulator's structure has been modified and the models are kept the same. In addition, the amount of memory used by the simulator has not diminished either.

3.2 Model Simplification

Even though the performance of the improved simulator is good, there are occasions in which there is a need for faster simulations. To this respect a new way of modeling routers was posed. It was capable of grouping all the components of a router in a single one. Thus reducing the overhead of inter-component communication.

In order to evaluate the effectiveness of this new approach, its accuracy bounds must be found. Therefore, two routers have been modeled, a very simple one and a complex one. This section will try to quantify the advantages in the performance of these new models and measure the loss of accuracy they exhibit.

To achieve this, several sets of simulations have been performed with the two versions of both routers. The selected simulations try to cover typical simulation scenarios.

3.2.1 The Deterministic Bubble Router

The simplicity of the bubble router leaves little room for optimizations and at the same time allows a simplified model to approach the real router fairly well. This fact can be observed in the following graphs (See figures 3.2, 3.3 and 3.4). Each

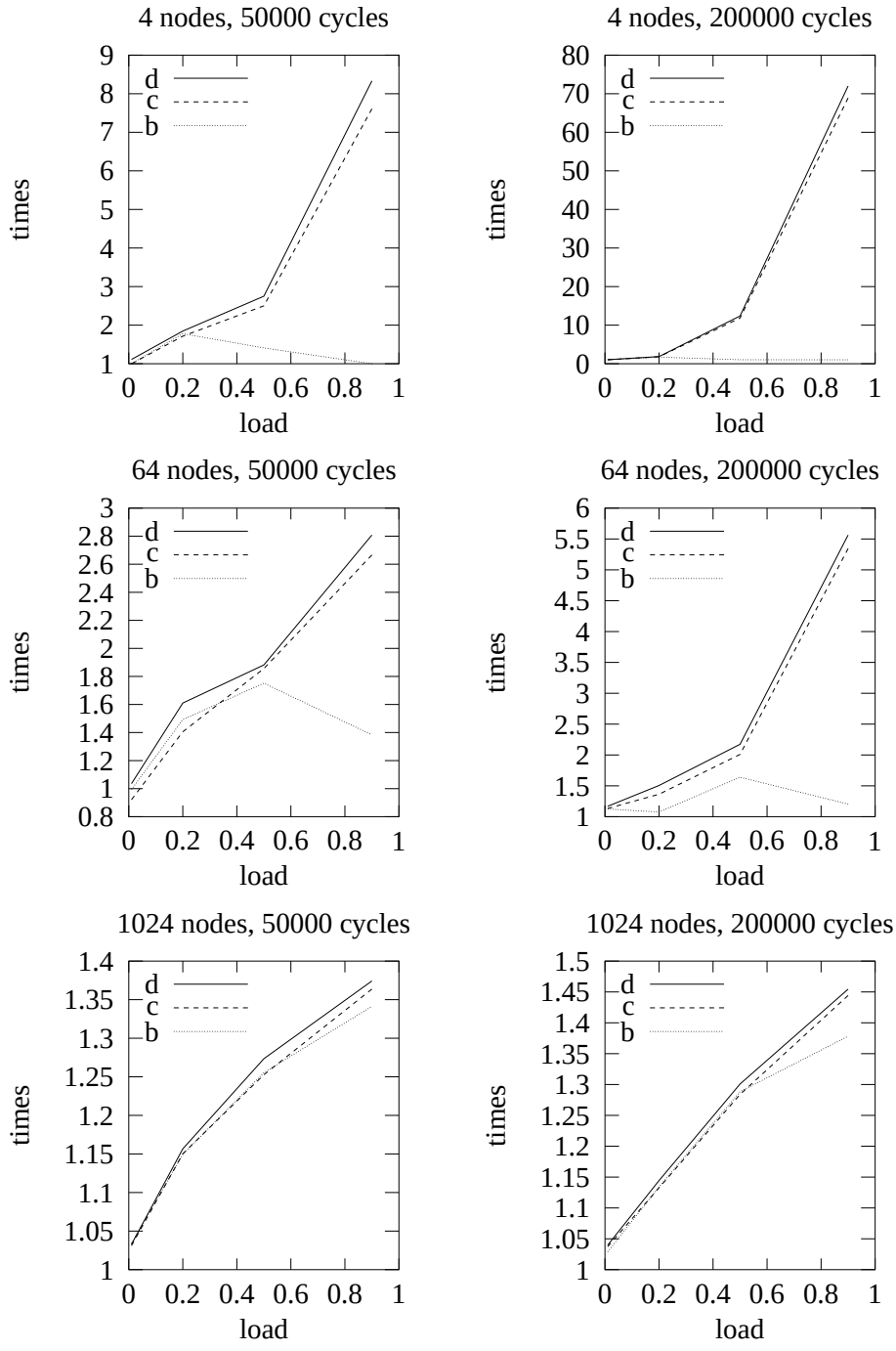

Figure 3.1: Improvement of SICOSYS's performance with a DOR bubble router

figure represents the results obtained from a set of simulations with a torus network of a particular number of nodes that spans different load conditions. Within each figure, there are four graphs. The latency and throughput of each model is represented by the leftmost graphs. On the right the simulation time improvement (speedup) is at the top and the relative error of both, latency and throughput, is at the bottom.

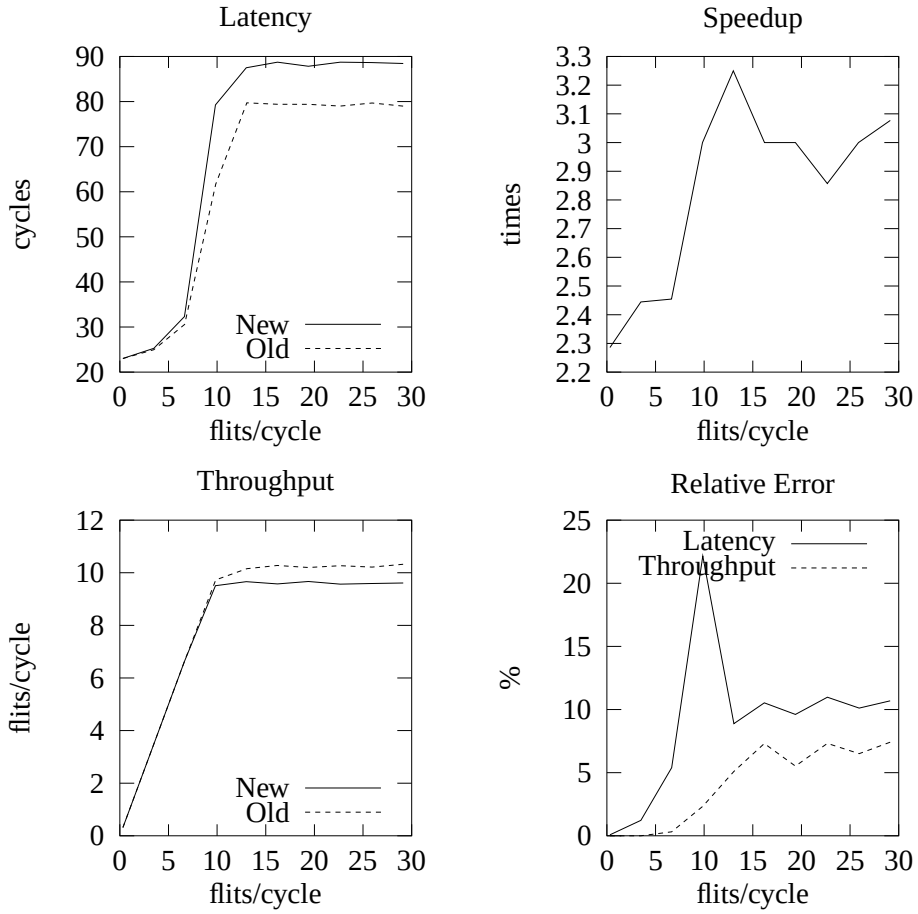

Figure 3.2: Performance of the simplified model of the deterministic bubble router. 16 node torus network.

