BrownPepper: a SystemC-based Simulator for Performance Evaluation of Networks-on-Chip

Jaison Valmor Bruch, Magnos Roberto Pizzoni and Cesar Albenes Zeferino
Embedded and Distributed Systems Group
University of Vale do Itajaí – UNIVALI
São José, BRAZIL
{jaison, magnospizzoni, zeferino}@univali.br

***Abstract*— The design space of Networks-on-Chip (NoCs) is very large, and there are several architectural alternatives for implementation. In order to select the best configuration of a NoC for a given application, it is necessary to have tools which aid the designer in the evaluation of each configuration. In this context, this work presents an integrated environment based on SystemC which allows evaluating different configurations of a NoC by means of simulation. The SystemC models are implemented by using RTL (Register Transfer Level) and mixed RTL/TL (Transaction Level) modeling. The simulator includes an interface that automates the design and the execution of experiments. It also provides instruments for performance evaluation by using graphical tools which make easier the analysis of results.**

Keywords-Systems-on-Chip; Networks-on-Chip; Performance Evaluation; SystemC; CAD tools.

I. INTRODUCTION

One of the problems in the design of future multi-core Systems-on-Chip (SoCs) with dozens of cores will rely on providing communication performance compatible with the requirements of the multiple processing cores.

As it has been largely discussed in the literature, bus-based architecture will no be able to deliver the communication performance required for future Systems-on-Chip (SoCs) with dozens of cores integrated onto a single chip [1]. As a solution for this problem, it is a consensus in scientific and industrial communities that on-chip switched networks will be the best communication architecture for futures SoCs, because they can meet the requirements of such systems. These architectures are largely known as Networks-on-Chip or NoCs [2][3]. A NoC can be described by its topology and by the techniques used for routing, switching, scheduling, buffering, and flow control, each of them with several alternatives for implementation. Therefore, the design space of NoCs is very large, and finding the best configuration of parameters for a given application is a very hard and time-consuming task.

In this context, this paper presents an integrated environment that allows the performance evaluation of a NoC for different configurations. This environment, named BrownPepper, is based on RTL (Register Transfer Level) and mixed RTL/TL (Transaction Level) SystemC models of SoCIN NoC [4], traffic generators and traffic meters. It integrates a set of tools developed to automate the tasks related to the simulation of the network operation under different traffic scenarios. It has a Graphical User Interface (GUI), which was developed in order to ease the configuration and the execution of experiments, and the analysis of results.

This paper is organized in six sections. Section II discusses some related works, while Section III summarizes the architecture of SoCIN NoC. Sections IV and V describe the architecture and the interface of BrownPepper. Following, Section VI illustrates the use of BrownPepper in an experiment of performance evaluation. Concluding, Section VII presents the final remarks.

II. RELATED WORK

Several studies about NoCs have been developed by different research groups around the world. Many of these works use some kind of simulation-based method to evaluate their proposals. In general, simulation models are built by using C/C++, VHDL, Verilog or SystemC [5]-[9]. Usually, these simulators are developed as in house CAD tools for internal usage, targeting the validation and evaluation of new architectural proposals. Many of them do not include GUIs, which could make easier the use by third party users.

One example of GUI-oriented simulation environment is MAIA [10]. It is an integrated environment with a visual interface designed to automate several processes related with the design flow of Hermes NoC [6]. It integrates tools for: (*i*) configuration and generation of VHDL models of Hermes; (*ii*) configuration and generation of traffic patterns to be applied on the network; (*iii*) simulation execution; and (*iv*) traffic analysis. For traffic generation, MAIA uses the model proposed in [11], which allows modeling different traffic patterns and offers alternatives for spatial and temporal distributions. MAIA is based on mixed SystemC/VHDL simulation and uses ModelSim from MentorGraphics® as simulation engine.

III. SOCIN ARCHITECTURE

SoCIN – SoC Interconnection Network [4] uses a 2-D mesh topology. Each router has up to five communication ports: one is a terminal used to attach a core and the other ones are used

for communication with the neighbor routers. Communication among cores attached to the NoC terminals is done by means of packets of unlimited length. SoCIN uses the ParIS (Parameterizable Interconnect Network) router [12] and its variations, which can be configured in order to apply different techniques of the NoC design space.

