A Flexible Network-on-Chip Simulator for Early Design Space Exploration

Cristian Grecu, André Ivanov, Resve Saleh
Department of Electrical and Computer Engineering
University of British Columbia
Vancouver, Canada

Claudia Rusu, Lorena Anghel
TIMA Laboratories
Grenoble, France

Partha P. Pande
School of Electrical Engineering and Computer Science
Washington State University
Pullman, USA

Vasile Nuca
City Express Corp.
Vancouver, Canada

Abstract— The communication requirements of large multi-core systems are conformed by on-chip communication fabrics generally referred to as networks-on-chip (NoC). We have designed a simulation environment that allows early exploration of the performance and cost parameters of network-on-chip communication architectures, which is able to handle arbitrary topologies and routing schemes. The simulator implements a flit-level message-passing mechanism and supports application data specified as input trace files or generated at run-time by synthetic traffic generators.

Network-on-chip; simulator; performance;

I. INTRODUCTION

In recent years, the importance of on-chip interconnects has surpassed the importance of the transistors as a dominant factor of chip performance. The ITRS roadmap document [1] enumerates a number of technological challenges for interconnects and states that “technology alone cannot solve the on-chip global interconnect problem with current design methodologies”.

The NoC design paradigm [2] is aimed at solving the global interconnect challenge for multi-core systems-on-chip through a shift in the design approach, where designers, rather than focusing mostly on the engineering of the computational cores with little concern for the inter-core wires, must employ structured, intelligent communication fabrics for transporting data across the chip.

The use of NoCs as communication fabrics for multi-processor system-on-chip (MP-SoCs) raises many challenges that are actively investigated by the research community. Of major importance is the support of research with CAD tools that can allow rapid experimentation with multiple design options [3].

A distinct step in the overall NoC design flow is the communication architecture simulation, where a NoC is exercised against predefined requirements defined at different abstraction levels, from application level down to physical level.

The level of abstraction at which data exchange is represented is extremely important for the nature and accuracy

of design options that can be exercised and the information that can be extracted from the simulation.

However, there is a gap in the data and hardware representation in existing simulation tools, between the low-level representations specific to RTL-level simulators, and high-level, transaction-based simulators. In this context, we present a simulation tool designed to bridge the gap between low- and high-level simulation approaches. It can be used independently or as a complement to other tools.

The NoC simulator addresses the evaluation of NoC performance early in the design stage, without requiring a very detailed representation of the NoC under simulation, and allows comparison of different architectures and communication protocols in a uniform environment. The source code of this tool is publicly available and can be downloaded from <http://www.ece.ubc.ca/~grecuc/simulator>.

II. RELATED WORK

The need for simulation tools for the analysis and design of NoC architectures appeared concurrently with the proposal of NoCs as a viable solution for the global communication problem of large MP-SoCs. Due to the lack of dedicated software, researchers used communication networks simulators, such as ns-2 [4] and OPNET [5], which are relatively complex, difficult to configure and adapt for on-chip scenarios. The latter are targeted towards the simulation of very large networks with complicated protocols (i.e., TCP-IP and similar ones) and traffic models not representative for MP-SoC applications.

The difficulties associated with using non-optimal tools have motivated research groups to develop a few dedicated NoC simulators. A major challenge when designing such tools is to achieve the right balance between the level of abstraction of data and NoC representation, and the accuracy of results and simulation speed.

We can mainly differentiate two categories of NoC simulators, based on the granularity of data and hardware representation, both having their merits and advantages. The first category includes simulators that use a low-level representation of the NoC components and data. Zeferino et al. [6] have ported a parameterized VHDL model onto an FPGA

to evaluate the behaviour of different router implementations based on HDL simulations. Moraes et al. [7] designed a tool set that generates NoCs based on mesh topologies of different sizes, with components configurable in terms of flit size, buffer size, and routing schemes. The VHDL model of the NoC thus generated can be then simulated using commercially available VHDL compilers and waveform viewers. This type of approach makes difficult (even impossible in some cases) the monitoring of NoC parameters such as link utilization, hot-spots, etc. More importantly, exercising a wider range of topologies and routing mechanisms is only possible by rewriting the part of code that generates the HDL-level description based on a high-level specification of the topology, which is a time consuming effort. There are also portability issues involved by the use of platform-dependent commercial software.

