Evaluating NoC communication backbones with simulation

Rikard Thid, Mikael Millberg, and Axel Jantsch
Royal Institute of Technology (KTH), Electrum 229, SE-164 40 Kista, Sweden
{thid,micke,axel}@imit.kth.se*

Abstract

This paper describes a Network on Chip simulator that was developed to evaluate our NoC architecture Nostrum. It is shown how SystemC's features for communication refinement is used to make a highly flexible simulator. The simulator is reconfigurable so that it is possible to try different NoC platforms and different mappings of workloads. In addition to the modeling of our Nostrum architecture, a bus-based architecture is modeled as well, and the performance for a simple workload model is compared.

1. Introduction

In the SIA silicon roadmap[1], it is predicted that the increase in chip capacity will continue for at least another 8-10 years and it will be possible to integrate systems with billions of transistors on a single chip. Current System-on-Chip (SoC) methodologies do not offer the required amount of reusability to enable system designers to meet the ever increasing time-to-market constraints. The desired future SoC methodology should enable, not only, reuse of traditional IP-cores but also communication infrastructure. Current bus-based platforms suffer from limited scalability and poor performance for large systems. In order to overcome these problems several approaches for networks on chip [2, 3, 4, 5, 6, 7] have been proposed. They allow reuse of the communication infrastructure among many products thus reducing the design and test effort and the time to market. In order to enhance reusability and to ease programmability, most NoC proposals recommend standardized and layered communication protocols for communication among cores.

The performance of busses versus NoC is mathematically analysed in [8]. A problem that arise when simulating NoC platforms is how to co-simulate the network with the rest of the chip. Our solution is based on the channel based communication model that is used in SystemC. This paper provides an simulation based comparison of busses and Nostrum to demonstrate the possibilities with the flexible NoC simulator.

The rest of this paper is organized as follows. Section 2 presents our NoC platform named Nostrum. In Section 3, the NoC simulator is presented. The modeling of workload models is described in Section 4. Section 5 explains and analyses the experiments that we perform. The last Section concludes the paper.

*This work is a part of the joint Finnish-Swedish EXSITE research program. This work was sponsored by TEKES, VINNOVA, Nokia Oyj, Ericsson Radio Systems AB, and Spirea AB Kista.

2. Nostrum

The overall purpose with a NoC platform is to act as host for a system that performs one or several tasks with hardware components. In the *Nostrum* architecture, the system is mapped to a set of *Resources*. A Resource is in this context a microprocessor, a memory, a FPGA, a digital signal processor, or an I/O - resource. An I/O-resource is a device that is connected to the chip's pins for the purpose of external communication. The Resources are physically organized in a two-dimensional mesh structure as depicted in Figure 1.

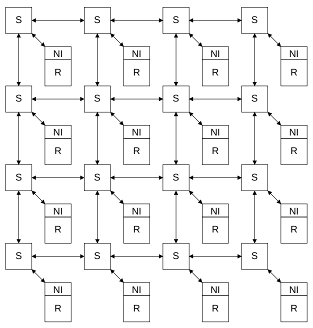

Figure 1. A Nostrum NoC with 16 Resources and switches.

All Resources are equipped with a *Network Interface*, which connects to the network in order to provide services for Resource-to-Resource communication.

The Switches route packets through the network using a hot-potato routing algorithm [9], which reduces the need for buffers within the switches. This is an attractive property for us since we want the area overhead to be as small as possible. Because of the ever increasing wire delays, it is desirable that information travels as short distances as possible. Therefore, Nostrum does not use a centralized router/arbiter such as many bus-based architectures. Instead all the routing decisions are made locally in the switches.

The Nostrum architecture is designed in a layered fashion: this allows us to partition functionality onto different layers, inspired by the OSI reference model. Notice that this does not necessarily mean that different layers are dealt with by different pieces of hardware or software. Instead we can merge functionality from different layers into the same piece of HW/SW. An *Entity* is the unit that implements the functionality of a layer. An example of an entity is a switch, which performs *Network layer* functionality. The way that the entities are interconnected is called *topology*.

A two dimensional mesh topology is used since it is mappable to two dimensions. This is due to the physical constraints of a chip, which does not allow more general topologies such as high-dimensional hypercubes.

