Development and Operation of Networks-on-Chip

Claas Cornelius **Dirk Timmermann**
Institute of Applied Microelectronics and Computer Engineering
College of Computer Science and Electrical Engineering, University of Rostock
Rostock, Germany
{claas.cornelius, dirk.timmermann}@uni-rostock.de

***Abstract* — Networks-on-chip have been suggested to cope with the issues caused by continuous scaling of transistor dimensions. To date, a large number of suggestions have been made for the different aspects of development and operation of large networks-on-chip, ranging from bit transmission to the abstract management of such systems. Modeling environments have also been presented to verify and evaluate the overall system performance. Their pros and cons are discussed to motivate the need for the developed simulator that is described in this paper. Two more sections cover the use of the simulator in a design flow for networks-on-chip and which investigations are intended to be done with it in the future.**

I. Introduction

Scaling of transistor dimensions has been the driving force of the semiconductor industry to achieve ever higher logic complexity and increased performance, at reduced costs per transistor. This development is about to continue and the International Technology Roadmap for Semiconductors (ITRS) projects by the end of this decade up to several billion transistors on a single chip with feature sizes below 50 nm, operating at frequencies in the 10 GHz range [1]. Thus, the story of success seems to continue but unfortunately, the scaling does not affect all properties likewise which causes new challenges and problems that need to be faced at miscellaneous design levels.

Leakage and subthreshold currents are dramatically increasing and will dominate the overall power consumption [2]. This does not just endanger the running time of mobile devices but also further increases the power density to unacceptable levels. Moreover, the wire delay, especially that of global wires in higher metal layers, does not scale well and will determine the system performance. Hence, in combination with large chip size and high frequencies, the implementation of synchronous designs will be very hard and demands additional area and power consumption. The clock generation and distribution will thereby limit the area of synchronous modules within integrated circuits to an acceptable size [3]. However, the arising problems are also noticeable at higher abstraction levels. One example is the memory-bottleneck which describes the fact that it takes up to several hundred clock cycles to access the memory while the computation stalls. Admittedly, this effect can be mitigated by deploying multi-level cache, run-ahead execution, or adaptive memory scheduling but at the price of extra area and power dissipation. There are further issues like reliability and parameter variations that also need to be considered thoroughly. Finally, the productivity for designing integrated circuits can not keep up with the rising logic complexity which is generally described as the design-productivity-gap. Recapitulating, advancements on all design and production levels or even new design methodologies are needed to overcome the ongoing and future expected issues of ultra deep submicron technologies.

Section II introduces Networks-On-Chip (NOCs) in principle and takes a look at concepts and existing test chips. Subsequently, the motivation for NOC simulators is described and several different implementations and their characteristics are presented. The chosen approach for a simulator is introduced in section III. Additionally, the integration of the simulator into a design flow and future intended investigations are explained before the paper is concluded in section IV.

II. Related Work

Networks-on-chip have been proposed as a promising option to overcome current issues of bus-based or point-to-point connected systems [4][5]. A NOC consists of miscellaneous independent resources that might work with different voltages, frequencies or even diverse technologies. The

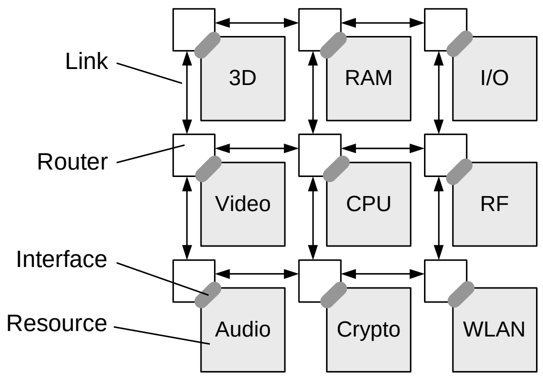

Figure 1. Example of a simple 3x3 network with regular mesh topology

resources are connected to a global network by an interface that encapsulates the communication- and computation-structures from another (see figure 1). The determined interface allows to exchange different modules or to incorporate resources from vendors without influencing other resources and their internal behavior. The network by itself is composed of links as well as routers and features services such as guarantees for bandwidth or latency that enable the resources to communicate with each other.

