Adv. Radio Sci., 7, 107–112, 2009
www.adv-radio-sci.net/7/107/2009/

Advances in Radio Science

Quantitative comparison of performance analysis techniques for modular and generic network-on-chip

M. C. Neuenhahn, J. Schleifer, H. Blume, and T. G. Noll

Chair for Electrical Engineering and Computer Systems, RWTH Aachen Univ., Schinkelstraße 2, 52062 Aachen, Germany

Abstract. NoC-specific parameters feature a huge impact on performance and implementation costs of NoC. Hence, performance and cost evaluation of these parameter-dependent NoC is crucial in different design-stages but the requirements on performance analysis differ from stage to stage. In an early design-stage an analysis technique featuring reduced complexity and limited accuracy can be applied, whereas in subsequent design-stages more accurate techniques are required.

In this work several performance analysis techniques at different levels of abstraction are presented and quantitatively compared. These techniques include a static performance analysis using timing-models, a Colored Petri Net-based approach, VHDL- and SystemC-based simulators and an FPGA-based emulator. Conducting NoC-experiments with NoC-sizes from 9 to 36 functional units and various traffic patterns, characteristics of these experiments concerning accuracy, complexity and effort are derived.

The performance analysis techniques discussed here are quantitatively evaluated and finally assigned to the appropriate design-stages in an automated NoC-design-flow.

1 Introduction

Due to advances in VLSI-technology chip complexity increases, leading to the implementation of Systems-on-Chip (SoC) that include several functional units (e.g. processor cores, memories and embedded FPGAs) on a single chip. With further technology improvements the number of components in a SoC grows continuously (Blume et al., 2005).

One key problem concerning SoC is the communication among the functional units of a SoC. Applications mapped to a SoC often come with complex communication demands: communication should feature low latency as well as a high throughput rate with a guaranteed quality of service (Angiolini et al., 2006). It should be flexible to map different applications to a SoC. Furthermore, the communication should cause minimal costs in terms of area and power dissipation. For compatibility to future SoC with even more functional units the communication architecture should be scalable (Wielage and Goossens, 2002).

Classic communication architectures like busses or point-to-point connections can not fulfill these requirements (Bainbridge and Furber, 2002). An often mentioned communication architecture which might solve these problems are so-called Networks-on-Chip (NoC) (Benini and De Micheli, 2002). These NoC are derived from computer networks and adapted to the needs of SoC. First prototypes and even products featuring NoC have been recently published (Vangal et al., 2007; Tilera Corporation, 2007).

The performance and costs caused by these NoC strongly depend on NoC-specific parameters like topology, data word length, routing algorithm and many more (Bjerregaard and Mahadevan, 2006). This multitude of parameters creates a large design space associated with NoC. Finding a suitable setup for each application thus requires tools for efficient performance analysis to evaluate the NoC in different stages of the design flow.

In this work several static and dynamic performance analysis techniques are quantitatively compared concerning

- simulation/emulation speed,
- monitoring options
- related modeling/analysis effort and
- accuracy

in various experimental setups differing for example in NoC size or traffic pattern.

Correspondence to: J. Schleifer
(schleifer@eecs.rwth-aachen.de)

Published by Copernicus Publications on behalf of the URSI Landesausschuss in der Bundesrepublik Deutschland e.V.

The static performance analysis is based on timing models, the NoC-parameters chosen and the traffic-patterns (see Sect. 3). The dynamic performance analysis techniques evaluated here are

– a VHDL-Simulator,

– a SystemC-Simulator,

– a Colored Petri Net-based simulation and

– a FPGA-based emulation.

Each of these techniques was first evaluated and then assigned to an appropriate design stage of a design flow for NoC.

2 Modular and generic NoC

The modular and generic NoC framework which was introduced in Neuenhahn et al. (2006) consists of three basic building blocks: Network Interface (NI), Routing Switch (RS) and Link. The Network Interface is the gateway between NoC and a functional unit, the Routing Switch routes data through the NoC and the link represents the connections between these building blocks (see Fig. 1). The arrangement of these blocks is defined by the NoC-parameter *topology*. The options of possible topologies range from classic topologies like mesh, torus or star topologies to application specific hierarchical topologies.

