Noxim: An Open, Extensible and Cycle-accurate Network on Chip Simulator

Vincenzo Catania*, Andrea Mineo*, Salvatore Monteleone*, Maurizio Palesi† and Davide Patti*
* University of Catania, Italy. first.last@dieei.unict.it
† Kore University, Enna, Italy. maurizio.palesi@unikore.it

***Abstract*—Emerging on-chip communication technologies like wireless Networks-on-Chip (WiNoCs) have been proposed as candidate solutions for addressing the scalability limitations of conventional multi-hop NoC architectures. In a WiNoC, a subset of network nodes are equipped with a wireless interface which allows them long-range communication in a single hop. This paper presents Noxim, an open, configurable, extendible, cycle-accurate NoC simulator developed in SystemC which allows to analyze the performance and power figures of both conventional wired NoC and emerging WiNoC architectures.**

I. Noxim Simulator

Recently, wireless on-chip communication mechanisms have been emerged as technological alternatives to the metal/dielectric system. A wireless NoC (WiNoC) [1] uses a wireless backbone upon the traditional wire-based NoC [2]. In this paper, we present a radically improved and extended version of Noxim, whose main novelty is the capability of simulating heterogeneous wired/wireless NoC architectures.

The Noxim simulator is developed using SystemC, a system description library written in C++. This choice is motivated by the fundamental requirements behind the Noxim project: allowing extensibility easiness and scalable performances while still supporting a cycle-accurate simulation. From a top-level perspective, the configuration of a specific network-on-chip architecture instantiates two main conceptual elements: a set of *tile nodes* and a *communication infrastructure*. Each tile node implements some computational/storage task, exchanging data asynchronously with other nodes by means of the communication infrastructure. The actual instance of the architecture is entirely determined by the *NoC configuration*, allowing the customization of several parameters of a NoC, each one potentially impacting not only the behavior of the NoC but also the time required for performing the simulation. The NoC instance is then simulated in the Noxim Runtime Engine (NRE), which contains the SystemC code for supporting the different NoC architectural elements and models. The NRE allows several topologies, buffer and packet sizes, traffic distributions, routing algorithms, packet injection rates, *etc.*. At the end of the simulation, several execution statistics are generated, both in terms of performance (delay, throughput) and energy-related figures. These information are delivered to the user both in terms of average and per communication results.

II. Noxim Configuration

The configuration space supported by the Noxim simulator consists of several parameters, which can be grouped in the following classes: topology and structure, workload, dynamic behavior and simulatotion. The class of *topology and structure* includes essential aspects of the NoC as the total number of instantiated nodes and the type of interconnection between them. With regard to nodes, Noxim classifies them into two categories, namely, tile nodes and hub nodes. In the first group, there are the computational/storage nodes of the NoC, while, in the second, there are nodes that modelize a sort of "gateway" for grouping tiles and shortcutting distant regions of the network. Three different types of interconnections that can be instantiated: (a) *Tile-Tile* wired point-to-point connection between two tile nodes (b) *Tile-Hub* wired point-to-point connection between a tile and a hub element (c) *Hub-Hub*: a connection between two hub elements. As it will described later, these connections can be both wired and wireless. Another important feature that further expands the set of topologies supported by Noxim is the inclusion of wireless communication mechanisms in Hub-Hub interconnections, modelized with the concept of *channel*. The *workload* class is related to all the parameters that affect the amount and the kind of computation that is given as input to the NoC. These are strictly related to the tasks that are performed in each tile and Noxim provides several commonly used data traffic models which abstract typical communication patterns [3]. In addition, it is possible to simulate a real application mapping its task communication graph into a customised table-based traffic. Parameters related to the *dynamic behavior* determine the way the NoC components perform decisions during the simulation. These may include decisions related to the routing of the packets between different network paths, the choice between a wireless or wired communication, *etc.*. While the most commonly used routing strategies [4], [5], [6], [7], [8] are already available in the Noxim Runtime Engine, an important feature is the plugin-like approach that easily allows to experiment any new algorithm with a little effort. The last category of parameters regards the simulation setup, that can be adapted to better match needs in terms of budget of time (the time available to perform the simulation) amount of required statistics, *etc.*.

III. Noxim Anatomy

In this section, we provide details on each single element of Noxim, both from the perspectives of NoC architectural choices and simulator design. They are: (a) *Tile nodes*, which represent the main components of the Noxim architecture. Although they can be conceptually associated to the computing/storage elements, more functionalities are required in order to support the distributed nature of data communication typical of a NoC based approach. (b) *Processing elements* (PE),