A multitude of ideas and solutions has been proposed for the lower layers and its functionality been demonstrated in such concepts as Nostrum and Æthereal [6][7]. For instance repeater, booster, and phase coding have been suggested for signal transmission, coding and error correction schemes have been analyzed as well as bufferless or asynchronous routers have been integrated in various topologies (e.g. mesh, torus, fat-tree) applying diverse routing schemes (e.g. dimension-ordered, adaptive, deflective).

As a result, first prototypes have been developed and their functionality has successfully been proven. Mullins et al. [8] presented a test chip with a 4x4 mesh topology using speculative routers (with about 5 million transistors, produced in a 180 nm technology). However, the implemented resources are only simple traffic generators that can select the destination as well as the packet length randomly or deterministically. Furthermore, the Korean Advanced Institute of Science and Technology (KAIST) has developed several types of test chips aiming at multimedia applications [9][10]. The chips use star topologies and a limited number of heterogeneous resources like RISC processors, SRAM, on-chip FPGA, and I/O. Another approach is targeted by the RAW microprocessor [11] which uses 16 resources in a regular topology whereas each of the resources has its own computation unit und local memory for data and instructions. The Raw microprocessor exploits its structure to effectively run streaming and parallelized applications.

Such test chips have pointed out the functionality of NOCs in principle, its connected benefits, and existing challenges. However, test chips are limited (e.g. by the number of resources or functionality) as well as inflexible and can not substitute the use of simulators to cover aspects of NOCs during investigation and development. Hence, the additional use of simulators is required to model the behavior of NOCs in terms of performance, power, area, thermal hot spots, and further network related metrics like bandwidth usage or areas of increased packet congestion. The motivation for specific NOC simulators is manifold because they

- are not limited to current technologies; simulators allow for instance the investigation of issues connected to future large NOCs with dynamically reconfigurable hardware or with several hundred resources, as predicted by [12] and others
- allow the developer to investigate and verify the influence of different hardware implementations as well as various software approaches; this includes for instance different links, routers, routing schemes, and topologies as well as load balancing or application mapping
- enable vendors and customers to evaluate different existing chips for their specific applications or domains
- can potentially avoid the time- and cost-intensive implementation of test chips by exploring design space and exposing the do’s and don’ts

The need for simulators and emulators has been recognized by the research community to explore design space and a large number of proposals has been presented in the last years. Zeferino et al. [13] have ported a parameterized VHDL model to an FPGA to evaluate the behavior of different router implementations based on HDL simulations. However, such an approach makes the monitoring of specific NOC features difficult or nearly impossible (e.g. link usage, thermal hot spots). To reduce the costs of synthesizable HDL development, high-level VHDL as well as SystemC simulations have also been proposed [14]. Those solutions are limited to a rather small number of resources or require large computational effort. To further raise the abstraction level and to increase simulation speed, simulators using C or C++ were proposed. Wiklund et al. [15] have presented such a simulator that is cycle-accurate and event-driven. Finally, an emulation framework for NOCs was developed by Genko et al. [16]. The implementation on an FPGA allows very fast emulation speed and monitoring of network related metrics. Nonetheless, the number of resources is obviously limited by the FPGA and by the overhead for the emulation environment.

III. Approach

This chapter describes the approach to implement a NOC simulator, how such a simulator can be integrated into a design flow, and what kind of open issues can possibly be investigated with it.

A. *The Simulator*

To assess the characteristics and the performance of NOCs is more complex compared to bus-based on-chip systems. That is due to the decentralized communication structure, in contrast to a shared medium, and the large number of resources that NOCs aim for. Therefore, new and NOC specific tools for the assessment need to be developed that suit the changed requirements. Simulation was favored over analysis for the assessment of NOCs due to the difficulties connected with accurate system analysis. Besides, emulation on a hardware platform was also considered but abandoned because it is limited to current technology (e.g. device behavior or chip size).