Each NoC component (NI, RS, Link) consists of different modules which can be implemented to provide the desired functionality. The modules for error-correction and – detection in NI and RS for example can be implemented using no coding, parity code or Huffman-codes (Neuenhahn et al., 2007).

3 Traffic-patterns

A reliable performance analysis of a NoC can only be performed for given communication demands of an application. These communication demands are usually given as traffic-patterns that can be distinguished into synthetic and application specific ones.

3.1 Synthetic traffic-patterns

Using synthetic traffic-patterns, the performance of NoC can be compared, as these traffic-patterns are representative for a certain application class.

These traffic-patterns are defined by the requested load of the NoC, the size of data blocks transferred over the network and the way traffic-source and traffic destination are chosen. For the last parameter different patterns are proposed like e.g. random traffic (Duato et al., 2003).

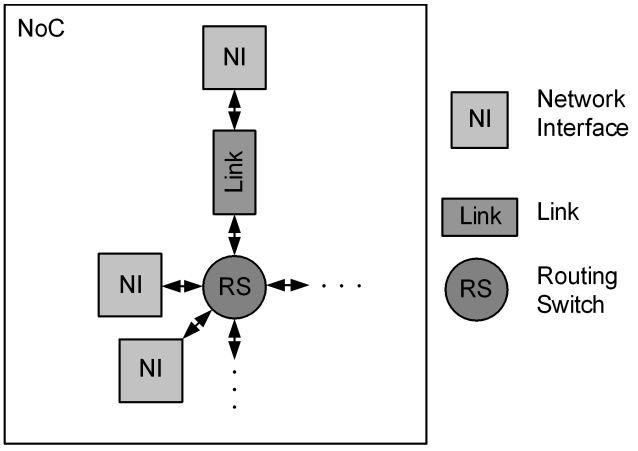

Fig. 1. Network-on-Chip.

3.2 Application specific traffic-patterns

The communication demands of an application can be modeled using a task graph (Dick et al., 1998). This graph consists of nodes representing certain tasks and directed edges, connecting these nodes. The weights of the edges represent an amount of data, which has to be transferred from one node to another. The weights of the nodes represent computation time. The task graph is restarted after a defined period of time, called hyper period.

4 Static performance analysis

During a static performance analysis traffic-patterns are mapped onto a NoC. Using this mapping, the expected loads on certain NoC-components are calculated. Dynamic effects, like e.g. already used links and/or repeated connection establishment due to blocked links can not be considered.

5 Dynamic performance analysis

5.1 FPGA-based NoC-emulator

The FPGA-based NoC-Emulator emulates a NoC on an FPGA (Neuenhahn, 2007). As even modern FPGAs lack the capacity to emulate a complex large SoC the functional units are replaced with abstract data sources and sinks.

The FPGA-based generic emulation environment consists of an FPGA and a PC. The PC is used to synthesize the FPGA-configuration, initiate experiments and evaluate the experiment results. The FPGA contains the NoC to be emulated (see Sect. 2), traffic-sources and -sinks which emulate functional units of a SoC and a soft-core processor (Nios II, Altera Corporation, 2007) which is used to control the data-sources and evaluate the raw results generated by the data-sinks.

5.2 SystemC-based NoC-simulator

Using a SystemC based simulation represents one possible alternative for performance analysis. SystemC is a C++ class library, which combines hardware description languages like VHDL or Verilog-HDL and object-oriented programming languages like C++ (IEEE, 2006). In addition, the open source SystemC library provides simulation capabilities.

Many SystemC-based simulators have been presented at different levels of abstraction. In this work a cycle-accurate NoC simulator was used. The structure of this simulator resembles that of the FPGA-based emulator presented in Sect. 5.1. It consists of a SystemC description of the NoC and a SystemC-model of data sources and sinks, which are controlled by the same C++-code as used in the FPGA-based emulator.

5.3 Simulator using colored petri nets

Another possibility to model and analyze NoC performance is simulation based on a Colored Petri Net (CPN) model.

