CINSim – A COMPONENT-BASED INTERCONNECTION NETWORK SIMULATOR FOR MODELING DYNAMIC RECONFIGURATION*

Dietmar Tutsch, Daniel Lüdtke, Arvid Walter, and Matthias Kühm
Technische Universität Berlin
Real–Time Systems and Robotics
D–10587 Berlin, Germany
{dietmart,dluedtke,asas,kuehmi}@cs.tu-berlin.de

KEYWORDS
stochastic simulation, performance modeling, analytical and numerical simulation, communication networks, computer systems

ABSTRACT
This paper presents a simulator for modeling dynamic reconfiguration of component-based interconnection networks. It is able to model and simulate all kinds of interconnection networks that are composed of switches, buffers, sources, and destinations. Steady-state simulation can be applied as well as terminating simulation to observe any transient behavior of the network. Due to the involved stochastic events, e.g. randomly generated traffic, confidence levels and estimated precisions are calculated to determine the accuracy of the results. Dynamic reconfiguration of networks can also be modeled by this simulator. Thus, the reconfiguration process can be observed to investigate the particular network behavior during this time interval. To the best of the authors' knowledge that is the first simulator that is able to deal with dynamically reconfigurable interconnection networks.

1 INTRODUCTION

Networks used to establish Ethernet switches or ATM switches consist of a similar topology. Further on, the same topologies also build the networks of multiprocessor systems and they are under investigation to be applied for networks on chip (NoC).

As an example, multistage interconnection networks (MINs) represent such a topology. They are frequently proposed to connect the nodes of a multiprocessor system (e.g. the IBM SP2 [1]) or to establish network switches (e.g. the Fujitsu FETEX-150 [2]). Thus, many investigations about MINs exist. A major task in these investigations deals with the performance evaluation of the MIN topology in question. The performance is usually determined by modeling, using simulation [3] or mathematical methods [4].

Mathematical methods are used by [5], for instance. Group communication in circuit switched MINs is investigated by applying Markov chains as a modeling technique. Markov chains are also used in [6] to compare the MIN performance in the case of different buffering schemes. Hot spot traffic performance in MINs is examined by [7]. [8] deals with multicast in Clos networks as a subclass of MINs. One of these authors also published [9] where MINs are used to establish active routers.

One of the drawbacks of the mathematical modeling methods is the high model development time: the finer the granularity with which the interconnection network is represented, the more complex the model becomes and the more development time is needed. That also means the model usually bases on many simplifications and assumptions to reduce the complexity. Thus, the performance results obtained by the model may become inaccurate. Additionally, many network parameters cannot be investigated because they are neglected and they are not incorporated into the model.

In consequence, simulation gives an alternative to mathematical methods. Its advantages are a more detailed network description and a shorter development time. Transient parameter observations can be performed which is more difficult with mathematical models. Nevertheless, the simulation of very large networks suffers from the long simulation run times, particularly if any stochastic events are involved and confidence levels must be fulfilled. But for reasonable network sizes and in all cases where mathematical methods cannot be applied, simulation is a good option.

This paper presents a new simulator for component-based interconnection networks. It was particularly designed to model irregular multistage interconnection networks in contrast to *MINSimulate* [3] which could only simulate regular structures of certain topologies. The new simulator, which is called *CINSim* (*C*omponent-based *I*nterconnection *N*etwork *Sim*ulator), is much more flexible. It can model any kind of interconnection network that consists of components such as buffers, switches, sources, and destinations.

Further on, this simulator is able to model the dynamic reconfiguration [10, 11] of interconnection networks. Dynamic reconfiguration of networks means that the network hardware changes during operation. That includes the network topology, the buffer and switch locations and sizes, as well as routing, switching, and scheduling strategies. The reconfiguration can be motivated by a changing network traffic, requiring fast connections between certain input-output pairs while other connections are not claiming such a short delay time [12]. Reconfiguration becomes more and more popular because today's technology provides fast and cheap

*This research is supported by Deutsche Forschungsgemeinschaft (DFG) under Grant Ho 1257/22-1 within the Priority Programme 1148 "Rekonfigurierbare Rechensysteme".