When constructing the new model, a special effort was put in adjusting the base latency. This is the latency of the router when packets flow without contention. Thus the latency at low loads has zero error. Nevertheless, the simplification of the model does not mimic the crossbar precisely and, as load increases, the discrepancy in the latency grows. The latency error becomes maximum when

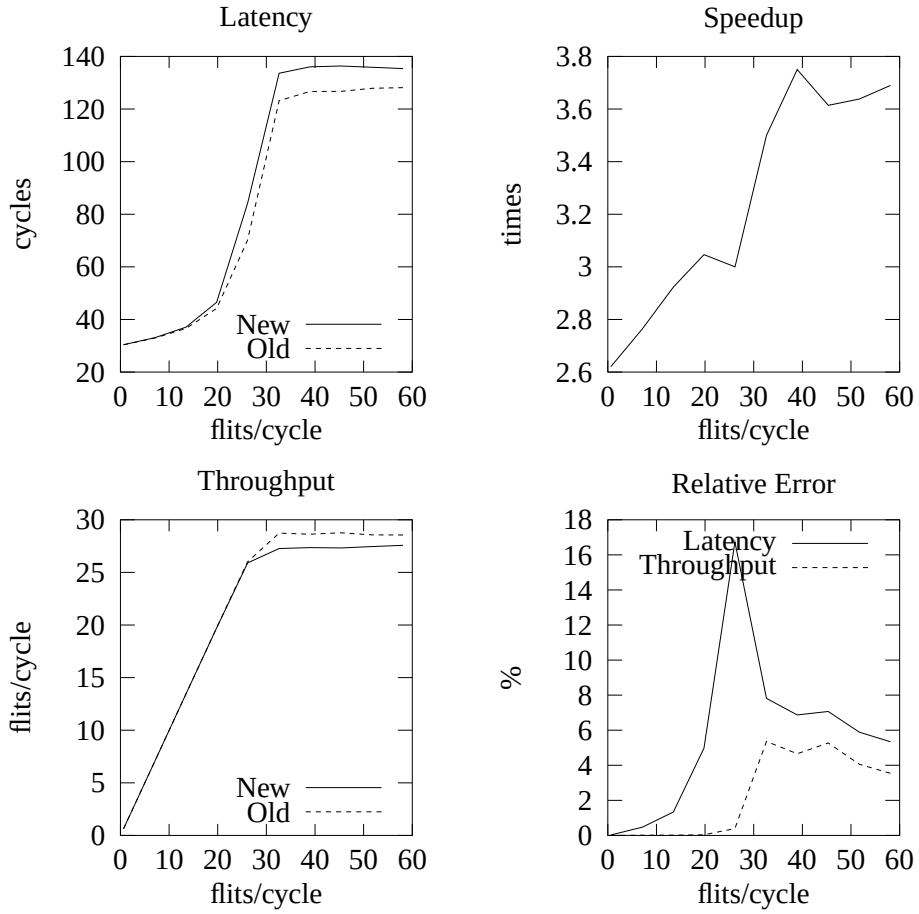

Figure 3.3: Performance of the simplified model of the deterministic bubble router. 64 node torus network.

the network is between the linear and the saturation areas. This is due to the huge slope of the latency in this area that provokes big errors even when minor details of the router are left out. However the error decreases just as drastically to a reasonable value when the network is saturated.

On the other hand, the throughput has a different behavior. When in the linear zone, the throughput is equal to the input load, this must be true for both routers. However, when approaching the saturation zone, a discrepancy between both routers appears. Unlike the latency error, the throughput error grows slightly and stabilizes in the saturation zone. As can be seen, the results obtained by the simple model are somewhat pessimistic. This means that the details left out by the simple model make the router perform worse and thus give an upper bound for the latency and a lower bound for the throughput. So being the average speedup