IV. BROWNPEPPER ARCHITECTURE

A. The SoC Platform

BrownPepper works with a SoC platform composed by a SoCIN network and instances of traffic generators (TGs) and traffic meters (TMs) attached to the NoC terminals. Each TG generates communication flows composed by a sequence of packets sent to a given destination under a specified traffic pattern. Besides injecting packets into the NoC, TGs also work as destination nodes. Each TM is responsible to monitor the traffic on a network terminal and collect data from the packets arriving at the TG attached to it. In order to stop the simulation, a centralized block named StopSim determines when all the packets are delivered to their destinations.

B. Traffic modeling

For traffic modeling, BrownPepper applies the model proposed in [11]. Each TG generates one or more communication flows, each one composed by a sequence of packets to be sent to a same destination node and based on a common set of traffic parameters. The major parameters include: (*i*) the required bandwidth; (*ii*) the number of packets to be sent; (*iii*) the address of the destination node or a spatial distribution used to define the destination node(s); and (*iv*) the type of injection rate, which can be constant or variable (also including bursting injection).

Additionally to these parameters, a communication flow can be classified in one of four classes related with QoS (Quality of Service) requirements. These classes are derived from the ones defined in QNoC [7]. In order to analyze how the NoC meets the QoS requirements, one can set a deadline (maximum End-to-End delay) to each flow. After simulation, it is possible to verify the rate of met (or missed) deadlines.

C. The SystemC library

BrownPepper is based on a set of SystemC models of the SoC components. ParIS is implemented as an RTL description based on the original synthesizable VHDL model of ParIS router [12]. TG is described as an RTL/TL model, using high level functions for traffic generation, and RTL blocks for network interfacing. TM is based on this same approach, using high level functions to extract traffic measurements from the monitored packets, which are written into an output file. The StopSim block is described as an RTL model.

Besides these components, the SystemC library includes two structural descriptions: *socin* (the network) and *main* (the SoC). These two models and the StopSim block depend on the system size, and their descriptions are automatically generated by tools which are discussed in the following sub-section.

D. Design flow

BrownPepper is based on a design flow composed by a set of tools used for: (*i*) generating the SystemC simulator; (*ii*) generating the traffic model; (*iii*) running the simulation; and (*iv*) analyzing the results.

Fig. 1 shows the first part of the BrownPepper design flow. The system parameters (*i.e.* system size and router configuration) are read by two code generation tools. The first one, named *gnoc*, generates the SystemC model for SoCIN. The second tool, named *gsoc*, generates the SystemC model for the SoC platform (main), for the StopSim block, and a file with the parameters used to configure the other SystemC models in the library. After code generation, the models are compiled by *gcc*, and the SystemC simulator is created (file *system.x*).

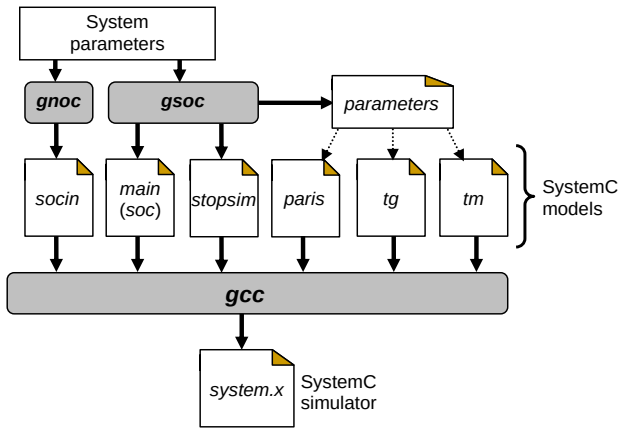


Figure 1. Generation of the SoC SystemC simulator.

The second part of the design flow is depicted in Fig. 2. After *system.x* is created, a traffic model file is generated by *gtr* tool. After that, the simulator is run and the traffic meters extract measurements and write them into output log files. These files are processed by *atr* tool, which generates reports for performance analysis by the designer.