NoC simulators in the second category use a high-level representation of both NoC hardware and data. The data exchange is modeled at message level or transaction level, where a transaction between two processes may consist of one or more messages. Cornelius et al. [8] presented a NoC simulation tool that models data at packet level, and routers as design entities with a configurable number of ports interconnected by links with configurable width. The timing assumption is explicit and untimed (the delays of links must be specified in absolute time units, NoC has no awareness of a common time reference). A similar approach was presented by Kogel et al. [9], with data processed at transaction level by a network engine that can implement high-level behaviour of point-to-point, bus-based, and crossbar interconnects. Coppola et al. [10] presented OCCN, a modeling and simulation framework that features a layered model for inter-module communication, with each layer translating transaction requests to a lower-level protocol.

The high level of abstraction and complexity of these types of simulators makes the evaluation of features such as different topologies, variable buffers size for routers, link utilization, and routing strategies, difficult to perform in an expeditious manner.

Other tools were developed in-house by chip design companies [11] and platform providers [12]. However, they are not publicly available to the community, and not many implementation details and features are known.

III. NOC MODEL FOR SIMULATION

In this section we present the basic assumptions for NoC architectures that can be handled by the tool.

We work under the assumption that a NoC infrastructure is built of two major types of components: routers (switches) and links. A third component – the IP (Intellectual Property) core – is employed to implement the communication behaviour of the NoC functional cores. These components can be connected together to build NoCs of arbitrary topologies and sizes.

We use a flit-level representation for data, and a cycle-accurate simulation model. In the next section, we present the set of features that characterizes our simulator. The structure of the simulation environment is presented in Fig. 1.

The core of the simulator is the simulation engine, written in Java, which processes the messages at flit level, implements routing policies and flow control, and collects measurement data.

Fig. 1: Block diagram of the NoC simulator

The information on the NoC topology and routing policy is contained in two different input files (*topology.xml* and *routing.xml*). The relevant data is extracted by an XML parser and fed to the simulation engine. The NoC input traffic information is defined in text format, and the corresponding file (*traffic.txt*) is read during the simulation. Data packets are created by the simulator according to the traffic description in this file. Finally, status information and measurement data are written in the *output.txt* file.

A. IP model

The *networkIP* class describes the behaviour of the NoC functional cores (further denoted as IP – *Intellectual Property* cores) with respect to their communication interface. They serve two main purposes:

1. Extract messages from the traffic input file and place them into their internal communication queues, whose size is user configurable.
2. Inject/extract data flits to/from their output/input ports.

Fig. 2: IP model and traffic injection

Messages are moved from the traffic input file into the IP internal communication queue in exactly one simulated clock cycle. From there, data is moved into the output port of the IP, one flit per simulated clock cycle (if free space exists in the IP's output buffers).

B. Router model

The NoC switch model is based on the pipelined model [13]. By default, a three-stage pipeline corresponding to an input/output buffering scheme is employed, shown in Fig. 3a. The three stages correspond to input buffering,

routing/arbitration, and output buffering operations, respectively. In cases where the NoC uses switches with less or more than three stages, the switch model can be modified accordingly by editing the *networkswitch* Java class.

Switches and IPs exchange data through *ports* (as depicted in Fig. 3b, which are unidirectional in the current implementation). The number of ports depends on the particular NoC topology and is defined in the *topology.xml* input file.