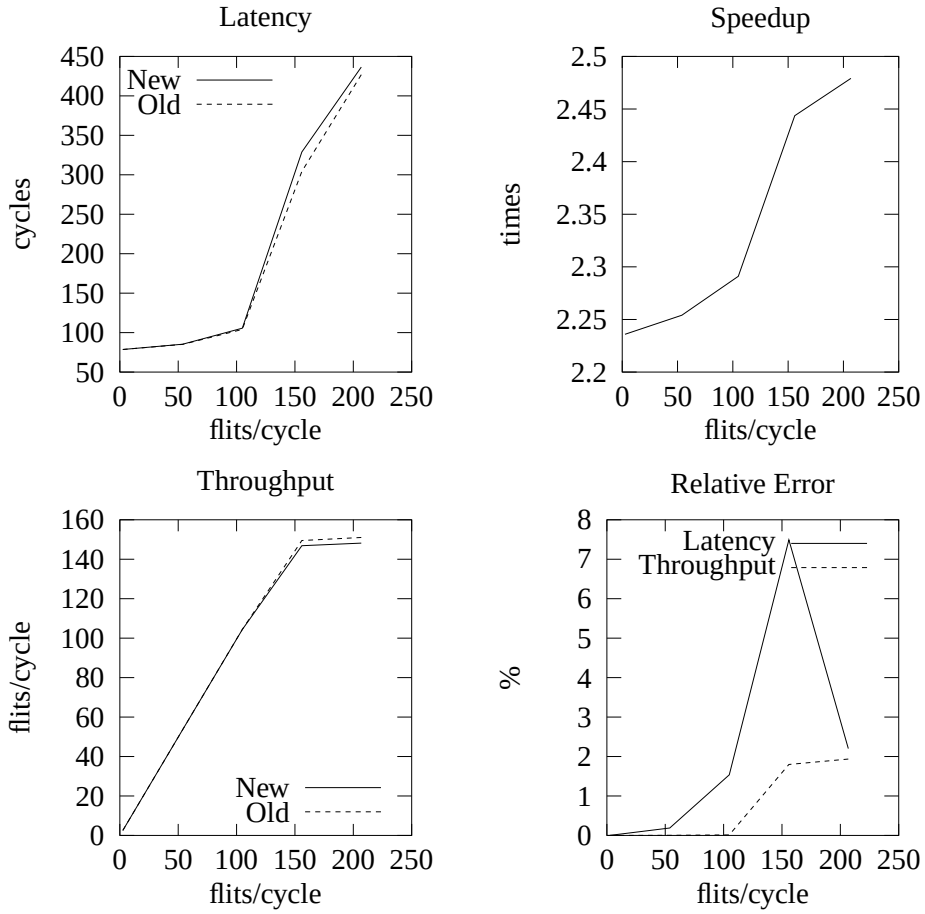

Figure 3.4: Performance of the simplified model of the deterministic bubble router. 1024 node torus network.

around 3, a worst case bound of what the real router will surely achieve can be obtained saving 66% of the simulation time.

In addition to the speedup reached by the new model, there is also an interesting parameter that is also reduced. This is the memory usage of the simulator. The reduction of the size of the simulator can make it perform even faster. As there is a bigger chance of fitting the process in the L2 caches of the machine, a reduction of time-consuming cache faults can be noticed. When running SICOSYS with the new router with various network sizes an average 80% reduction in memory usage was observed in the size of the simulator.

3.2.2 The Complex Adaptative Router

To get an estimation of the upper bound of the error a complex adaptative router has been implemented. The complexity of it leaves a lot of room for simplifications and, therefore, the saving of a lot of simulation time and memory.

In fact the adaptative router has many components and connections within it. When using the new monolithic scheme, all the connections and interface objects are eliminated. In addition, the individual models of all the components are compound in a single model. This enables the drastic simplification of the router, cutting down simulation time and memory usage. Then again, the error that this router is bigger than in the deterministic one.

