Network on chip simulations for benchmarking

Daniel Wiklund, Sumant Sathe, and Dake Liu

Dept. of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden
{danwi,sumant,dake}@isy.liu.se

Abstract

Networks are becoming increasingly popular for use as on-chip interconnects. The problems with specification and performance evaluation increase with these solutions compared to the traditional interconnect. This paper describes the design and simulation environment developed in the SoCBUS network-on-chip project. This environment is used as a basis to develop the benchmarking procedures necessary to assess the performance of the networks. Two benchmarking examples are presented and used for evaluation of the SoCBUS network. These examples show how the simulation environment can be used to find the load bottleneck. They also show the appropriateness of the SoCBUS solution for (hard) real-time systems.

1. Introduction

A core part of almost all electronic designs is the internal communication infrastructure. Today this is mostly constructed as a shared bus or by using dedicated point-to-point links. The buses use time-division multiplexing (TDM) to allow multiple communications to share the media. This is a very simple and useful solution that is applicable for all designs with reasonable demands on the communication performance in terms of bandwidth, latency, flexibility, quality of service (QoS), etc. The other extreme is point-to-point links where the situation is opposite. All flexibility is gone but the physical performance is much higher.

The interconnection problem can be subdivided into many problems. Four of the most important physical problems that must be addressed by the interconnect structure are:

- Port format adaptation - Different intellectual property (IP) cores tend to adhere to different interconnection standards.
- Speed difference between IP cores.
- Connectivity - A set of IP cores must be able to communicate with each other in a way that often cannot be fully predicted at the chip design time.
- Bandwidth of communication channels - Both the actual bandwidth of a single transmission is interesting as well as the number of possible concurrent transmissions.

The concept of using networks as on chip interconnect has reached a broad acceptance in both the academic society as well as industry today [1], [2]. The SoCBUS project at Linköping University was started in 1999 and the result so far is a complete proposal for a network on chip. Behavioral models for the network components have been extracted and implemented. These models have been used to investigate the behavior of the SoCBUS network. Different SoCBUS routers have been implemented at the register transfer level (RTL). Our achievements show that all four subproblems described above have been addressed as presented in earlier papers [3], [4].

We have realized the importance of benchmarking to find the bottlenecks in networks on chip. In this paper we present an outline of the benchmarking process and two examples of benchmarks. These benchmarks are used to show the validity of the real-time application domain for the SoCBUS project. Sections 2 and 3 give a brief introduction to networks on chip and the SoCBUS project. Section 4 introduces the design environment and simulation flow for SoCBUS. Definitions of benchmarks and bottlenecks are introduced in section 5. Sections 6 and 7 defines the example benchmarks and summarizes the simulation results. Finally, in section 8, we draw some conclusions and outline future work in this area.

2. Networks on chip

Lately there has been a move in usage of networks from communication between systems to communication within systems. Extending this trend further there is an obvious possibility to use networks on chip. But the demands on network design change considerably when moving the design towards lower levels, shorter distances, and physically smaller implementations. One of the main issues is the physical size of the network components used. No longer can a network interface or a router occupy an entire board or chip. Rather it has to fit in a space considerably smaller than a simple on-chip processor system of a few hundred thousand gates. Preferably, the silicon cost for the connection network should be less than 10% of the silicon cost for the (custom) functions.

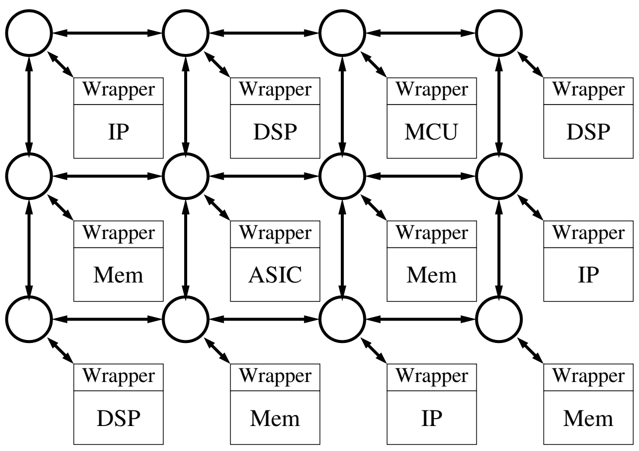

Figure 1. Overview of a SoCBUS network

This leads to restrictions on size of the components - especially memories - used in the network. Because of the rather stringent demands on size the network components must be kept reasonably simple. This in turn puts restrictions on such things as network protocols, signaling complexity, and buffering of data in the switches.