Proceedings 12th International Conference ASMTA 2005
Khalid Al-Begain, Gunter Bolch, Miklos Telek © ECMS, 2005
ISBN 1-84233-112-4 (Set) / ISBN 1-84233-113-2 (CD)

hardware solutions for realizing dynamic and partial reconfiguration [13], e.g. FPGAs (field programmable gate arrays) [14].

The paper is organized as follows: The topologies of interconnection networks that the new simulator is able to handle are described in Section 2. Section 3 presents the structure of the simulator, its features and benefits. The current and future work is outlined in Section 4 and some results obtained by the simulator are depicted. Section 5 summarizes and gives conclusions.

2 INTERCONNECTION NETWORKS

The term "component-based interconnection networks" describes networks that consist of any combination of the components buffer, switch, source, and destination. The new simulator is able to model such kind of networks. These networks can be divided into two classes: networks of regular structure and networks of irregular structure.

Networks of regular structure that consist of buffers and switches (particularly: crossbars) are for instance the well-known multistage interconnection networks (MINs). There, crossbars are arranged in stages and connected by interstage links. Often, buffers are introduced at the inputs or outputs of the crossbars. The link structure of the interstage links and the amount of crossbars characterizes the MIN.

MINs [9, 6, 5] with the banyan property are networks where a unique path exists from an input to an output. Such MINs of size $N \times N$ (N inputs and N outputs numbered from 0 to $N-1$, respectively) consist of $c \times c$ crossbars (crossbars of c inputs and c outputs numbered from 0 to $c-1$, respectively) [15] with $N = c^n$ and $n, c, N \in \mathbb{N}$. The number of stages is given by n. Figure 1 depicts a 3-stage MIN.

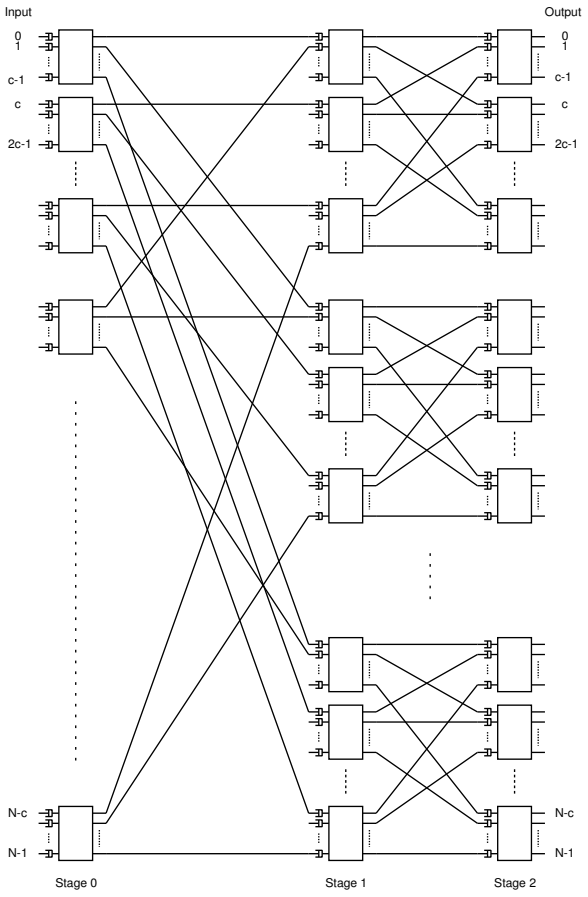

Figure 1: 3-Stage MIN consisting of $c \times c$ Crossbars

To achieve synchronously operating crossbars, the network is internally clocked. If buffers are applied, a message to be transferred is divided into packets and the network deals with packet switching. In this paper, packets of equal size are assumed. The packets usually consist of a header representing the routing tag and of a payload field filled with the corresponding part of the message. The routing tag is given as a binary N-digit number, each digit representing a network destination output. Then, the routing can be performed such that at each network stage, the digits are interpreted resulting in the desired output numbers of the current crossbar [16]. That means the crossbar must provide router properties.