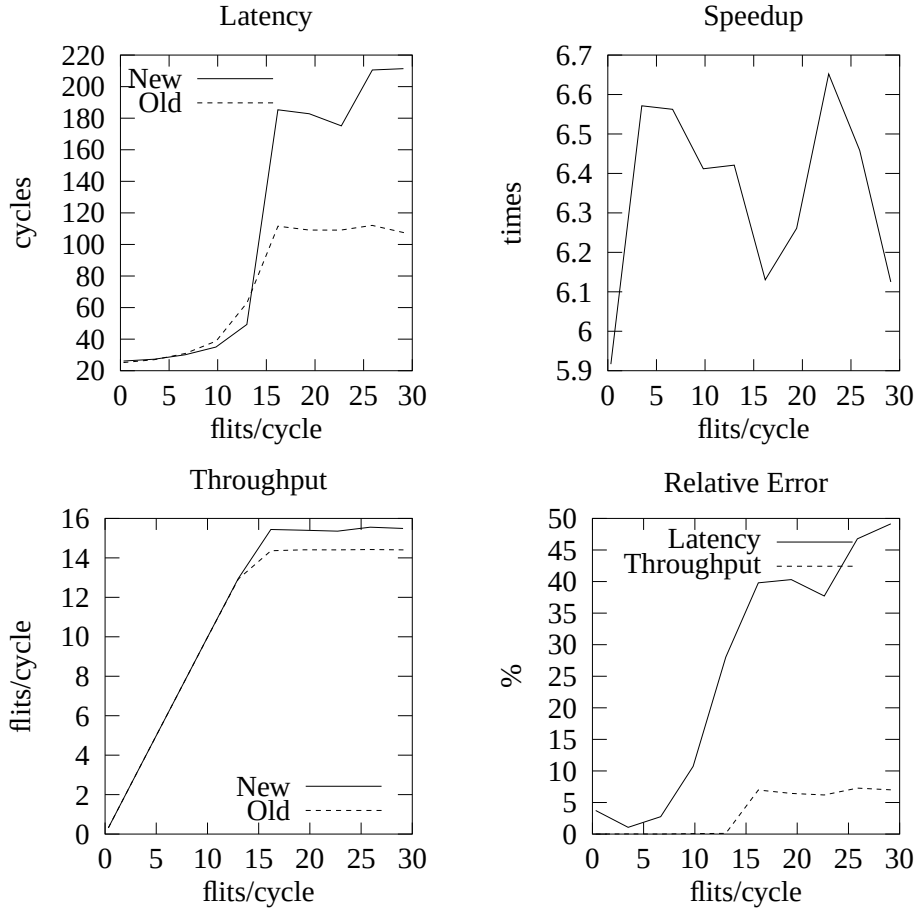

Figure 3.5: Performance of the simplified model of the adaptative router. 16 node torus network

To profile the adaptative router a smaller range of network sizes have been used. The cost of simulating a 1024 node torus network with the old adaptative