3. Simulation environment

3.1. SystemC

In order to evaluate our Nostrum architecture, we have developed a SystemC based simulator. SystemC [10] is a superset of C++ targeted at simulating whole systems with both hardware and software components. In this paper, we will only deal with SystemC's properties as a simulation language.

Models in SystemC basically consist of *modules* whose behaviour is defined in C++. Each module has any number of *ports* that it uses to interact with other modules.

In order to cope with the increasing complexity of communication, SystemC (from version 2.0 and forward) has the ability to organize the communication into *channels*. Channels have interfaces that the modules use to communicate through. An example of ordinary channel is dedicated wires as depicted in Figure 2a. Hierarchical channels can connect to any number of modules and implement several other, non-hierarchical channels. A hierarchic channel could be for instance a bus (as in Figure 2b) or a NoC infrastructure.

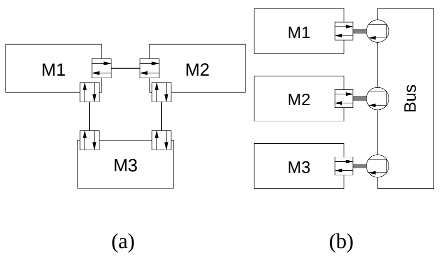

Figure 2. Three modules that are interconnected by (a) several primitive channels or (b) one hierarchical channel.

3.2. The simulator

The simulator is divided into an *Application Domain* and a *Communication Domain* as depicted in Figure 3. The system is distributed into Resources that are modeled in SystemC by *Resource models*. The purpose of them is to generate traffic so that the behaviour of the network for a given workload can be studied. These models interact by sending and receiving messages over the communication platform. The placement of the Resources is managed by the *Resource mapper.* A designer can easily change the mapping of Resources since all mapping is done in the Resource mapper, and no other part of the simulator is directly affected by the mapping.

The communication domain consists of models of entities that implement various layers. Four layers are represented in the simulation environment, namely the Transport(TL), Network(NL), Data link(LL), and the Physical layer(PL).

A topology generator is used to instantiate and connect entities with each other. The simulator has one topology generator that creates Nostrum models of arbitrary size. A small 2x2 example is depicted in Figure 4. There is also a bus topology generator that uses the same TL, LL, and PL entities as the Nostrum generator. However, while the Nostrum NL entity models a hot-potato routing algorithm, the bus NL entity uses a round-robin arbitration scheme.

In order to interface with SystemC models of Resources in a natural way, our simulator is a hierarchic channel. Any SystemC modules that will use NoC for communication does so using an *interface*. This interface features, the opening of channels, blocking, and non-blocking send and receive primitives. No knowledge of the platform is necessary when writing the Resource models, since all communication is handled through interfaces. This enables a user to change the network model but still use the same workload models.

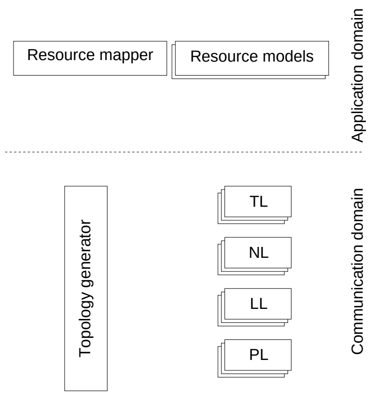

Figure 3. The main components of the simulation environment.

4. Workload models

In order to study our Nostrum architecture, we designed a simple workload model using SystemC. As the communication platform interface is very small, it is easy to write and integrate a simple workload model to the

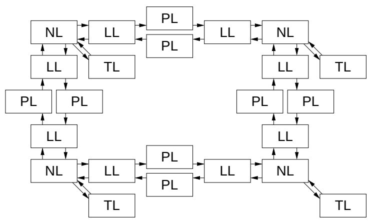

Figure 4. Entity interconnection in a 2x2 mesh topology.