The buffers are located in each stage k ($k \in \mathbb{N}$ and $0 \leq k \leq n-1$), e.g. as crossbar input buffers [17, 6] as shown in Figure 1. They are usually established as FIFO buffers to store a maximum number of m_{max} packets ($m_{max} \in \mathbb{N}$). The packets move by store-and-forward switching or virtual cut-through switching from a stage to its succeeding one.

Packets that are destined to full buffers can be handled by dropping those packets [18] or by applying backpressure mechanism [19]. The backpressure mechanism keeps packets in the preceeding stage until the required buffer becomes available again. That means that no packets are lost within the network in contrast to dropping packets. Local and global backpressure are distinguished. Local backpressure only observes the destination buffer of the next stage: The packet of stage k is sent if space at stage $k+1$ is available. Global backpressure gets additional information about packet flows: The packet of stage k is sent even if no space at stage $k+1$ is available but becomes available till the packet is received. Such a situation may arise if a packet leaves stage $k+1$ at the same clock cycle.

Besides MINs with the banyan property, multistage interconnection networks with redundant paths also exist. For instance the Beneš network can be established by combining two MINs with the banyan property: a standard one followed by its inverse one. Several paths are available to connect a particular input-output pair of the network: redundant paths exist. The Beneš network is known as non-blocking multistage interconnection network of lowest complexity.

Bidirectional MINs as in Figure 2 apply bidirectional links in contrast to previously presented topologies. Network inputs and outputs are located at the left. Dependent on the input-output pair to connect, a path is set up by forwarding packets to the right for a sufficient number of stages. Then, the packet turns and moves back to the left until the desired output is reached.

Up to now, only regular network structures were presented. But irregular structures may also occur in network topologies. For instance, Figure 3 shows such an irregular network. One of the most irregular networks is the Internet.

All kinds and variations of previously presented net-

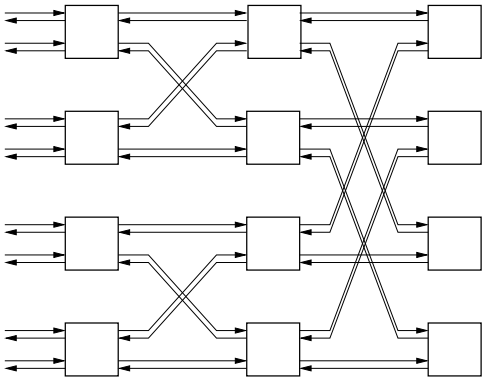

Figure 2: Bidirectional MIN

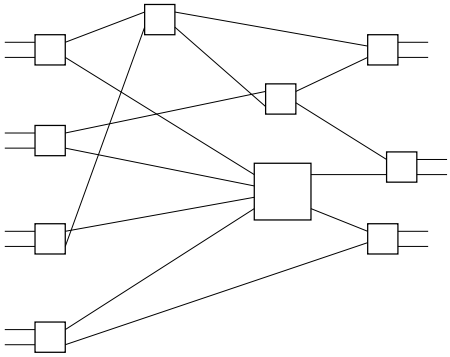

Figure 3: Irregular Network

works can be modeled by the simulator that is presented in this paper, including networks with redundant paths for input-output pairs.

3 SIMULATOR *CINSIM*

The simulator *CINSim* is a tool for simulation of component-based interconnection networks. It is designed to provide a single simulator for different kinds of network architectures that base on atomic components such as switches and buffers. A particular regular topology as in a former approach (a simulator called *MINSimulate* [3]) is not required.

The *CINSim* tool consist of two parts: a simulator core performing the simulation runs and a simulator graphical user interface (GUI) to design and draw the networks under investigation (see Figure 4).