Fig. 3: (a) NoC switch model; (b) IP cores, switches and links connected in a NoC

Each switch can have a number of virtual channels, which can be set by user. The capacity of each virtual channel (in terms of number of flits it can store) can also be set by the user.

C. Link model

Each NoC link connects two router ports or a router port and an IP port. In the current implementation, all links are unidirectional. For bidirectional communication, a pair of links in opposite directions must be instantiated. The latency (in number of cycles) along the link can be set by the user. The default link latency value is one.

D. Data switching model and packet organization

The switching protocol that governs the data transmission is wormhole routing, in which packets are divided into basic flow control units (flits). The first flit of each packet (the header flit) reserves the routing resources (buffers and links) along the path between a source and a destination. The data flits, carrying the useful information, follow the path reserved by the header, while the tail flit (the last flit of the packet) ends the transmission and frees the resources reserved by the header. At any moment during simulation, a flit can be found in one of the network resources: buffers or links.

E. NoC simulator features:

1. Topology flexibility: the NoC simulator is able to simulate any topology, regular (mesh, torus, tree-based, etc.) or irregular (the case of application-specific networks-on-chip).

2. The level of abstraction for data representation is intermediate, between the low level representation (RTL – register transfer level - for VHDL based simulators), and high level of abstraction (transaction level simulators).

3. Ability to work with various traffic sources: traces files collected from applications running on real systems or generated synthetically off-line; run-time, reactive traffic generators; or behavioural models of real processing elements executing a real application. This feature is enabled by the use of communication queues in the IP model, which can be filled with data from different sources: trace files, traffic generators, etc.

4. “Safety” features: deadlock detection, topology consistency checking (check whether unconnected NoC elements exist), hotspot detection, routing consistency checker (i.e., check whether a route between a source and a destination exists before sending a message). Also, the capability to have automatically generated control messages (for instance, ACK/NACK and ARQ messages) is available.

5. Clocking: in the current implementation, all NoC components are driven by a unique clock reference. This allows for simulating fully synchronous and plesiochronous NoC systems.

IV. EXPERIMENTS

The results presented in this section are obtained using the direct mesh topology (an IP is connected to each NoC switch), due to its simplicity and wide use.

The first experiment shows the message latency for different traffic granularities. An 8x8 NoC is considered. Two different simulations were run, both with the same traffic load. Results are depicted in Fig. 4. The first simulation (Fig. 4a) uses shorter messages (20 flits) and higher message injection rate (0.01 messages/cycle/IP), while the second one (Fig. 4b) uses longer messages (40 flits) and lower injection rate (0.005 messages/cycle/IP).

The message latency is measured from the moment the message header leaves the source IP communication queue, to the moment the header reaches the input port of the destination IP.

The X and Y axes represent the node coordinates in the mesh NoC. The message latency is represented on the vertical axis, and it is an average of the received messages latencies for each mesh node.

In the first case (Fig. 4a), lower message latencies can be observed when compared with the second one (Fig. 4b).

Fig. 4: Message latency for different traffic granularities

In fact, the message latency increases with the message length as the switching technique (wormhole) reserves NoC resources for the entire message length transportation.

This experiment reveals that marginal nodes present higher average latency than central nodes, which is the expected behaviour for a mesh topology.

Next, we analyze the message latency for two different traffic loads. The same message length is considered for both cases (20 flits), but different injection rates (0.01 and 0.03 messages/cycle/IP). Simulation results on the same 8x8 mesh NoC are presented in Fig. 5.

Fig. 5: Message latency for different traffic loads

As the traffic load increases, the message latency also increases. This is due to the higher utilization of the NoC resources in case of higher traffic load.

The simulation duration depends on the size of the simulated NoC and the traffic load. The next experiment presents the traffic load influence on the simulation duration. Simulations were run injecting uniform traffic in a 4x4 mesh NoC, for 100,000 cycles. Fig. 6 depicts the simulation duration and the throughput for different traffic loads (constant message length of 20 flits, but different message injection rates).