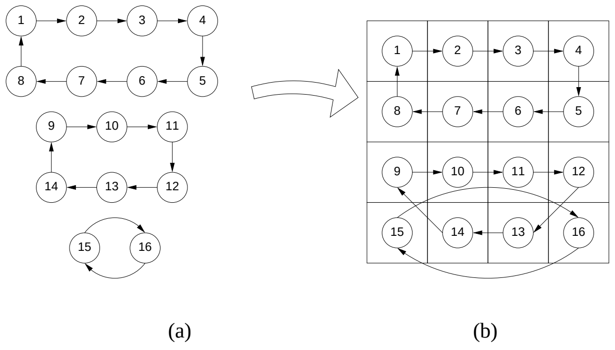

Figure 5. Workload model and its mapping on a 4x4 mesh.

simulator. Our model consists of 16 identical Resources that are logically interconnected in three rings as depicted in Figure 5a. The Resources were placed so that the number of hops needed to send packets is reasonably low (but not optimal). The placement is depicted in Figure 5b. The Resources repeatedly sends a packet, then wait a random number [1] of clock cycles before sending again. This kind of behaviour is what a pipelined signal processing application could look like. In average each Resource sends a message every ninth clock cycle. With a Resource clock frequency of 1 Ghz, approximately 111 million messages/second will be sent through the network interface. As the network interface need only one clock cycle per message, it may be possible to run the communication network on a lower clock frequency than the Resources, possibly saving power. Thanks to the flexible SystemC simulation engine it is easy to scale the frequency of the network.

5. Experiments

The purpose of the experiments is to determine how efficiently our Nostrum architecture can perform given the workload model previously discussed. It is also interesting to see how the Nostrum architecture performs in relation to a bus-based architecture. What are the differences in latency and what clock-frequency is the lowest that can be used without data loss due to low throughput?

[1]Normal distribution, mean value 9, standard deviation 2.

5.1. Clock frequency scaling

In order to minimize the power consumption on chips, the clock frequency is generally kept as low as possible. Most of the power in CMOS is consumed during the switching of logic gates. This is called the dynamic power consumption and it is expressed with the following formula:

$$P_{dyn} = C_L \cdot V_{DD}^2 \cdot f \cdot \alpha \qquad (1)$$

The power consumption (P_{dyn}) depends on the capacitive load (C_L), the supply voltage (V_{DD}), clock frequency (f), and switching probability (α). Since an increase in clock frequency requires the supply voltage to be higher, we wish to run the clock as slow as possible. In a network, we should not drop the frequency to low since the performance will be lower and packets may be dropped. The lowest possible clock frequencies for the two discussed architectures for a specific workload are investigated .

5.2. Method

The simulator was configured to run the Nostrum and the bus-based architectures with the same workload models. The TL-entities, which are the highest entities in the protocol stack and therefore experience the most latency, measure the average and the maximum time between sending and receiving of data. The entities are configured to report any loss of data that occurs when more data is put into the system than it can handle. Multiple simulations were run with different clock frequencies for the respective communication infrastructures ranging from 125 MHz to 4 GHz. The Resource models were always running at 1 GHz and therefore the same amount of data was attempted to be sent. The average and the maximum latencies were recorded for each simulation. The lowest possible frequency for each platform was noted.

The PL-entities, which represent the links between wires measure how frequently data is transmitted over them. This information represents α in Formula 1, and together with the clock frequency the power can be calculated.

5.3. Result

The results from the simulations are plotted in Figure 6 and 7. For the simulations with too low clock frequencies, the latencies are very high and they are not included in the plots.

For the bus architecture, a bus clock faster than 1.8 GHz is required, and the Nostrum only needs 200 MHz to handle all data. It is natural that the latency decreases with longer clock periods. However, as shown in the right figures in both cases the latency is not fixed in terms of clock cycles. In the bus-based architecture the decreased efficiency is explained with that packets may have to wait until they become routed since only one packet is routed each clock cycle. In the Nostrum architecture it is possible to route four packets in each switch simultaneously.