The NOC simulator has to bring together the advantages of gate level and network simulators. That is, reproduction of the functionality and characterization of the given setup by means of performance, power, area, thermal hot spots and further network related metrics like bandwidth usage or areas of increased packet congestion. The well-known and widespread programming language C++ was chosen for the implementation because it offers short execution time and eases the extension with existing libraries for graphical user interfaces (GUI) or plotting of results. Figure 2 depicts the simulator with its input and output data. The

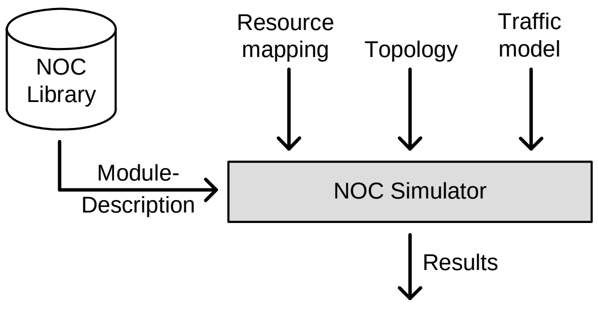

Figure 2. The NOC simulator and its input and output data

topology description includes the used types of links and routers and how they are connected to each other. This definition also implicitly contains the routing scheme that is supported by the routers. The different types of resources, its interfaces, and their positions are given by the input called resource mapping to specify the complete design of the chip. The third input is the traffic model which defines how often the resources communicate, to which destinations packets are sent, and what kind of message size is used.

The NOC simulator sets up the given scenario with the use of predefined modules from the NOC library. Thereby, all modules are specified in terms of behavior as well as delay, power consumption, area, and further necessary parameters. For instance, such a module could be a router with five ports supporting dimension-ordered routing with a bus width of 64 bits. The links and interfaces are defined similarly. However, the resources possess additional parameters that relate to the network behavior, like packet injection rate. These parameters are not predefined and will be given by the traffic model. For instance, if a resource is an audio decoder it will send packets mostly to only one destination requiring dedicated bandwidth. On the other hand, a resource administrating the whole chip will rather send short control messages to varying destinations requiring minimal delay. The classification of the resources (e.g. streaming or control) enables the preliminary evaluation of a system for a specific application domain without knowing the exact application.

The simulator is event-driven and the simulation interval can be set by the user to trade off simulation time for accuracy. For example, let us assume the accuracy is set to 50 ps, the delay of the link is 250 ps, and a transmission of a packet via a link is initiated at time instance t_0. Then, an event will be generated after $t_0 + 250$ ps and the packet will be passed to the appropriate connected router. From this it follows that the delay for a complete packet transmission between two resources is the sum of all involved modules. Thus, the simulation is performed on the level of modules and not on gate or even transistor level which shortens simulation time significantly.

Each module possesses counters which are increased every time the module is activated. These counters represent different parameters like activity or power consumption. This allows amongst others the evaluation of power dissipation, thermal distribution as well as network related metrics like link usage and areas of packet congestion. Those results are periodically logged into a file so that the state of the network can be examined at all these

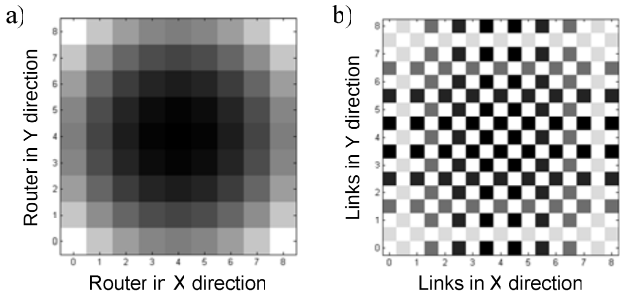

Figure 3. a) Router and b) link usage in a 9x9 mesh network with dimension-ordered routing

time instances. It is planned to extend the simulator with a graphical interface to ease the examination of results. A possible graphical representation is depicted in figure 3. The exemplary results for the routers and links were obtained for a regular 9x9 mesh network with dimension-ordered routing. The packet injection rates were equal for all resources and the destinations were chosen randomly for a large number of packets. It can be seen that dimension-ordered routing concentrates the activity in the center of the network which is denoted by the darker areas.