Considering traffic loads under the saturation level for the respective NoC configuration, the simulation duration increases linearly with the traffic load (Fig. 6a). However, for a traffic load approaching 0.06 injected messages/cycle/IP, the simulation duration increases drastically. This is due to the fact that for relatively high traffic loads, the maximum throughput for the current NoC configuration is reached (Fig. 6b).

(a) Simulation duration (b) NoC throughput

Fig. 6: Simulation duration and NoC throughput for different traffic loads

V. CONCLUSIONS AND FUTURE WORK

We developed and presented a NoC simulator that can successfully serve at optimizing the decisions space in early stages of NoC architecture design. Due to its flexibility, it allows diverse design choices on NoC hardware. Other advantages of the simulator are portability (enabled by the programming language used - Java), speed, and ease of utilization. We plan to extend the simulator for use with more complex input sources [14] and add the ability to handle multiple clock domains.

REFERENCES

- [1] International Technology Roadmap for Semiconductors, ITRS 2007, <http://www.itrs.net/Links/2007ITRS/Home2007.htm>
- [2] L. Benini, G. De Micheli, "Networks on chips: a new SoC paradigm", *Computer*, Vol. 35, Issue. 1, January 2002, pp: 70 -78.
- [3] J.D. Owens, W.J. Dally, R. Ho, D.N. Jayasimha, S. W. Keckler, L-S. Peh, "Research Challenges for On-Chip Interconnection Networks", *IEEE Micro*, Vol. 27, Issue 5, Sept.-Oct. 2007, pp: 96-108.
- [4] ns-2 - The Network Simulator, http://nsnam.isi.edu/nsnam/index.php/Main_Page
- [5] OPNET, <http://www.opnet.com/>
- [6] C.A. Zeferino, M.E. Kreutz, A.A. Susin, "RASoC: A Router Soft-Core for Networks-on-Chip", *Design, Automation and Test in Europe Conference and Exhibition Designers' Forum (DATE'04)*, 2004, pp: 30198.
- [7] F. Moraes, N. Calazans, A. Mello, L. Möller, L. Ost, "HERMES: an infrastructure for low area overhead packet-switching networks on chip", *Integration, the VLSI Journal*, Vol. 38, Issue 1, October 2004, pp: 69-93.
- [8] C. Cornelius, F. Sill, D. Timmermann, "High-Level Simulations of On-Chip Networks", *9th Euromicro Conference on Digital System Design (DSD)*, pp: 57-58.
- [9] T. Kogel, M. Doerper, A. Wiefenink, R. Leupers, G. Ascheid, H. Meyr, S. Goossens, "A modular simulation framework for architectural exploration of on-chip interconnection networks", *CODES+ISSS 2003*, pp: 7-12.
- [10] M. Coppola, S. Curaba, M.D. Grammatikakis, G. Maruccia, F. Papariello, "OCCN: a network-on-chip modeling and simulation framework", *Design, Automation and Test in Europe Conference and Exhibition (DATE'04)*, Vol. 3, pp: 174- 179.
- [11] P. Paulin, C. Pilkington, E. Bensoudane, "StepNP: a system-level exploration platform for network processors", *IEEE Design & Test of Computers*, Nov/Dec 2002, Vol.19, Issue 6, pp: 17-26.
- [12] Arteris NoCexplorer, <http://www.arteris.com/productnocexplorer.htm>
- [13] P.P. Pande, C. Grecu, M. Jones, A. Ivanov, R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures", *IEEE Transactions on Computers*, Vol. 54, Issue 8, pp: 1025-1040.
- [14] C. Grecu, A. Ivanov, P.P. Pande, A. Jantsch, E. Salminen, U. Ogras, R. Marculescu, "Towards Open Network-on-Chip Benchmarks," *The First International Symposium on Networks-on-Chip (NOCS'07)*, 2007, pp: 205.