Petri Nets (Petri, 1962; Girault and Valk, 2004) provide a graphical modeling formalism based on automata theory and find widespread use in different fields of engineering (Dotolia and Fanti, 2006; Shang et al., 2004). Petri Nets feature two basic components: places and transitions. Places are related to states in an automata graph and can be marked with an arbitrary number of tokens. Transitions are connected to places as input and output and can remove and generate tokens in these places. By allowing several places to be marked at the same time, Petri Nets facilitate modeling of concurrent processes.

Modeling with Colored Petri Nets (CPN) (Jensen, 1980) is a further expansion of these possibilities, especially by inclusion of data structures into tokens.

For design space exploration purposes a library of CPN models of NoC components was compiled (Schleifer, 2008). As only traffic generation is relevant for NoC performance modeling, clients are replaced by simple data sources and sinks that generate traffic either according to a random distribution or a task graph. Messages passing through the network are modeled as tokens containing the appropriate control data. Each component of the NoC is modeled in a modular fashion. It is thus possible to exchange modules such as the router or traffic generator without further changes to the model.

5.4 VHDL simulator

The VHDL-code which is used for VLSI-implementation and the FPGA-based emulator can be used for simulation, too. The control-signals for data sinks have to be derived from the NoC-description.

For the evaluation of this analysis technique the simulation software ModelSim (Mentor Graphics, 2007) was used.

6 Experimental results

Several experiments have been performed to determine the simulation speed and the simulation accuracy of the presented analysis techniques. Exemplarily the following NoC-specific parameters were chosen:

- Mesh-topology,
- static XY-routing,
- 16-bit data word length,
- circuit switching and
- no error protection.

The NoC-Parameter NoC-size, i.e. the number of functional units attached to the NoC, was varied.

Furthermore, traffic-patterns with synthetic random data traffic were used. The traffic parameters

- requested load and
- data block size

were varied during these experiments. The experiments were executed on a PC using a Pentium Core Duo Processor (3 GHz) and 1 GB DDR RAM. The FPGA-based emulator was implemented on an Altera Stratix II FPGA (EP2S60, Altera Corporation, 2007).

6.1 Accuracy

The performance of NoC can be determined using different measures. In this case we used the average latency of the data transmitted over the NoC. The relative error of this measure was below two percent for all analysis techniques but the static analysis technique. Using static performance analysis, an accurate performance analysis can only be performed for traffic-patterns with a low usage of the NoC.

The results produced by the FPGA-based emulator have been taken as reference values, as the VHDL-code of the NoC-emulator is identical to the code used for the VLSI-implementation. Moreover, the high simulation speed enables experiments with huge amounts of data.

6.2 Simulation speed

To analyze the performance of traffic-patterns on NoC most of the analysis techniques need some preparation, which consumes a certain lead time. This preparation has do be performed once for each NoC, e.g. the VHDL-Code used in a VHDL-simulator has to be compiled and the FPGA-based emulator has to be synthesized.

In the case of CPN modeling, at least with the software tool employed here, a syntax check of the complete model has to be performed before simulation. In case of a five by five NoC this takes approximately 10 min. This syntax

Table 1. Lead time for different analysis techniques using a 5×5 NoC with 16 Bit data word length and 1 GBit data.

	Lead time
Static Analysis	–
VHDL simulation (compilation of the VHDL-code)	<1 min
SystemC simulation	<1 s
CPN simulation (syntax check)	<10 min
FPGA-based emulation (synthesis of NoC-emulator)	<1 h

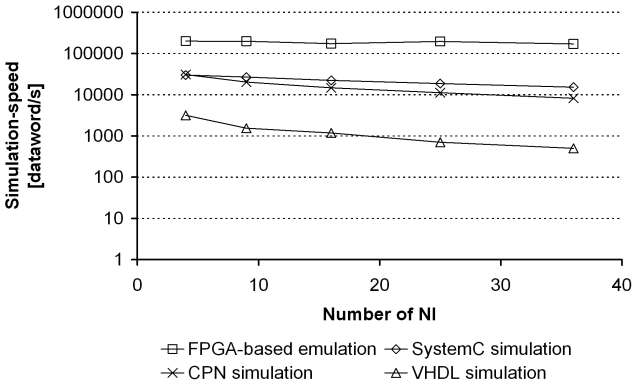

Fig. 2. Simulation speed of NoC using different NoC-sizes at 50% requested load and a data block size of 100 data words.

check is needed only after any major modification (like e.g. changing the topology of the NoC). The duration of this syntax check increases with NoC size and complexity. Small changes in the experimental scenario, such as longer messages or redistribution of traffic load amongst the different data sources, only require a partial syntax check, which is done in less than one minute.