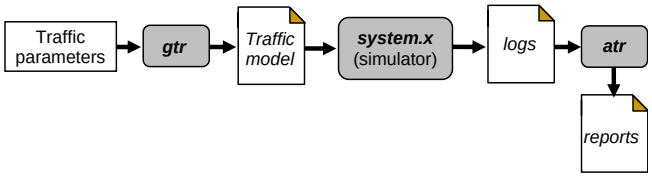


Figure 2. Generation of the traffic model, simulation and results analysis.

E. Design flow

All the tools implemented for the design flow were coded in C, totalizing more than 3,300 lines of code. The SystemC models, excepting the ones automatically generated by *gnoc* and *gsoc* tools, totalize more than 5,800 code lines.

V. GRAPHICAL USER INTERFACE

BrownPepper GUI is organized in three tabs, which are described in the following subsections.

A. *System Configuration tab*

The first tab of BrownPepper (shown in Fig. 3) is used to configure the system size, the channel width and the traffic model. A traffic pattern is added to the traffic model by selecting its check box and clicking on its *Edit* button. After that, a dialog box is presented, allowing the designer to specify the parameters of the traffic pattern. In the dialog box for node (0, 0), a button allows the replication of the traffic pattern to all the other nodes, saving work when the same traffic pattern needs to be defined for all the other nodes.

The major feature of the implemented interface is that it allows designers to easily specify very complex traffic models in which a single node can generates different traffic patterns for different classes.

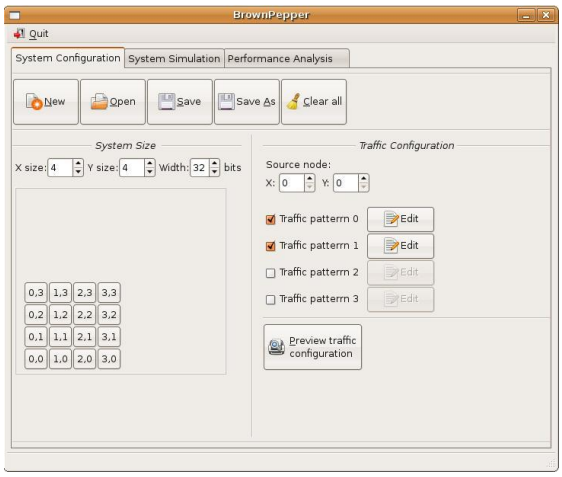


Figure 3. BrownPepper GUI: the *System Configuration* tab.

B. *System Simulation tab*

This tab (shown in Fig. 4) allows the designer to easily configure and execute multiple experiments for up to five different router configurations. It also allows setting up a range of network operating frequency to be varied in order to analyze the relationship between the traffic model requirements and the network bandwidth. If the user wants to obtain a waveform diagrams, it can request the generation of the VCD (Value Change Dump) file needed for that.

By configuring the experiments and clicking on the Run button, the BrownPepper design flow is executed iteratively to run the multiple experiments. The first part of the design flow (shown in Fig. 1) is executed once for each router configuration, and the second part of the design flow (shown in Fig. 2) is executed once for each clock frequency in the specified range (for each router configuration).

Different of typical tools in which the network bandwidth is fixed and the bandwidth required by the application is varied, in BrownPepper, no changes are done in the traffic model. Since the operating frequency is varied in a range (and therefore the network bandwidth), the offered load also varies because the "required bandwidth / network bandwidth" relationship varies too. In this way, one can explore the design space in order to determine the minimal operating frequency that allows meeting the application requirements.

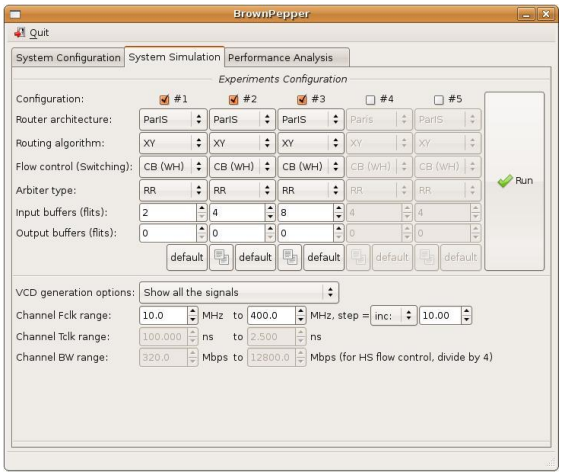


Figure 4. BrownPepper GUI: the *System Simulation* tab.