As reliability issues intensify, sporadic errors will be unavoidable and malfunction of resources will have to be dealt with. Thus, it is intended to add another property to the modules to generate such failures randomly or at deterministic time instances to be able to observe the influence on the system. Another aspect is the study of reliability connected with parameter variations (e.g. technology, temperature, or supply voltage). Such investigations are possible with the simulator in principle but it requires describing the modules in the NOC library for all these cases that are to be investigated. Hence, this is not supported yet.

B. Integration in a Design Flow

Against the background of missing design-productivity and very large complex circuits, the usage of an automated design flow is absolutely mandatory. For this reason, a design flow is considered that integrates all design steps, from the architecture concept till physical layout, and comprises the NOC simulator to simplify the implementation of networks-on-chip.

A possible design flow is shown in figure 4 whereas iterative loops between design steps are omitted for simplification. The entry point is an application specification which can be an abstract description of system requirements in UML or a tangible implementation in C++. The originated tasks need to be partitioned into hardware and software components depending on the given constraints. The HW/SW partitioning with respect to NOCs corresponds to specific resources (also called intellectual properties, IPs, or accelerators) and source code that has to be executed on processors (i.e. general purpose resources). The NOC library provides the simulation models as the foundation to be able to evaluate the trade offs between efficient hardware and flexible but slow software implementation of the tasks. At this stage, the traffic generated by the resources is already determined. This enables the next step to configure and optimize the complete chip. That is, links and routers are additionally chosen and connected to the given resources to meet the requirements. Thereto, different types of links, routing schemes, or resource positions can be used for the optimization.

Topology, traffic model, and resource mapping are fixed by now, or can alternatively taken from an existing chip, so that the NOC simulator can be used to evaluate and visualize the results of the given NOC as described in the previous section. If the results are satisfying and no further enhancements shall be applied, the NOC description based on the simulation models needs to be translated to a Register Transfer Level (RTL) description. This step, called NOC generation, requires that all simulation models of the NOC library exist additionally as implemented components. The further steps of such a flow, like synthesis,

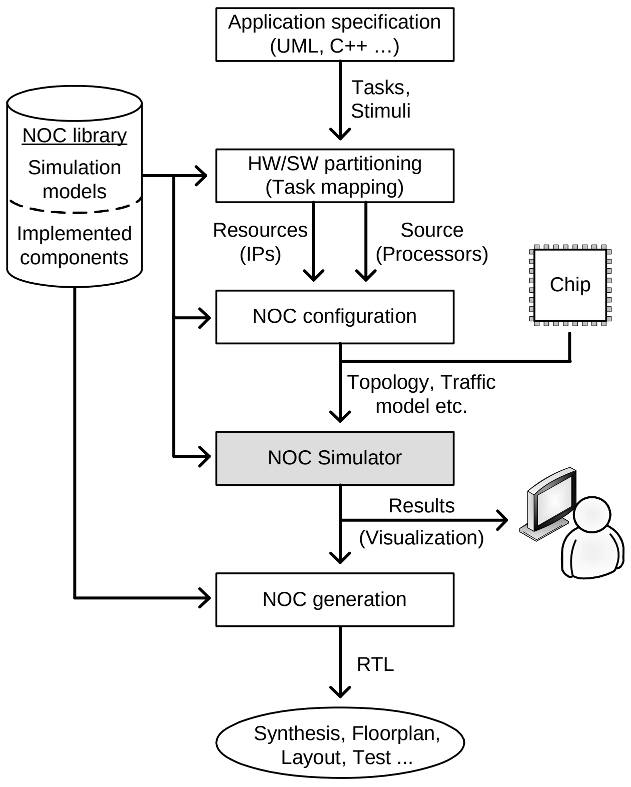

Figure 4. The NOC simulator as part of a design flow for networks-on-chip

placement, or routing, will have to incorporate and deal with issues of ultra deep submicron technologies that are not solely related to NOCs like reliability and signal integrity, yield optimization, as well as parameter variations.

C. Intended Investigations

However, the simulator is understood as a foundation to investigate open questions, focusing on large NOCs with more than hundred resources. For instance, an appropriate programming model is required to exploit the parallel structure of the resources and to deal with distributed memory. Subsequently, it needs to be considered how such a program can be mapped onto the different resources. This resource mapping can be part of the development but it also has to be considered during on-line operation and requires the attention of a wide range of aspects like communication-computation trade-offs, power issues, quality of service requirements, or thermal characteristics.