The other techniques presented have no or a negligible lead time. In Table 1 lead times for different analysis techniques using NoC with 25 functional units and 16-bit data word length are compared.

The simulation speed in this work is measured in data words transferred over the NoC per second. Other metrics such as clock-cycles per second have been analyzed as well, which lead to similar results.

As no simulation is conducted for the static performance analysis, this technique is not regarded in the following section.

Simulation speed is affected by NoC-specific parameters as well as traffic patterns. The impact of each parameter is different for each analysis technique. As an example of NoC-specific parameter dependency the results of experiments varying NoC-size in a range from 2×2 to 6×6 (4 to 36 functional units) are shown in Fig. 2.

The number of FUs has a negligible impact on the simulation speed of the emulator due to the inherent parallelism of the FPGA. The simulation speed of the other techniques decreases with increasing number of FUs because the complexity of the NoC to be simulated increases. This effect is small for the SystemC-based simulator and stronger for the VHDL-simulator and the simulator using CPNs.

Furthermore the speed of both the FPGA-based emulator and the CPN-simulator depend linearly of the data block size. This dependency of the emulator is based on the fact that most of the time is spent generating traffic requests and evaluating received data traffic. As with an increased data block size the number of data blocks is reduced, simulation speed improves.

Due to the event-driven nature of the CPN-simulator its speed increases depending on the data block size. Increasing data word size reduces the number of events per transmitted data word and thus increases the simulation speed. The other techniques are not affected by the parameter data word size.

All these experiments show that using an FPGA-based emulator is at least one order of magnitude faster than the simulation using CPNs or the SystemC-based simulation which in turn are about one order of magnitude faster than the VHDL-simulation. CPN-based and SystemC-based simulations yield a comparable simulation speed. Depending on system-specific parameters (e.g. data block size) one or the other technique has its advantages.

7 Evaluation of performance analysis techniques

7.1 Static performance analysis

The advantages of a static timing analysis are short analysis times and good monitoring options, e.g. the utilization of a single link. As no dynamic effects can be considered, the accuracy of this technique is reduced if utilization of the NoC increases. Certain NoC-parameters like adaptive routing techniques are difficult to include into this analysis technique.

Due to the very short analysis time this technique can be advantageously used in early design stages, where various NoC-parameter settings (e.g. topology, data word length, clock frequency) can be compared to optimize the mapping of a task-graph onto the NoC. The reduced accuracy is negligible in this early design stage.

7.2 VHDL-simulator

The VHDL-simulation performs a cycle accurate simulation of the VHDL-code, which is used during the VLSI-implementation of the NoC. As every signal can be monitored an optimal tracing of the NoC is possible. Because the simulation speed of the VHDL-simulation is very slow compared to the other analysis techniques it is not applicable for huge NoC and extensive performance analysis.

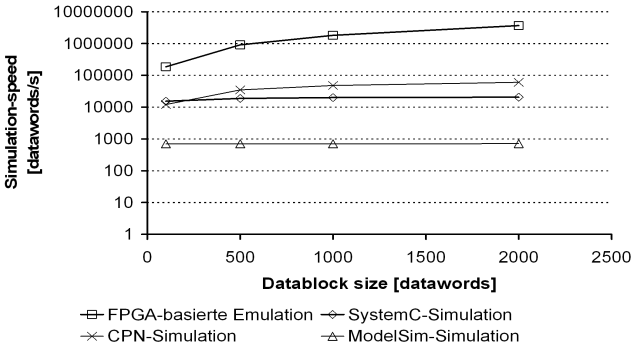

Fig. 3. Simulation speed for a NoC with 25 NI and varied data block size.