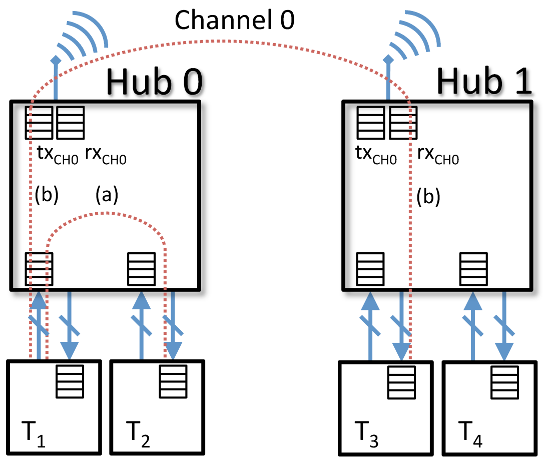

Fig. 1. Internal architecture of hub nodes and wireless communication

which are the core components actually performing computing/storage operations, usually mapped to one of the tasks in which the whole computation has been partitioned (*e.g.*, hardware implementation of a decoder or a small memory). A PE is responsible for generating and consuming data packets and its behavior at run-time strictly depends on the Workload parameters. (c) *Routers*, which are the architectural elements responsible for dispatching data packets, implementing flow control and most of the communication-related mechanisms in the network. (d) *Hubs*, components that play a fundamental role in the Noxim architecture, allowing the communication between tiles that are not adjacent. This further expands the set of topologies supported since regions of the network that are spatially distant can be thought as topologically connected via Hub-Hub connections. In wired routing, the hub simply moves the flits between two tiles, acting as an intermediate. A more interesting form of communication is the Hub-Hub wireless connection (see Figure 1). The only requirement for the Hubs in this case is having at least one channel in common. For the sake of clarity we will assume a simple wireless policy that uses wireless if the current tile is connected to a hub that can reach another hub connected to the final destination. In other words, packet flit travel across the network using whatever traditional routing algorithm (*e.g.*, XY), but switching to wireless if destination is reachable from a hub connected to the current tile. (e) *Channels*, components that abstract a flit transmission using a given wireless frequency. Only one hub per time can occupy the frequency for the transmission, thus some kind of access control mechanism must be adopted to avoid collisions/interferences. In order to simulate wireless communications, the Transaction Level Model (TLM) library of SystemC has been integrated in Noxim. One advantage of TLM is allowing a higher level of abstraction that assumes as granularity the whole transmission delay instead of a single clock cycle. In this way, Noxim can easily compute the T_{delay} required for transmitting the flit dividing the *flit size* (bits) by the *Data Rate* (bit/s) of the antenna. Then, the value obtained can be directly set as transaction delay in TLM, so that the receiver will be acknowledged of the completed transmission after a proper amount of time.

IV. Power Models

The power contribution of the network interfaces, routers, links, and the electrical part of the hubs is taken into account for computing the power and energy figures. The energy contribution of the basic element e in a generic clock cycle c is computed as:

$$E(e,c) = \alpha(e,c)P_{avg}(e)T_{CK},$$

where, T_{CK} is the clock period, $P(e)$ is the average power dissipated by the basic element e, and $\alpha(e,c)$ is the activity function which returns 1, if e is active on clock cycle c, and 0 otherwise. Overall, for a N cycles simulation, the total wired communication energy is computed as:

$$E_{total}^{(wired)} = \sum_{c=1}^{N} \sum_{e \in BE} E(e,c) + N \times \sum_{e \in BE} E_{leak}(e),$$

where we indicated with BE the set of the basic elements in the NoC and with $E_{leak}(e) = P_{leak}(e)T_{CK}$ the leakage energy of the basic element e spent during a period T_{CK}. All the basic elements of the NoC have been modelled in VHDL and synthesized using Synopsys Design Compiler considering a 65 nm CMOS standard cell library from UMC operating at 1 GHz. Power contributes reported in [9] and [10] have been considered for the wireless front-end.

V. Conclusions

Exploring the design space of a NoC by assessing its power and performance requires the availability of fast and accurate simulation tools. In this paper we presented Noxim, an open source, cycle accurate platform for the simulation of both conventional wire-based NoCs and emerging WiNoC architectures.

References

[1] D. Zhao and Y. Wang, "SD-MAC: Design and synthesis of a hardware-efficient collision-free QoS-aware MAC protocol for wireless network-on-chip," *IEEE Transactions on Computers*, vol. 57, no. 9, pp. 1230–1245, 2008.

[2] S. Deb, A. Ganguly, P. P. Pande, B. Belzer, and D. Heo, "Wireless NoC as interconnection backbone for multicore chips: Promises and challenges," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 2, no. 2, pp. 228–239, 2012.

[3] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA: Morgan Kaufmann, 2004.

[4] C. J. Glass and L. M. Ni, "The turn model for adaptive routing," in *25 Years ISCA: Retrospectives and Reprints*, 1998, pp. 441–450.

[5] J. Hu and R. Marculescu, "DyAD - smart routing for networks-on-chip," in *ACM/IEEE Design Automation Conference*, San Diego, CA, USA, Jun. 7–11 2004, pp. 260–263.

[6] G.-M. Chiu, "The odd-even turn model for adaptive routing," *IEEE Transactions on Parallel Distributed Systems*, vol. 11, no. 7, pp. 729–738, 2000.

[7] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "Application specific routing algorithms for networks on chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 20, no. 3, pp. 316–330, Mar. 2009.

[8] V. Catania, A. Mineo, S. Monteleone, and D. Patti, "Distributed topology discovery in self-assembled nano network-on-chip," *Computers & Electrical Engineering*, vol. 40, no. 8, pp. 292 – 306, 2014.

[9] X. Yu, J. Baylon, P. Wettin, D. Heo, P. P. Pande, and S. Mirabbasi, "Architecture and design of multichannel millimeter-wave wireless noc," *Design Test, IEEE*, vol. 31, no. 6, pp. 19–28, Dec 2014.

[10] X. Yu, S. P. Sah, Rashtian, H. Rashtian, S. Mirabbasi, P. P. Pande, and D. Heo, "A 1.2-pj/bit 16-gb/s 60-ghz ook transmitter in 65-nm cmos for wireless network-on-chip," *Microwave Theory and Techniques, IEEE Transactions on*, vol. 62, no. 10, pp. 2357–2369, Oct 2014.