The management of a NOC, as part of an appropriate operating system is another factor that shall be investigated. Such management can be done in a central resource that controls the mentioned task mapping, but also controls load balancing, power issues, hardware reconfiguration, availability of resources and services, as well as more administrative tasks. However, such a central unit will restrict the performance with increasing network size due to the large number of required control messages so that a hierarchical organization or completely autonomous resources will probably be the better choice.

Two more things shall be mentioned that need to be examined. Firstly, testing and verification has to be performed at some stage due to the fact that sporadic as well as static errors will be inevitable. The detection of such errors will allow static adaptation of the chip’s behavior or even on-line self-healing and can increase the yield and reliability of large integrated circuits. Lastly, it calls for benchmarks to assess own solutions or to compare results within the research community. These benchmarks will have to be developed for a number of different use cases to be appropriate for the miscellaneous application domains.

IV. Conclusions

The implementation of a simulator for networks-on-chip has been presented in this paper. The aim was to achieve short simulation times to be able to accurately evaluate and investigate large chips with more than hundred resources. This was achieved by the use of C++ as programming language and by performing the simulations on module level. Additionally, the integration of the developed simulator into a design flow was discussed to efficiently design large complex systems. Lastly, future intended investigations were proposed that can now be simplified by the use of the developed simulator.

Acknowledgment

This work is supported by the German Research Foundation (DFG) under grant number GRK-466.

References

[1] Semiconductor Industry Association (SIA), “International Technology Roadmap for Semiconductors,” 2003.

[2] N. Kim et al., “Leakage current: Moore's law meets static power,” Computer, Vol. 36, No. 12, pp. 68-75, December 2003.

[3] D.Sylvester and K.Keutzer, “A global wiring paradigm for deep submicron design,” IEEE Transactions on CAD/ICAS, Vol. 19, No. 2, pp. 242-252, February 2000.

[4] W. Dally and B. Towles, “Route packets, not wires: On-chip interconnection networks,” Proc. of DAC, pp. 684-689, 2001.

[5] L. Benini and G. de Micheli, “Networks on chips: a new SoC paradigm,” IEEE Computer, Vol. 35, pp. 70-78, January 2002.

[6] S. Kumar et al., “A network on chip architecture and design methodology,” Proc. of ISVLSI, p. 147, April 2002.

[7] K. Goossens, P. Wielage, A. Peeters, J. van Meerbergen, “Networks on silicon: Combining best-effort and guaranteed services,” Proc. of DATE, pp. 423-427, 2002.

[8] R. Mullins, A. West, and S. Moore, “The design and implementation of a low-latency on-chip network,” Proc. of ASP-DAC, pp. 164-169, 2006.

[9] S. Lee et al., “An 800MHz star-connected on-chip network for application to systems on a chip,” Technical Digest of ISSCC, pp. 468-469, February 2003.

[10] K. Lee et al., “A 51mW 1.6GHz on-chip network for low-power heterogeneous SoC platform,” Technical Digest of ISSCC, pp. 152-153, February 2004.

[11] M. Taylor et al., “Evaluation of the Raw micro-processor: An exposed-wire-delay architecture for ILP and streams,” Proc. of ISCA, June 2004.

[12] A. Jantsch, “NoCs: A new contract between hardware and software,” Proc. of DSD, pp. 10-16, September 2003.

[13] C. Zeferino, M. Kreutz, and A. Susin, “RASoC: A router soft-core for networks-on-chip,” DATE, p. 30198, 2004.

[14] D. Tortosa and J. Nurmi, “VHDL-based simulation environment for PROTEO NoC", Proc. of HLDVT, October 2002.

[15] D. Wiklund, S. Sathe, and D. Liu, “Network on chip simulations for benchmarking,” Proc. of IWSOC, Banff, Canada, pp. 269-274, July 2004.

[16] N. Genko et al., “A complete network-on-chip emulation framework,” Proc. of IEEE DATE, pp. 246-251, 2005.