3. SoCBUS network

The network considered in this paper is the SoCBUS project at Linköping University. This system uses a hybrid packet-circuit switching method known as packet connected circuit (PCC) [5]. This approach uses a small routing packet that traverses the network and sets up a circuit switched route for the bulk payload data to follow. In general, any network consumes a lot of memory for data buffering. These data buffers also give longer transport latency which is not acceptable on a chip. By using the PCC concept, both the need for buffers and thus the latency penalty are eliminated. The drawback is that a failed routing will cause a retry which may lead to increased congestion.

The primary network components are the routers and the wrappers. These are connected to each other according to the topology of the network. The topology can be arbitrary but we propose the use of a two-dimensional mesh as an initial topology for synthesis purposes, see Fig. 1. There are several reasons to propose a mesh topology but the most obvious is that is maps well onto the flat surface of a chip with short wires and simple routing. This mesh can then be optimized by adding and removing routers and links according to the demands put on communication by the application. Of course, other initial topologies, e.g. fat trees, are also possible as a starting point in the design.

3.1. SoCBUS routers

The routers are the core component within the network. The PCC approach results in simple routers in the network with no need to buffer the data in the router except for retiming purposes. This allows the routers to be kept at a minimum in size while maintaining high throughput.

The current implementation of a router shows a data latency of one clock cycle at 16 bits data width and a clock rate of 1.2 GHz in a 0.18 um technology. The latency experienced during route setup is 6 cycles per router [6], [7]. The only buffering done in the routers is one word per input port used for retiming purposes.

3.2. SoCBUS wrappers

The wrappers are the interfaces that connect the local IP block(s) to the network. The purpose of these wrappers is to allow a specific network format without putting (too severe) constraints on the interfaces allowed to the rest of the design. It is not possible to create a generic wrapper that will take any I/O format and transform this into the network format. There will rather be a set of wrappers for different style connections on the IP side.

The most efficient use of the network is probably message passing but the transition from the current design style will require that transaction style (e.g. read, write, cache line fetch, etc.) must be supported by the wrappers (and the network).

4. Design environment

The main design difference from designs based on traditional interconnect structures (e.g. buses) is that the complexity in assessing the overall performance of a network on chip is significantly higher. The design environment must be able to support such a performance assessment through analysis or simulation. Simulation has been chosen as the primary method in the SoCBUS project because of the difficulties involved in accurate analysis of a system.

The design environment consists of design libraries and simulation support. The design libraries contain component implementations and simulation models. The simulation models are used together with a simulator. This simulator takes descriptions of the traffic and network and does a performance evaluation for this setup.

The typical flow for a network on chip design, see Fig. 2, starts with an application or an application domain. From this it is possible to extract the communication patterns and demands that should be put on the network. This is done in a similar fashion as would be done for a classical TDM based interconnect structure. Generally, this step must be done with great care to make sure that the extracted traffic model matches the actual communication behavior in the application.

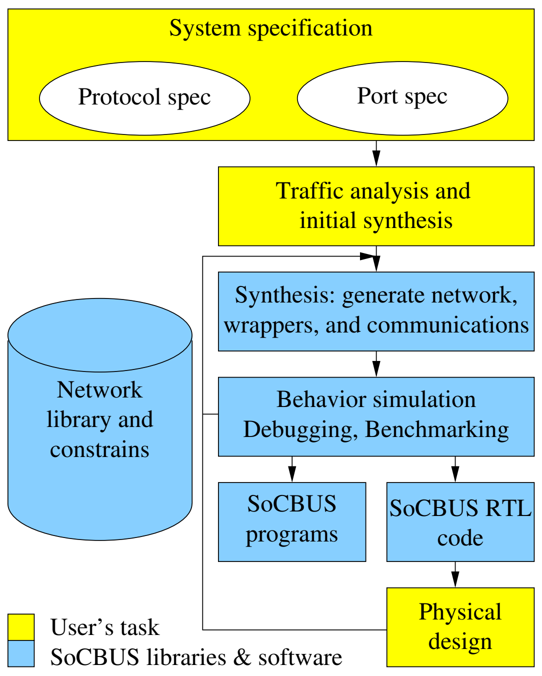
System specification
Protocol spec
Port spec
Traffic analysis and initial synthesis
Synthesis: generate network, wrappers, and communications
Network library and constrains
Behavior simulation Debugging, Benchmarking
SoCBUS programs
SoCBUS RTL code
Physical design
User's task
SoCBUS libraries & software

Figure 2. Design flow for network on chip based systems

After the analysis of the application an initial network can be synthesized. This network is used in simulations to assess the performance of the network solution under the considered traffic model.