C. *Performance Analisys tab*

This tab (Fig. 5) provides the instruments for performance analysis. It allows the visualization of the reports generated by *atr* at each iteration of the BrownPepper design flow. After the experiments are run, this tab allows the designer to analyze the performance metrics for all the communication flows, for a specified traffic class or for a selected flow. The analysis tools allow viewing results by means of tables or by using graphics generated by GNUPlot (a command-line driven interactive data and function plotting utility). Also, for a single experiment, it is possible to plot a waveform diagram based on the generated VCD file by using GTKWave (a fully featured GTK+ based waveform viewer).

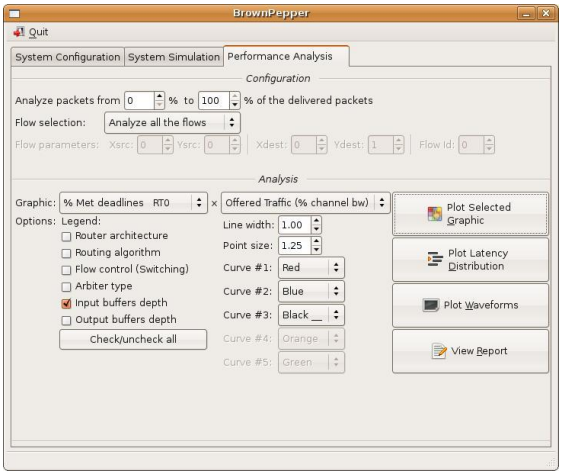


Figure 5. BrownPepper GUI: the *Performance Analisys* tab.

D. *Implementation*

The GUI of BrownPepper is based on GTK+ and was implemented using GLADE, a framework for the development of visual interfaces based on GTK+ and GNOME. The manual written code of the interface consumes about 5,000 code lines.

VI. USING BROWNPEPPER

This section aims at illustrating how BrownPepper can be used to evaluate the performance of a NoC. A 4×4 32-bit system was defined and each traffic generator was configured with the traffic patterns described in Table I.

TABLE I. TRAFFIC PATTERNS USED IN THE EXPERIMENTS

Parameter	Traffic Pattern 0 (TP0)	Traffic Pattern (TP1)
Spatial distribution	Non-Uniform	Non-Uniform
Traffic class	Signaling (RT0)	Audio/Video (RT1)
Type of injection	Constant injection	Constant injection
Deadline	100 ns	125,000 ns
Required bandwidth	320 Mbps	320 Mbps
Message size	128 bits	640 bits

The first pattern emulates the sending of interrupts and alarm messages, while the second one emulates periodic connections of 320 voice channels (PCM-64 Kbps) with a maximum End-to-End delay of 125 μs. These patterns are based on the traffic model used in [7].

In the experiments, it was fixed the router architecture (XY, routing, credit-based flow control and round-robin arbitration) and the depth of the input buffers was varied (2-, 4- and 8-flit). Fig. 6(a) shows that deeper buffers allow reducing the network operating frequency before the saturation threshold is reached. On the other hand, Fig. 6(b) shows that if deadlines requirements are taken into account, it is possible to determine the minimum operating frequency to reach a target rate of met (or missed) deadlines for a given network configuration.

VII. CONCLUSION

This paper presented BrownPepper, an integrated environment for performance evaluation of NoCs based on SystemC simulation. BrownPepper includes a set of tools which automates several tasks necessary for the exploration of the design space of NoCs. A lot of efforts were invested in order to build a robust and easy way to use it. Almost 35% of the code lines is due to the GUI, while 23% is due to the design flow tools and 42% is due to the SystemC library.

One of the important features of BrownPepper is that it is totally based on free software and technologies. Differently of some similar works, it does not use any tool that depends of some kind of payable license for use.

Ongoing developments include de implementation of features aiming at measure the switching activity in the network in order to estimate the power consumption. Future works include the implementation of optimization functions which will allow to automatically finding the best configuration for a given application, by taking into account the requirements of a target application.

ACKNOWLEDGMENT

This work was supported by the Brazilian research agency CNPq within the scope of the projects 475768/2006 and 476465/2008.