The simulator core contains the implementation of network components and their behavior. Further on, a coroutine is included for simulating the interconnection networks in question considering desired switching and routing algorithms. Any regular or irregular, non-cyclic network can be modeled if based on the following components or a subset of them:

- **Switches (Routers)** are components to realize dynamically changing connections between switch inputs and outputs. Inputs and outputs are connected according to the requested network output of the message. Thus, they perform some routing and may also be called routers. If multiple inputs contain messages destined to the same output, a scheduling algorithm chooses one of the messages. Currently, random choice, round-robin, least recently used, most recently used, least frequently used, and most frequently used are implemented.

- **Buffers** store packets (if packet switching is applied). Shared buffers connected to multiple switch inputs/outputs are implemented as well as non-shared (single-queued) buffers.

- **Sources (Generators)** produce traffic which is offered to the network. Various destination traffic patterns and time-dependent traffic patterns can be generated, both combined with an arbitrary offered load. The traffic generators are driven by a random number generator. Up to now, just packet generators are implemented but message generators for circuit switching are also planed.

- **Destinations (target buffers)** represent the outputs of the network. They are in charge to remove the messages from the outputs as soon as they arrive.

- **Routes (Links)** connect previously mentioned components to form a network.

Additionally to these components, *CINSim* also offers **analyzers** for performance measurement. Analyzers can be connected via observer lines to buffers, sources, or destinations to determine the source or destination throughput, the delay times, or the buffer queue sizes.

Due to the involved stochastic events, confidence levels and estimated precisions must be observed while simulation to achieve a given accuracy. For this reason, the presented project incorporated the toolkit *Akaroa* [20]. It observes the simulation by permanently collecting the measured performance results and calculating the accuracy. If the termination criteria are met, it stops the simulation. Steady-state simulation is supported as well as terminating simulation. Terminating simulation is used to investigate the transient behavior of the networks in question.

Further on, *Akaroa* offers a random number generator with very long cycles. It also supports distributed simulation to accelerate the simulation runs by starting multiple replications of the simulation in parallel on connected computers.

The simulator core of *CINSim* is written in C++. Thus, its quite easy to implement new components, their behavior, or (parts of) the coroutine by using the inheriting techniques provided by that language. The coroutine consists of separate parts (modules) representing the switching techniques, routing algorithms, and scheduling algorithms.

As far as possible, simulation features have been encapsulated into classes. Factories are used to create instances of classes by their name. Thus, classes can be created outside of *CINSim* although their specification is not known to the simulator.

Another modern concept of developing software is the extensive use of XML. A valid network description needs to be provided as an XML file to the simulator core. The file must follow the specification of a particular grammar,