It is rather obvious from the above that simulations are a very important step in the design flow. The simulations are not only needed to achieve good system performance at low cost but also to guarantee success at all [5], [8].

4.1. Details on the simulation flow

The simulations for the SoCBUS projects are done using an in-house simulator, developed by the authors, that is tailored to the task of simulating on-chip networks [4]. The simulator is used in conjunction with a in-house stimuli generator tool according to the simulation flow in Fig. 3. The inputs to the flow are the traffic and network models, represented in XML.

The traffic model is completely decoupled from the network topology. The mapping of traffic sources and destinations are done using literal names in the model files. Each named source/destination must be present as a named source/destination in the network model. If a source named "Src" has been specified in the traffic model there must also be a wrapper/IP block connection named "Src" in the network model. This allows the allocation of connections between wrappers and routers to be done entirely in the network model.

The generated stimuli uses absolute time stamps (i.e. it is

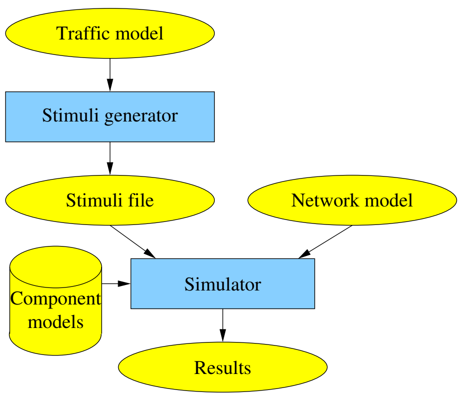

Figure 3. SoCBUS simulation flow

scheduled) so the information in the stimuli file is just the source, destination, start time, and size for each transfer.

The simulator is event based in a fashion similar to a typical VHDL simulator. All network components (routers, links, sources, and destinations) are implemented as compiled-in behavioral models written in the same programming language as the simulator. Every time a network component generates an event a message is created. These event messages are distributed to the proper destinations through a message server. The central message server handles all time ordering of events/messages through a message queue. This guarantees that the events will happen in the correct order independent of the order in which they have been created. The simulator is implemented in C++ to achieve high simulation speeds. Using C++ also allows extensions to be made to the tools in this commonly known and efficient programming language.

4.2. Simulation models of network components

All network related models that are used in the simulations are based on the implementations that have been made at the register transfer level in Verilog. The current implementation of the router model uses six cycles for a routing decision and one cycle for transferring data and acknowledgments. The links are modeled as a simple discrete time delay of zero (or more) cycles. The sources and destinations are currently also modeled in a rather simple fashion. The IP block and the wrapper are seen as a single entity that will generate and consume transfers according to the stimuli file.

Models are easily changed or added if new implementations of components are made or the need to model certain IP blocks in a more thorough fashion arises. Multiple models for a single component type are supported and are selected through model names in the network description file. A simple API is used to implement the models in C/C++.

5. Benchmarking and bottlenecks in on-chip networks

A benchmark is the combination of (traffic) specification and results (generally in terms of a performance estimate) that has been used and achieved in the process of benchmarking. The benchmarking process will have different goals, e.g. finding the effective bandwidth (i.e. accepted load), latency, optimal transfer sizes, etc. depending on the final application (domain) for the system.

The limiting factor in any application, whether it is computation, communication, or something else is commonly called a bottleneck. The aim of the benchmarking process is generally to find the bottlenecks in a system.

In a one dimension bus the fundamental bottleneck is the shared medium. This is a bottleneck can be described using basically a scalar value telling the total amount of data that can be transferred within a given time period. This value shows little dependence on the number of sources and destinations, the size of transfers, etc. In a multidimensional system the bottleneck will be considerably more complex and will require multiple variables.

6. Benchmark specification examples

A typical benchmark situation will involve a specific pattern of transfers. This is the basis for the benchmark and will model the behavior of the surrounding system. Here we present two simplified examples.