The VHDL-simulation is applied during the implementation of new NoC-components for the NoC-library. It is used for verifying the correct operation and for debugging purposes.

7.3 Simulator using Colored Petri-Nets

The model used for CPN/based simulation of a NoC in contrast to the other methods presented does not include modeling of the complete data transmission. Instead only the necessary control sequences are modeled as actual data is not relevant for network performance. The result of this approach is shown in Fig. 3: The less control overhead is needed for the transmission of a given amount of data the faster CPN-based simulation is. Furthermore, the CPN-based NoC model is very abstract and only includes parameters influencing network performance. Due to this small number of required parameters CPN modeling is well suited for performance analysis in early stages of the design flow. Since transition throughput and occupation of places in the CPN are easily accessible, all relevant data can be gained without major modification of the model.

This method of dynamic performance analysis allows identification of hotspots as well as characterization of NoC performance with parameters like average latency and load. Due to the interdependence of simulation speed and message size CPN based simulation is especially well suited to evaluation of traffic scenarios with large message sizes.

7.4 SystemC-based NoC-simulator

Using a cycle accurate SystemC-based simulator has advantages in performance analysis of NoC as no lead time is required, dynamic effects are considered while obtaining a quite high simulation speed. Other advantages are good monitoring options and the theoretical unrestricted NoC-size.

The disadvantages of this analysis technique are the increased modeling effort, as each NoC-component has to be modeled for simulation (SystemC) and VLSI-implementation (e.g. VHDL). Another important disadvantage is the fact that simulations with huge, real world data amounts last very long time and are therefore impractical.

This analysis technique should be applied in the design-stage "Dynamic performance analysis" of the NoC-design flow, using traffic-patterns with small to medium data amounts. Bottlenecks can be identified during this simulation, too.

Table 2. Qualitative comparison of performance analysis techniques.

	static performance analysis	VHDL simulator	SystemC simulator	CPN simulator	FPGA-based emulator
simulation speed	+ +	– –	+	+	+ +
lead time	+ +	o	+ +	–	– –
consideration of dynamic effects	– –	+ +	+ +	+ +	+ +
monitoring options	+	+ +	+ +	+	–
simulation-accuracy	– –	+ +	o	o	+ +
arbitrary NoC size	+ +	o	+ +	– –	– –
modeling effort	+ +	o	+	–	o

7.5 FPGA-based NoC-emulator

The fastest simulation speed is achieved by FPGA-based NoC-emulation. It is about an order of magnitude faster than the simulation speed of simulations featuring CPNs and SystemC-based simulations. Additionally, the same source-code used for VLSI-implementation is used during the emulation, too. Hence, dynamic effects are perfectly considered at cycle-accuracy. The emulations speed of the emulator is about a magnitude slower than a NoC implemented on a SoC.

As the synthesis of the NoC-emulator results in a high lead time an application in early design-stages where many NoC-parameter settings have to be compared is not advisable. The size of NoC is limited due to the capacity of the available FPGAs.

Monitoring options in FPGA-based emulators are limited. The results of performance analysis are general measures like average or maximal latency of transmitted data.

The high simulation speed enables a final functional verification of the NoC and a performance analysis using real world data amounts and different traffic-patterns. The appropriate state where the NoC-emulator should be used in the design-flow is the dynamic performance analysis, too.

All the features for the different analysis techniques are summarized in Table 2.

8 Conclusions

In this work different performance analysis techniques for a generic and modular NoC architecture were presented, quantitatively evaluated and compared. For this evaluation traffic-patterns with real world data amounts for NoC featuring up to 36 functional units have been used. The comparison of the different modeling methods discussed shows significant differences in precision, simulation speed, lead time and modeling/analysis effort.

Based on these results the analysis techniques were evaluated and assigned to the appropriate design stage: in an early design stage a static performance analysis is used, as it is very fast at the cost of reduced accuracy. In the succeeding design stages other analysis techniques (SystemC-based simulation, Colored Petri-Nets (CPN) based simulation or FPGA-based emulation) are used. Which analysis technique is best suited in each case depends on the chosen NoC-parameters and parameters describing the applied traffic-patterns. For example, the SystemC-based simulation and the simulator using CPNs are applicable for small to medium data amounts, whereas the FPGA-based emulator can be advantageously used for real world data amounts.