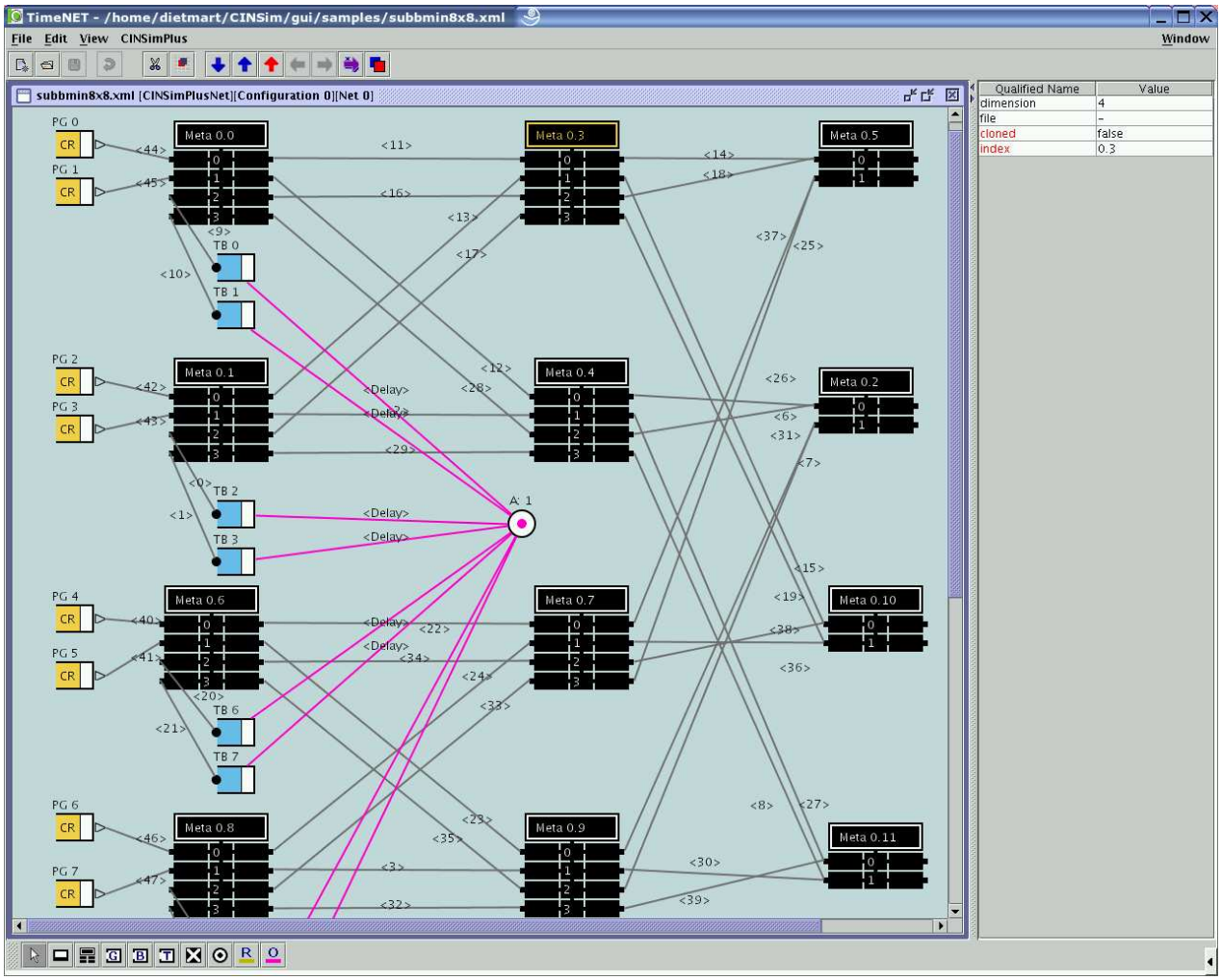

Figure 4: GUI of *CINSim*

given by a XML scheme. *CINSim* parses this file containing the XML network description to validate its correctness before simulation starts. Other programs such as the graphical user interface may apply the same grammar to generate appropriate network descriptions in XML language.

The GUI as shown in Figure 4 is written in the Java language. It provides a comfortable editor to draw the network that shall be investigated. The predefined components like buffers, switches, etc. can be added to the drawing area to construct the network. Copying parts of the current drawing is supported as well as creating meta components (black boxes) with underlying subnetworks. A meta component can again consist of meta components. Thus, a hierarchical drawing and modeling can be realized. For instance, Figure 4 shows the drawing of an 8×8 bidirectional MIN consisting of 2×2 bidirectional crossbars (realized by 4×4 unidirectional ones). The crossbars are represented by meta components of four inputs and four outputs. Clicking those meta components leads to a lower layer which gives their internal structure (Figure 5). In the given example, the meta component represents a 4×4 switch with a buffer at each input.

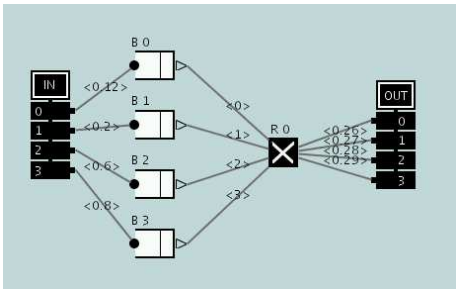

Figure 5: Internal Structure of a Meta Component (Fig. 4)

4 WORK IN PROGRESS: RECONFIGURATION

The development of this new simulator was driven by two goals. First, the simulator should be able to handle any kind of component-based interconnection network. Second, it should also be able to model the dynamic reconfiguration of such networks.