6.1. Common specification

The two following traffic model cases have some general constrains/specifications in common. These are:
- The network is a 2-d mesh of size 8 by 8.
- All routers have one full duplex wrapper/IP block connected to each.
- The simulations are run for 1.000.000 cycles.
- The statistics are captured between 10% and 90% of the simulation time.

The primary goal of the example benchmarks is to find the accepted load bottleneck in the system. The *accepted load* is the traffic that is actually transferred across the network. The counterpart is the *offered load*, i.e. the amount of traffic that the sources try to put into the network.

6.2. Random transfers

The random transfers benchmark is to a large extent unconstrained. The closest real system would probably be a general purpose computing system, i.e. a parallel computer. The specification is as follows:
- All sources generate equal number of transfers during the simulation time.
- All transfers end up in a random destination.
- The transfers are distributed randomly over the simulation time according to a flat distribution.
- All transfers are of random size within the interval 32-1200 words according to a flat distribution. (616 words per transfer on average.)

6.3. Data plus control

The data plus control case can be though of as a very simplified model for many complex signal processing applications. It involves semi-deterministic, frequent data transfers intermixed with more infrequent but non-deterministic control information transfers. This example does not in any way claim to be a model of any real system but will - to a certain extent - model the typical behavior of many telecom applications. The system considered will involve 64 processing blocks connected to an 8 by 8 router network. Two of the processing blocks will run a real-time operating system (RTOS) and therefore dispatch the tasks to the other blocks through control messages.

The simplified specification can be summarized as follows:
- No data transfer will be over a distance longer than three hops (links). This means that all processing blocks will only be able to communicate with their nearest neighbors, see Fig. 1.
- The data transfers will be scheduled hard and will contain between 32 and 1200 words (randomly chosen) of information. Thus the data packets are on average 616 words.
- The control transfers will always originate in one of the two RTOS blocks and end in one of the other blocks. Control here is on the level of application control and has nothing to do with the network level control.
- The control transfers are randomly 10-30 words in size.
- There are a total of 100 control messages during the simulation time. This means that the ratio data/control messages will vary from 64 and up.

These specifications are implemented in the traffic model to form two test cases. The test cases are used to exercise the network in the simulator to produce a set of results. By repeating the benchmark implementation test cases with increasing intensity (i.e. total number of transfers for a given time) in the traffic it is possible to find the maximum performance in the form of effective bandwidth.

7. Results

7.1. Random transfers

The total number of links in the network system is 176[1]. Assuming a use of three links per connection (i.e. minimum distance) we get a theoretical maximum accepted load of approximately 59 word per cycle.

[1]An 8 by 8 network means $2 \cdot 7 \cdot 8 = 112$ links between routers and 64 links to wrappers.

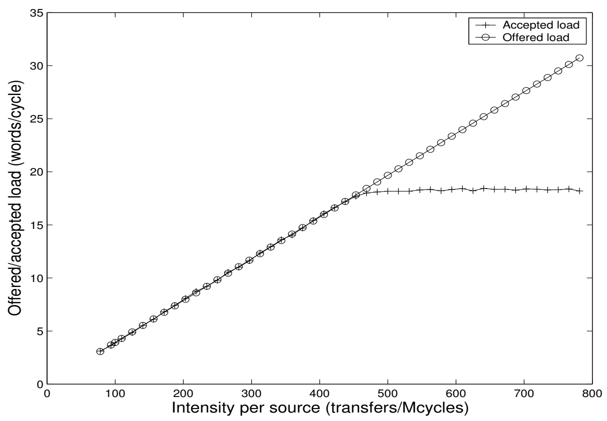

Figure 4. Offered vs. accepted load for the random case

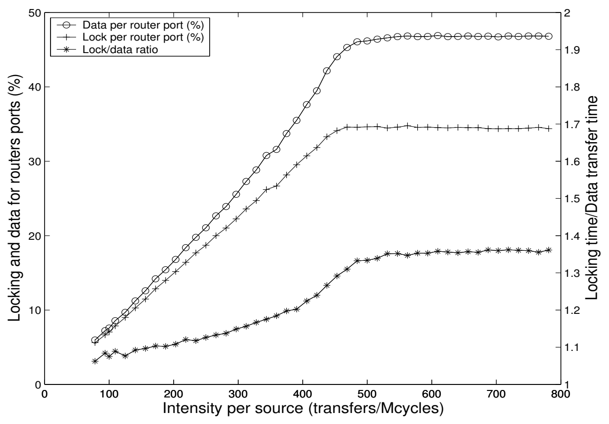

Figure 5. Total router occupation vs. data router occupation per port for the random case

Fig. 4 shows the offered vs. accepted bandwidth as a function of intensity. From the figure the saturation point of the network is around 470 transfers per million cycles. This corresponds to approximately 18.5 transferred words per cycle.