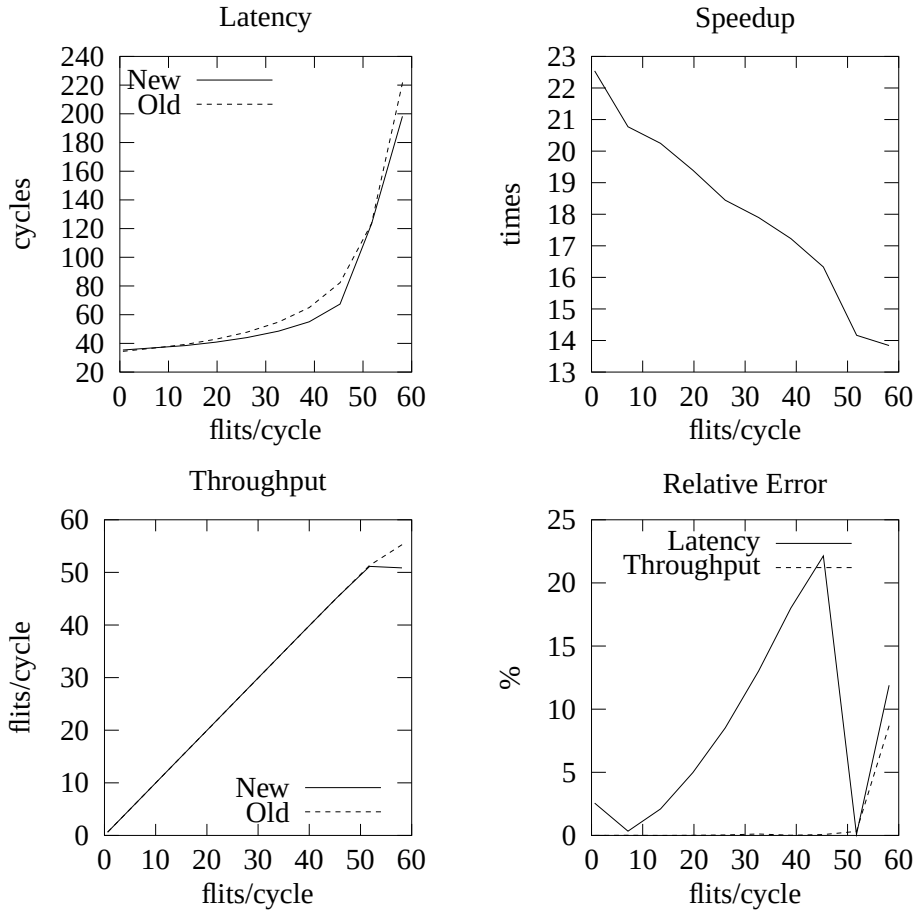

Figure 3.6: Performance of the simplified model of the adaptative router. 64 node torus network

router would be too big. Thus, the chosen network sizes are 16, 64 and 256. The results of the simulations are shown in figures 3.5, 3.6 and 3.7.

Just as with the deterministic router, the design of the internal structure payed special attention to the base latency. This fact is exposed by the the results of the simulations. The error in the latency is very small at low loads. However, as the load is increased the discrepancy between both routers grows. On the other hand, the throughput of both routers necessarily follow each other when working in the linear zone, but when in the saturation area, the new router shows a different performance.

As was expected, the error shown by the adaptative router is somewhat bigger than in the deterministic one. This is caused by the stronger simplifications made to the model to enable its monolithic implementation. It must be kept in mind that,