Concerning the first goal, the presented simulator *CINSim* fulfills this aim. Just some slight enhancements must

still be implemented. For instance, the restriction to non-cyclic networks will be removed in the near future. Some particular cyclic topologies can already be handled, like mesh networks, but a general approach must still be implemented. Further on, the switching technique of circuit switching is planned to be incorporated. Up to now, just packet switching can be applied (but with all its variants, e.g. store-and-forward switching and virtual cut-through switching). An additional feature that *CINSim* will provide soon is multicast. Multicast will be realized by packet replication while routing.

To incorporate the second goal, new concepts how to deal and model dynamic reconfiguration were required.

The implementation to allow the simulation of dynamically reconfigurable interconnection networks in *CINSim* is currently performed. This work is near completion. The GUI already supports depicting the reconfiguration process. Figure 6 shows the top layer of the hierarchical model. Each rectangular box represents an inter-

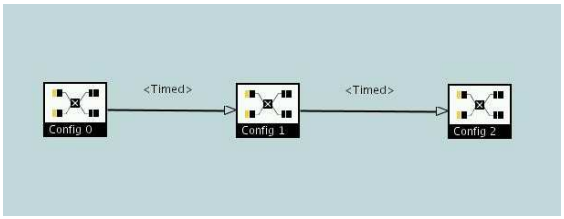

Figure 6: Reconfiguration in *CINSim* GUI

connection network (there are three networks in the given figure). The arrows between the rectangular boxes define the reconfiguration procedure in two ways: the reconfiguration sequence is given and the time parameters for reconfiguration are determined.

In the shown example, the left rectangular box contains the bidirectional MIN of Figure 4. That means clicking on this box leads to the representation as shown in Figure 4. The box in the middle and this one to the right constitute modifications of the bidirectional MIN. The arrows in between give the timeline: First, the left network will instantiated. Then, after some time passed by, the network is reconfigured to this one in the middle. Some time later, a further reconfiguration starts resulting in the right network.

Three time parameters define each reconfiguration. The first parameter represents the point in time when the current network stops accepting packets at the inputs. That allows the packets that are already somewhere in the network to leave it while no new packets enter it. The network runs empty before anything is changed. This leads to the easiest way of reconfiguration because it has not to be taken care of "old" packets (e.g. concerning their obsolete routing tag) from previous configurations. Future work will of course also consider non-empty networks to be reconfigured.

The second time parameter exactly represents this time needed to run the network empty. It can either be determined by calculating the worst case (longest time) of running empty or it can be set by the user. If the user chooses this time to low such that packets remain in the network, these packets are dropped before it is switched to the new configuration.

The third parameter gives the time that is needed to switch the hardware (for instance the FPGA) to the new network configuration. This time depends on the technology that is used.

The time parameters of a reconfiguration sequence are stored in the same XML file as the network descriptions participating in the reconfiguration. This file is loaded by the simulator core and manages the reconfiguration as well as it defines the network variations.

The simulator core already basically deals with such reconfigurations as previously described. But the reconfiguration of non-empty interconnection networks cannot yet be handled. Currently, our main efforts are directed to overcome this drawback. We expect a completion of the related implementation soon but exhaustive testings must also be performed.

An example result of the current implementation is depicted in Figure 7. It shows the mean delay times of pack-