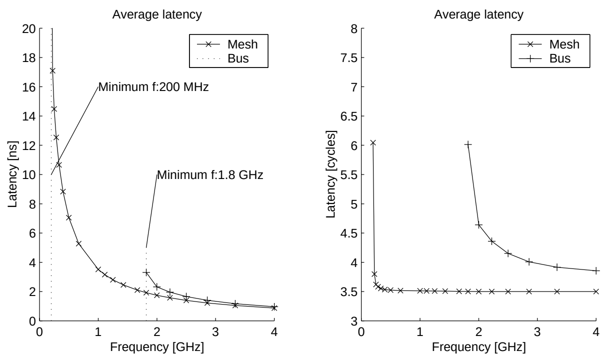

Figure 6. Average latency.

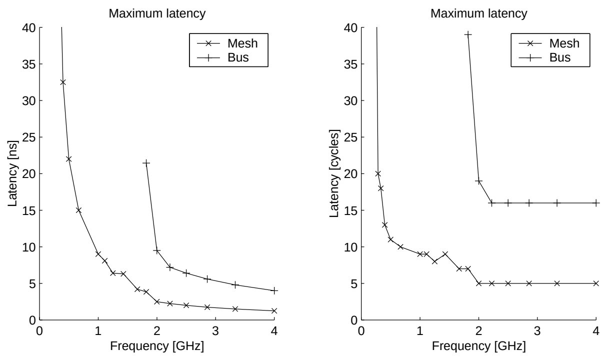

Figure 7. Maximum latency.

The decrease in efficiency is explained by the congestion that occurs when many wires are occupied. Figure 8 shows how occupied the links between switches are, and how this affects the link power consumption. The power figure is calculated with Formula 1, with V_{DD} and C_L constant.

6. Conclusions

It was demonstrated that NoC simulation can be done in a flexible manner thanks to the channel based communication model in SystemC. Two different communication platforms were modeled with the same workload model and it was shown that our *Nostrum* platform can operate at a much lower clock frequency than a shared bus platform. Topics that are interesting for future works include to study the impact of more workload models and comparison with other NoC platforms than Nostrum.

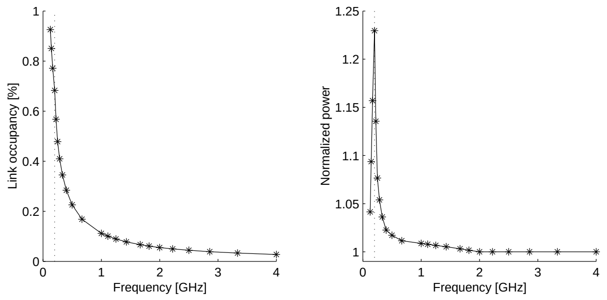

Figure 8. Link occupancy and power in Nostrum wires.

References

[1] "The international technology roadmap for semiconductors," International SEMATECH:Austin, TX., 2001.

[2] P. Guerrier and A. Greiner, "A generic architecture for on-chip packet-switched interconnections," in *In Proceedings of Design, Automatation and test in Europe*, pp. 250–256, 2000.

[3] G. D. M. Luca Benini, "Powering networks on chips," in *Proceedings of the international symposium on Systems synthesis*, pp. 33–37, 2001.

[4] L. Benini and G. D. Micheli, "Networks on chips: A new SoC paradigm," *IEEE Computer*, pp. 71–78, January 2002.

[5] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Öberg, K. Tiensyrjä, and A. Hemani, "A network on chip architecture and design methodology," in *Proceedings of IEEE Computer Society Annual Symposium on VLSI*, April 2002.

[6] K. Goossens, J. van Meerbergen, and P. Peeters, A.and Wielage, "Networks on silicon: combining best-effort and guaranteed services," in *Design, Automation and Test in Europe Conference and Exhibition, Proceedings*, 2002.

[7] W. J. Dally and B. Towels, "Route packets, not wires: On-chip interconnection networks," in *38th Design and Automization Conference*, 2001.

[8] C. a. Zeferino, M. E. Creutz, L. Carro, and A. a: Susin, "A study on communication issues for systems-on-chip," in *Proceedings of Integrated Circuits and Systems Design (SBCCI)*, 2002.

[9] U. Feige and P. Raghavan, "Exact analysis of hot-potato routing," in *Foundations of Computer Science, Proceedings*, pp. 553 – 562, IEEE, 1992.

[10] T. Grötker, S. Liao, G. Martin, and S. Swan, *System Design with SystemC*. Kluwer Academic Publishers, 2002.