Fig. 5 shows the average time (in %) that the router inputs are locked vs. the time that data is actually sent. The figure also shows the ratio between these figures. The ratio goes up significantly around the saturation point showing that the PCC scheme actually will worsen the situation beyond the saturation point for random transfers.

This is further corroborated by Fig. 6. This graph shows the total number of routing blockings vs. the number of blockings due to a busy destination. This distinction is important because if the destination is busy two (or more) incoming transfers try to use the same port at the same time which of course is impossible. This is primarily a problem of the higher level system scheduling. But in this case the vast majority of the blockings are due to network situations

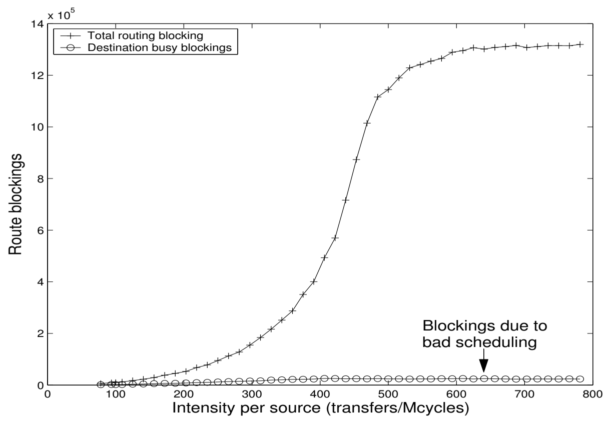

Figure 6. Total blockings vs. blockings because of busy destination for the random case

which shows that the network performance is the primary constrain for the overall performance.

7.2. Data plus control

Fig. 7 shows the offered vs. accepted load for the data+control test. As can be seen in the figure the load bottleneck is appearing at an intensity of about 650 transfers per source and Mcycle. This corresponds to an accepted load of approximately 35 words per cycle compared to the theoretical maximum of 59. This shows that this particular “application” and network combination performs at roughly 60% of theoretical maximum and that it performs on the level of between 35 and 59 conventional TDM buses with global connectivity.

Fig. 8 shows that the impact of the saturation point on the routing lock overhead is smaller than for the random case. This is due to the shorter average distances in the routing setup.

It is important to note is that Fig. 9 shows the opposite situation compared to the random case. Almost all of the routing blockings occur because of busy destinations, i.e. the system level scheduling. This implies that the performance of the network in reality is higher for a system with better traffic scheduling.

Embedded systems for (hard) real-time applications generally need to be scheduled. This is necessary in order to meet the rather stringent requirements on timing put on the system. This implies that the on-chip interconnect will have reasonably well scheduled traffic that will lower the destination busy blockings to a minimum and raise the overall network performance closer to the theoretical boundary.

8. Conclusions and future work

This paper identifies the importance of the simulation environment in the context of design flow for network on chip