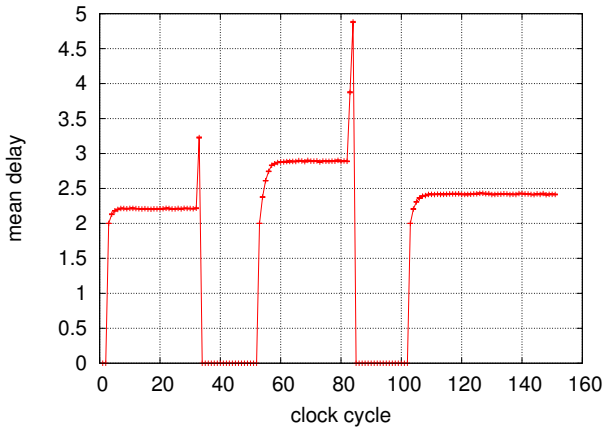

Figure 7: Reconfiguration Applied to Three Networks

ets leaving small multistage interconnection networks of size 4×4 consisting of 2×2 bidirectional crossbars. Store-and-forward switching is applied using buffers to store a single packet at each crossbar input. Three networks are loaded in sequence which forces two reconfigurations. The three networks just differ by their sources. In the first network, packet generators offering 30% load to the network are modeled, the second network uses packet generators producing 100% load, and the generators of the third network offer 50% load. The results are achieve by terminating simulation using a confidence level of 95% and an estimated precision of 2% for each clock cycle with non-zero mean delay time.

At clock cycle (time) 0, the first network configuration is started. Due to the two network stages, it takes three clock cycles till the first packets leave the network. Just packets that were not blocked leave the network at this moment. Thus, they were very fast. Packets leaving the network at a later time are a mixture of block and non-blocked packets: their mean delay time increases.

At clock cycle 30, the first reconfiguration starts and packet generators are stopped. Two clock cycles later, just blocked packets are left in the network leading to a peak in delay time. After the network ran empty (depicted by a delay time of zero), the second network con-

figuration is loaded at clock cycle 40. The loading is assumed to take 10 clock cycles. In consequence, the second network is started at clock cycle 50. Again, the network is filled with packets leading to increasing delay times clock cycle by clock cycle till a steady state is reached.

At clock cycle 80, the second reconfiguration starts. Ten clock cycles are offered to the network to run empty before the third network configuration is loaded between clock cycle 90 and 100. Then, the third network is started.

The results of Figure 7 can be used to investigate the particular reconfiguration behavior of the network sequence. Of course, this is only a small example but the reconfiguration of much more complex networks will yield a much more complex behavior which can now be studied using the simulator *CINSim*.

5 CONCLUSION

This paper presented the tool *CINSim*, a component-based interconnection network simulator for modeling dynamic reconfiguration. It is able to model and simulate all kinds of regular or irregular interconnection networks that are composed of switches, buffers, sources, and destinations. Links connect those components. Analyzers can be added for performance measurement. They give the kind of performance measure and the location where the measurement shall take place.

Steady-state simulation can be applied as well as terminating simulation to observe any transient behavior of the network. If stochastic events are involved, e.g. randomly generated traffic, confidence levels and estimated precisions are calculated to determine the accuracy of the results.

One of the main goals for developing this simulator is to model and evaluate dynamically reconfigurable interconnection networks. This implementation is not yet completed but all basic features concerning reconfiguration are already realized. First simulations of reconfigurable networks can be performed. To the best of the authors' knowledge that is the first simulator that is able to deal with dynamically reconfigurable interconnection networks. In its current version it is already used by three universities.

6 ACKNOWLEDGEMENTS

The authors would like to thank all further developers that are or were involved in this project. Particularly, Daniel Benecke, Marcus Brenner, Peter Bulisch, Hans-Christian Rahloff, and Li Zhou who maintained and implemented parts of *CINSim*. Without their work and ideas, this project would never have been as successful.

References

[1] Stunkel, C.; Shea, D.; Abali, B.; Atkins, M.; Bender, C.; Grice, D.; Hochschild, P.; Joseph, D.; Nathanson, B.; Swetz, R.; Stucke, R.; Tsao, M.; Varker, P. 1995, "The SP2 High-Performance Switch." IBM Systems Journal, 34, no. 2: 185–204.

[2] Soumiya, T.; Nakamichi, K.; Kakuma, S.; Hatano, T.; Hakata, A. 1999, "The Large Capacity ATM Backbone Switch "FETEX-150 ESP"." Computer Networks, 31, no. 6: 603–615.