References

Altera Corporation: Nios II Processor Reference Handbook, Altera Corporation, 2007.

Altera Corporation: Stratix II Device Handbook, Altera Corporation, 2007.

Angiolini, F., Meloni, P., Carta, S., Benini, L., and Raffo, L.: Contrasting a NoC and a Traditional Interconnect Fabric with Layout Awareness, in: Design, Automation and Test in Europe, DATE '06, 1–6, 2006.

Bainbridge, J. and Furber, S.: Chain: a delay-insensitive chip area interconnect, IEEE Micro, 22, 16–23, 2002.

Benini, L. and De Micheli, G.: Networks on chips: a new SoC paradigm, Computer, 35, 70–78, 2002.

Bjerregaard, T. and Mahadevan, S.: A survey of research and practices of Network-on-chip, ACM Comput. Surv., 38, p. 1, 2006.

Blume, H., Feldkaemper, H. T., and Noll, T. G.: Model-Based Exploration of the Design Space for Heterogeneous Systems on Chip, J. VLSI Signal Proc., 40, 19–34, 2005.

Dick, R. P., Rhodes, D. L., and Wolf, W.: TGFF: task graphs for free, The Sixth International Workshop on Hardware/Software Codesign (CODES/CASHE '98), 97–101, 1998.

Dotolia, M. and Fanti, M. P.: An Urban Traffic Network Model via Coloured Petri Nets, Control Eng. Pract., 14, 17 pp., 2006.

Duato, J., Yalamanchili, S., and Ni, L.: Interconnection Networks – An Engineering Approach, San Francisco, USA, Morgan Kaufmann Publishers, 479-482, 2003.

Girault, C. and Valk, R.: Petri Nets for Systems Engineering, Berlin, Springer, 9–68, 2004.

IEEE: IEEE Std 1666–2005 IEEE Standard SystemC Language Reference Manual, IEEE Std 1666–2005, 0_1-423 pp., 2006.

Jensen, K.: Net Models in System Development, Ph.D. thesis, Aarhus University, 1980.

Mentor Graphics: ModelSim, Mentor Graphics, 2007.

Neuenhahn, M. C., Blume, H., and Noll, T. G.: Quantitative analysis of network topologies for NoC-architectures on an FPGA-based emulator, Kleinheubacher Tagung, 2006.

Neuenhahn, M. C., Blume, H., and Noll, T. G.: Quantitative Design Space Exploration of Routing-Switches for Network-on-Chip, Kleinheubacher Tagung Miltenberg, 145–150, 2007.

Neuenhahn, M. C., Lemmer, D., Blume, H., and Noll, T. G.: Quantitative Cost Modeling of Error Protection for Network-on-Chip, ProRISC Workshop, Veldhoven, 331–337, 2007.

Petri, C. A.: Kommunikation mit Automaten, Ph.D. thesis, Institut für instrumentelle Mathematik, Friedrich-Wilhelm-University Bonn, 1962.

Schleifer, J., Blume, H., and Noll, T. G.: Performance Analysis of Networks on Chip Using Coloured Petri Nets, in: ProRISC 2008, Veldhoven, The Netherlands, 94–99, 2008.

Shang, D. B., Koelmans, F., Yakovlev, A., and Xia, F.: Asynchronous system synthesis based on direct mapping using VHDL and Petri nets, IEE P.-Comput. Dig. T., 151, 12 pp., 2004.

Tilera Corporation: TILE64™ Processor Product Brief, Tilera Corporation, 2007.

Vangal, S., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Iyer, P., Singh, A., Jacob, T., Jain, S., Venkataraman, S., Hoskote, Y., and Borkar, N.: An 80-Tile 1.28TFLOPS Network-on-Chip in 65 nm CMOS, in: IEEE International Solid-State Circuits Conference (ISSCC), Digest of Technical Papers, 98–589, 2007.

Wielage, P. and Goossens, K.: Networks on silicon: blessing or nightmare?, in: Digital System Design, 196–200, 2002.