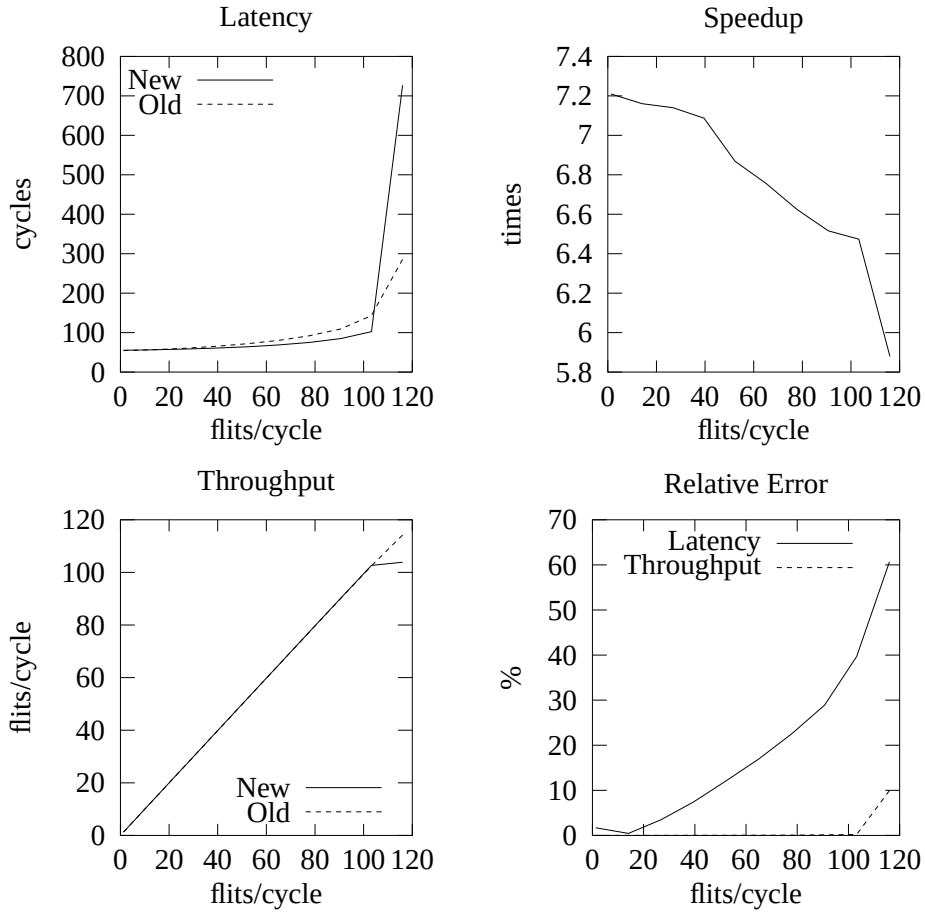

Figure 3.7: Performance of the simplified model of the adaptative router. 256 node torus network

although this router has a significant error, it is still a perfectly working router. It may not be suitable for tuning router parameters, but its a very good alternative to experiment other parts of a parallel machine. Having reached speedup ranges from 6 up to 22 and reducing the size of the simulator's memory usage down to 47% this router constitutes a very good tool to test networks with more than 1024 nodes or simulate real traffic.

3.2.3 Conclusions

The accuracy and performance of the two models has been measured, this has given a set of bounds for the accuracy of this new model strategy. Globally the latency error, the biggest of both, ranges from 16% to 60% while speedup lies in

the range of 2 to 22. However the higher errors and lower speedups occur in the saturation zone. When connecting SICOSYS to SimOS, only the lower load zone is used. If only the lower load area is observed, say 0 to 0.4, the latency error lies in the range of 5% to 10%. This will help giving a good approximation to the behavior of the network at a lower computational cost.

Chapter 4

Conclusion

4.1 Achievements

At the beginning of the project, there was a simulator that did not satisfy current demands. This simulator, SICOSYS, had very good qualities and there was a lot of effort put in it. Although its design posed no limits to the size and complexity of the simulations, the time it took to perform them was exceedingly high. By thoroughly studying its code a way of optimizing it has been found. This has allowed to deliver a renewed simulator capable of meeting actual simulation demands.

Being this optimization not enough for certain applications, a new way of implementing routers has been devised. This method allows the implementation of the whole router as a single component. Opposed to the multi-component architecture available in SICOSYS, this new technique speeds up the simulations and reduces the size of the simulator in memory at the cost of a loss of accuracy.

These improvements have made SICOSYS a tool for the future. The versatility of its design in combination with its optimized core allowed its integration with SimOS. At the time of writing several simulations have been performed showing excellent results.

Although the multicomponent models exhibit better performance, the monolithic router component allows the implementation of routers in a very compact an optimized way. This makes it an ideal candidate to build early prototypes of new routers.

4.2 Future Developments

The ever growing computational cost of the simulations will compel SICOSYS to improve its performance even further. Although SICOSYS has been run on parallel machines it is a single threaded application that takes no advantage of them.

Up to now multiple instances of the program were run at a time with different conditions. Nevertheless, when performing coosimulation, such as with SimOS, there must be only one instance of SICOSYS and it may not meet performance requirements in the future. One possible solution for this may be the parallelization of the simulator in order to make it take full advantage of the available parallel infrastructure.

Bibliography

[1] C. Carrión, R. Beivide, J.A. Gregorio, F. Vallejo, "A Flow Control Mechanism to Avoid Message Deadlock in k-ary n-cube Networks", International Conference on High Performance Computing, India, December 1997.

[2] W.J. Dally, "Performance Analysis of k.ary n-cube Interconnection Networks", IEEE Transactions on Computers, Vol. 39, no. 6, pp. 775-785, June 1990.

[3] J. Duato, "A Necessary and Sufficient Condition for Deadlock-Free Routing in Cut-Through and Store-and-Forward Networks", IEEE Transactions on Parallel and Distributed Systems, Vol. 7, no. 8, pp. 841-854, August 1996.

[4] J. Duato, S Yalamanchill, L. Ni, "Interconnection Networks", IEEE Computer Society, 1997.

[5] R. Hempel, "The MPI Standard for Message Passing", Lecture Notes in Computer Science, Vol. 797, pp. 247-252, 1994.