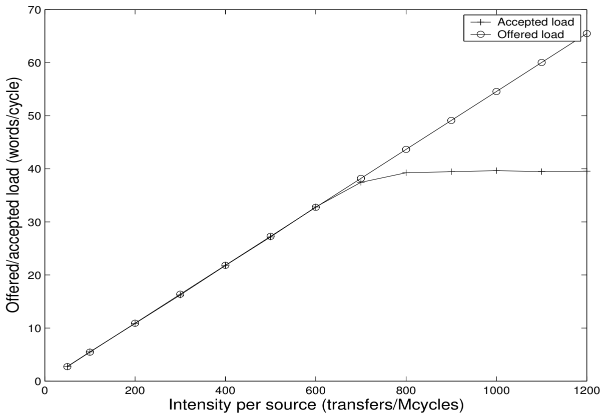

Figure 7. Offered vs. accepted load for the data + control case

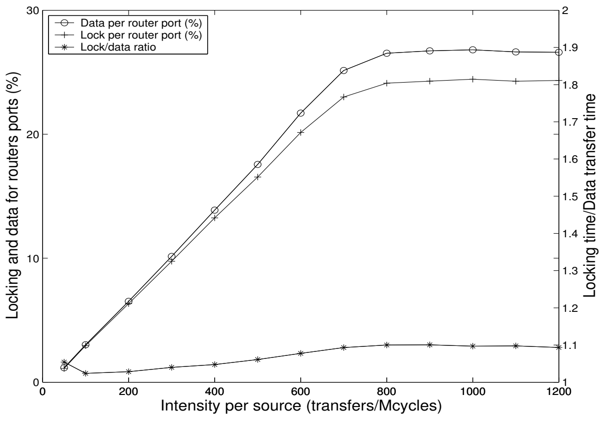

Figure 8. Total router occupation vs. data router occupation per port for the data + control case

designs. With the origin in the design and simulation flow developed for the SoCBUS network on chip project, two benchmark examples have been implemented and simulated to find the load bottleneck in the network. The results from the benchmarks show the validity of using a circuit switched network for (hard) real-time applications. Also shown is the importance and impact of the system level scheduling on the interconnect performance.

The upcoming tasks are to model and simulate several application examples from the networking and telecom area to find the bottlenecks and performance in these cases. Two applications that will be investigated are the baseband DSP part of a 3G basestation and a (Internet) network core router. These simulations will be done during spring 2004. Simultaneous work includes a more elaborate wrapper for connection to the OCP/Wishbone bus and improvements to the router design.

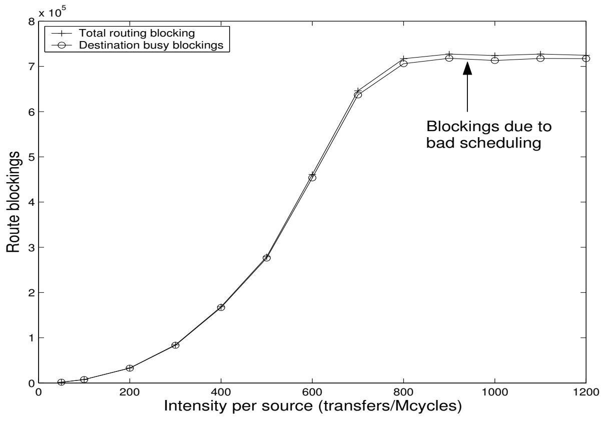

Figure 9. Total blockings vs. blockings because of busy destination for the data + control example

Acknowledgments

This work is supported by the Swedish Foundation for Strategic Research (SSF) through the Stringent electronics research center at Linköping University.

References

[1] Kees Goossens, J. van Meerbergen, A. Peeters, and P. Wielage, “Networks on silicon: Combining best-effort and guaranteed services,” in *Proceedings of the design automation and test conference*, Mar. 2002.

[2] William J. Dally and Brian Towles, “Route packets, not wires: On-chip interconnection networks,” in *Proc of the design automation conference (DAC)*, 2001.

[3] Daniel Wiklund and Dake Liu, “Design of a system-on-chip switched network and its design support,” in *Proc of the Int’l conference on communications, circuits and systems (ICCCAS)*, June 2002.

[4] Daniel Wiklund and Dake Liu, “Socbus: Switched network on chip for hard real time systems,” in *Proc of the Int’l Parallel and Distributed Processing Symposium (IPDPS)*, Apr. 2003.

[5] Daniel Wiklund, *An on-chip network architecture for hard real time systems*, Licentiate thesis, Linköping Studies in Science and Technology, Jan. 2003, ISBN 91-7373-577-9.

[6] Sumant Sathe, Daniel Wiklund, and Dake Liu, “Design of a switching node (router) for on-chip networks,” in *Proc of the ASICON 2003 conference*, Oct. 2003.

[7] Sumant Sathe, Daniel Wiklund, and Dake Liu, “Design of a low latency router for on-chip networks,” in *Submitted manuscript*, 2004.

[8] Santiago Gonzalez Pestana, Edwin Rijpkema, Andrei Radulescu, Kees Goossens, and Om Prakash Gangwal, “Cost-performance trade-offs in networks on chip: A simulation-based approach,” in *Proceedings of Design, Automation and Test in Europe Conference*, Feb. 2004.