[3] Tutsch, D.; Brenner, M. 2003, "MINSimulate – A Multistage Interconnection Network Simulator." In *17th European Simulation Multiconference: Foundations for Successful Modelling & Simulation (ESM'03); Nottingham*, SCS, 211–216.

[4] Tutsch, D.; Hommel, G. 2002, "Generating Systems of Equations for Performance Evaluation of Buffered Multistage Interconnection Networks." Journal of Parallel and Distributed Computing, 62, no. 2: 228–240.

[5] Yang, Y.; Wang, J. 2004, "A Class of Multistage Conference Switching Networks for Group Communication." IEEE Transactions on Parallel and Distributed Systems, 15, no. 3: 228–243.

[6] Zhou, B.; Atiquzzaman, M. 2002, "A Performance Comparison of Four Buffering Schemes for Multistage Interconnection Networks." International Journal of Parallel and Distributed Systems and Networks, 5, no. 1: 17–25.

[7] Jurczyk, M. 2001, "Performance Comparison of Wormhole-Routing Priority Switch Architectures." In *Proceedings International Conference on Parallel and Distributed Processing Techniques and Applications 2001 (PDPTA'01); Las Vegas*, 1834–1840.

[8] Turner, J.; Melen, R. 2003, "Multirate Clos Networks." IEEE Communications Magazine, 41, no. 10: 38–44.

[9] Wolf, T.; Turner, J. 2001, "Design Issues for High Performance Active Routers." IEEE Journal on Selected Areas of Communications, 19, no. 3: 404–409.

[10] Bondalapati, K.; Prasanna, V. 2002, "Reconfigurable Computing Systems." Proceedings of the IEEE, 90, no. 7: 1201–1217.

[11] Compton, K.; Hauck, S. 2002, "Configurable Computing: A Survey of Systems and Software." ACM Computing Surveys, 34, no. 2: 177–210.

[12] Lüdtke, D.; Tutsch, D.; Walter, A.; Hommel, G. 2005, "Improved Performance of Bidirectional Multistage Interconnection Networks by Reconfiguration." In *Proceedings of 2005 Design, Analysis, and Simulation of Distributed Systems (DASD 2005); San Diego*, SCS, 21–27.

[13] Becker, J. 2004, "Dynamically and Partially Reconfigurable Architectures." it – Information Technology, 46, no. 4: 218–225.

[14] Alderighi, M.; Casini, F.; D'Angelo, S.; Salvi, D.; Sechi, G. R. 2002, "A Fault-Tolerant FPGA-based Multi-Stage Interconnection Network for Space Applications." In *Proceedings of the First IEEE International Workshop o Electronic Design, Test and Applications (DELTA'02)*, 302–306.

[15] Tutsch, D.; Hommel, G. 2002, "Comparing Switch and Buffer Sizes of Multistage Interconnection Networks in Case of Multicast Traffic." In *Proceedings of the High Performance Computing Symposium 2002 (HPC 2002); San Diego*, SCS, 300–305.

[16] Tutsch, D.; Hendler, M.; Hommel, G. 2001, "Multicast Performance of Multistage Interconnection Networks with Shared Buffering." In *Proceedings of the IEEE International Conference on Networking (ICN 2001); Colmar*, IEEE, 478–487.

[17] Tutsch, D. 2002, "Buffer Design in Delta Networks." In G. Hommel; S. Huanye, eds., *The Internet Challenge: Technology and Applications*, Kluwer Academic Publishers, 93–101.

[18] Yang, Y. 1999, "An Analytical Model for the Performance of Buffered Multicast Banyan Networks." Computer Communications, 22: 598–607.

[19] Gianatti, S.; Pattavina, A. 1994, "Performance Analysis of ATM Banyan Networks with Shared Queueing – Part I: Random Offered Traffic." IEEE/ACM Transactions on Networking, 2, no. 4: 398–410.

[20] Ewing, G.; Pawlikowski, K.; McNickle, D. 1999, "Akaroa2: Exploiting Network Computing by Distributing Stochastic Simulation." In *Proceedings of the European Simulation Multiconference (ESM'99); Warsaw*, SCS, 175–181.