REFERENCES

[1] P. Guerrier and A. Greiner, “A generic architecture for on-chip packet-switched interconnections”, In: Proc. of DATE, 2000. pp. 250-256.

[2] L. Benini and G. De Micheli, “Networks on Chip: a new SoC paradigm”, Computer, vol. 35(1), pp. 70-78, Jan. 2002.

[3] A. Jantsch and H. Tenhunen (Eds.), Networks on Chip, Kluwer, 2003.

[4] C. A. Zeferino and A. A. Susin, “SoCIN: A parametric and scalable Network-on-Chip”, In: Proc. of 16th SBCCI, IEEE CS Press, 2003. pp. 169-174..

[5] A. Andriahantenaina et al., “SPIN: a scalable, packet switched on-chip micro-network”, In: Proc. of DATE, 2003. pp. 70 73.

[6] F. Moraes et al. “A low area overhead packet-switched Network-on-Chip: Architecture and Prototyping”. In: Proc. of VLSI-SoC, 2003. pp. 318-323.

[7] E. Bolotin et al., “QNoC: QoS architecture and design process for Network-on-Chip”, Journal of Systems Architecture, vol.50, pp.105-128, 2004.

[8] J. Chan and S. Parameswaran, “NoCGen - a template based reuse strategy for Networks-on-Chip”, In: Proc. of the 17th Int. Conf. on VLSI Design, 2004. pp. 70 73.

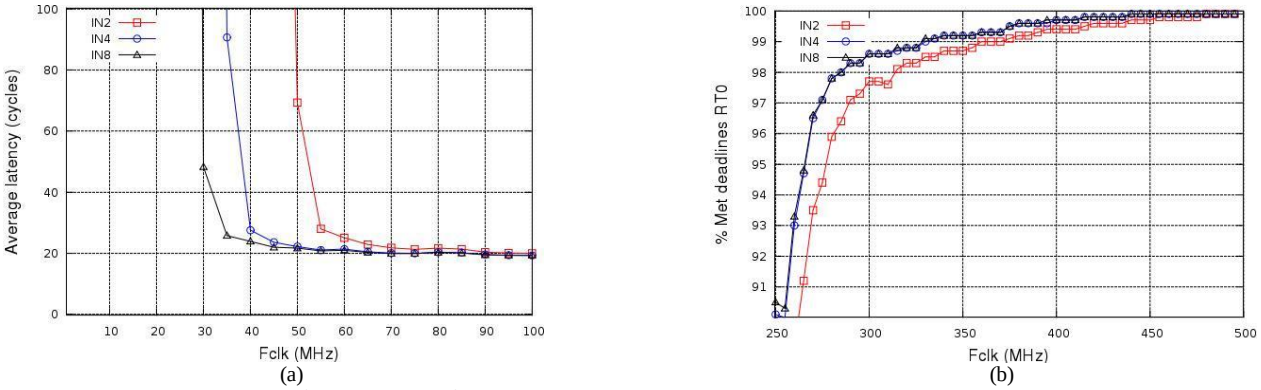
[9] M. E. Kreutz et al., “Design space exploration comparing homogeneous and heterogeneous Network-on-Chip Architectures” In: Proc. of the 18th SBCCI, 2005.pp. 190-195.

[10] L. C. Ost et al., “MAIA - a framework for Networks on Chip generation and verification”. In: Proc. of ASP-DAC, 2005. pp.18-21.

[11] L. P. Tedesco et al. “Traffic generation and performance evaluation for mesh-based NoCs”, Proc. of the 18th SBCCI, 2006. pp. 184-189.

[12] C. A. Zeferino; F. G. M. Santo, A. A. Susin, “ParIS: a parameterizable interconnect switch for Networks-on-Chip”, In: Proc. of 17th SBCCI, ACM Press, 2004. pp. 204-209.

[13] J. Duato, S. Yalamanchili and L. Ni, Interconnection networks. Los Alamitos, IEEE CS Press, 1997. p. 406.



(a) (b)

Figure 6. Analyzing results of experiments performed using BrownPepper: (a) Average latency × Fclk; (b) %Met Deadlines RT0 × Fclk.