[6] P. Kermani, R Kleinrock, "Virtual cut-through: a new computer communication switching technique", Computer Networks, Vol. 3, no. 4, pp. 267-286, 1979.

[7] A.R. Larzelere, "Creating Simulation Capabilities", IEEE Computational Science & Engineering. Vol 5, no. 1, pp 27-35, January/March 1998.

[8] V.S. Pai et al., "RSIM: An Execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors". IEEE TCCA Newsletter, October 1997.

[9] J.M. Prellezo, V. Puente, J.A. Gregorio, R. Beivide, "SICOSYS: Un Simulador de Redes de Interconexión para Computadores Paralelos", VIII Jornadas de Paralelismo, Septiembre 1998.

[10] V. Puente, J.A. Gregorio, C. Izu, R. Beivide, "Impact of Head-of-Line Blocking on Parallel Computer Networks: Hardware to Applications", Europar'99, September 1999.

[11] M. Rosenblum, E. Bugnion, S. Devine, S. Herrod, "Using the SimOS Machine Simulator to Study Complex Computer Systems", ACM TOMACS Special Issue on Computer Simulation, 1997.

[12] S.C. Woo, M. Ohara, E. Torrie, J.P. Sing, A. Gupta, "The SPLASH-2 programs: Characterization and Methodological Considerations", Proceedings of the 22nd ISCA, pp. 24-37, June 1995.

[13] `http://www.top500.org`

Appendix A

Introduction to SICOSYS

It is frequently seen, in the analysis of interconnection networks, that analytical and simulation techniques often cooperate. The use of analytical tools is compromised by the complexity of the system under study. This is because normally the system must be simplified in order to allow its analysis. Frequently, these simplifications make the model not very similar to the system. In these cases, the way of obtaining convincing data could be by doing experiments with the real system. But this is not always affordable because the system is not normally built at the time of its study. This only leaves the possibility of simulating the system. Though this involves modeling the system as well, the computing infrastructure supporting the simulation process allows a great approximation of the model to the system.

When modeling a system, there must be a selection of the aspects that will conform the model. This is, the model should only have the features that determine the functionality of the system, and not others that can unnecessarily complicate it. Therefore, the difficulty of building a model lies on the appropriate choice of the system's features depending on the purpose of the analysis.

Simulation is a numeric technique that performs experiments on a given model in order to extract data that describes the functioning of the modeled system. Normally, a simulation involves a huge amount of operations and variables. Thus, simulation is nearly always done with the aid of computers.

When using a simulator, there are various things the must be kept in mind:

- The execution of a simulation can be very long. Therefore the model must be kept as simple as possible while resembling the system's features in maximum detail.

- As the results depend on the manipulation of random variables. There should be an averaging of many experiments in order to get a proper result.

- Seldom there is a model that mimics the system completely. There is always some detail that is not taken into account. Hence, the results of the simulation are never to be interpreted as the performance of the real system.

- In many cases the simulation operates in a transient phase before reaching a stationary state. To get results of the stationary state, no statistical measures should be taken during the transient phase.

The most precise way of simulating interconnection networks is by simulating a hardware implementation (at logical or physical level) of the network and routers. This procedure allows the simulation of details down to the transistor level. Anyway these simulations are very expensive in terms of simulation time and memory usage. Hence, the group of Computer Architecture and Technology at the University of Cantabria developed a simulation environment that can perform simulations of interconnection networks with less resources while achieving a very high accuracy. This tool gets the name SICOSYS form "Simulator of Communication Systems".

SICOSYS is a time driven simulator developed in C++ having in mind modularity, versatility and connectivity with other systems. The models used are intended to resemble the hardware description. In this way, the simulator mimics the hardware structure of the routers instead of just implementing their functionality.

Another objective of the project was giving a tool that could implement a wide variety of routers and networks and presented a homogeneous and simple interface to the user. Thus, SICOSYS has a collection of components like multiplexers, buffers or crossbars. And these components can connected to each other to conform a router and these, in turn, are connected in a certain network fashion. All this is defined in SGML which can be thought of as a superset of HTML. In fact, all the configuration that SICOSYS needs is written in SGML that makes it easy to understand and code to the user.