MAIA – A Framework for Networks on Chip Generation and Verification

Luciano Ost[1], Aline Mello[1], José Palma[2], Fernando Moraes[1], Ney Calazans[1]

[1]FACIN-PUCRS
Av. Ipiranga, 6681 - Porto Alegre – 90619-900 – BRAZIL
{ost, alinev, moraes, calazans}@inf.pucrs.br

[2]II - UFRGS
Caixa Postal 15064- 91501-970 - Porto Alegre – BRAZIL
jcspalma@inf.ufrgs.br

Abstract - The increasing complexity of SoCs makes networks on chip (NoC) a promising substitute for busses and dedicated wires interconnection schemes. However, new tools need to be developed to integrate NoC interconnection architectures and IP cores into SoCs. Such tools have to fulfill three main requirements: (i) automated NoC generation; (ii) automated production of NoC-IP core interfaces; (iii) seamless analysis of NoC traffic parameters. The objective of this paper is to present the MAIA framework, which includes functions to address all these requirements. NoCs generated by the MAIA framework have been used to successfully prototype SoCs in FPGAs.

1. Introduction

The increasing complexity of *Systems-on-Chip* drives the research of new intra-chip interconnection architectures. Traditional on-chip interconnection architectures, such as *dedicated wires* and *shared busses*, can be considered inefficient for future SoCs. Dedicated wires present poor reusability and flexibility, while shared busses transmit only one word per clock cycle and offer limited scalability.

According to ITRS estimation, in 2012, SoCs will have hundreds of hardware blocks (called IP cores), operating at clock frequencies near 10 GHz. In this context, a Network on Chip (NoC) appears as a possible solution for future on-chip interconnections. A NoC is an on-chip network [1] composed by cores connected to routers, and routers interconnected by communication channels.

Shared bus architectures are the current dominant on-chip communication structure used in SoCs. Consequently, CAD tools are available to support shared bus architectures. NoC design requires new features, such as support for different switching modes, routing algorithms, and message packets segmentation and reassembly procedures [1][2][3].

It is important to develop tools for NoC generation and verification, due to the expected relevance of NoCs in future SoCs. The objective of this paper is to present a framework which allows exploring and optimizing SoC designs employing the HERMES[1] NoC [4].

The objective of this paper is to present the MAIA framework for NoC generation and verification. MAIA can generate different traffic patterns, for different load conditions and source/target pairs. Using the generated traffic and the automatically produced simulation scripts, it is possible to validate and evaluate the NoC and the associated SoC using commercial tools such as the Modelsim simulator. MAIA can also generate OCP (*Open Core Protocol*) network interfaces (NIs). While generating NIs, the tool is capable of supporting different communication models.

Finally, MAIA contains a module named *traffic analyzer*. It verifies if all packets were correctly received, and generates basic statistic data concerning time do deliver packets. Besides, this module can be used to validate the NoC internal implementation.

This paper is organized as follows. Section 2 presents related works in NoC design space exploration. The main features of the HERMES infrastructure are briefly presented in Section 3. Section 4 details MAIA main features. Section 5 presents how the MAIA framework can be used to evaluate the performance of different NoC configurations. Section 6 presents conclusions and directions for future work.

2. Related Work

Several research groups have proposed techniques to specify, simulate and generate NoCs. Some of these try to adapt generic network simulators to the intra-chip environment, while others propose NoC specific tools. This Section reviews several such efforts, including: (i) NoCSim, (ii) OPNET, (iii) Kogel framework, (iv) NoCGEN, (v) NS-2, (vi) OCCN, (vii) Pestana environment, and (viii) NOCIC.

NoCSim is a NoC simulator based on IP cores communicating through a packet switching network. NoCSim generates limited forms of statistic traffic, using Constant Bit Rate and random Poisson distributions [4].

OPNET [6] is a general purpose network simulator used to simulate NoC-based architectures [7]. OPNET presented some disadvantages to modeling NoCs, including that: (*i*) it does not allow setting a time unit smaller than 1 second; (*ii*) distance between nodes in the network is measured in meters only; (*iii*) OPNET assumes asynchronous communication.

NS-2 is another general purpose network simulator that gives support to describing the network topology, the communication protocols, routing algorithms and traffic (*e.g.* random traffic) [8]. NS-2 provides simulation traces for interpreting results and provides NAM (Network AniMator), a graphic aid to observe network message flow.

Kogel et al. [9], proposed a modular framework for system level exploration of the on-chip interconnection architecture. This framework is able to capture performance effects (like latency and throughput) of different on-chip architectures like shared bus and NoC topologies.

NoCGEN creates VHDL NoC descriptions used for simulations and synthesis [10]. This tool employs a set of parameterizable templates to build routers, with variable number of ports, routing algorithms, data width and buffer depth. Besides NoC parameterization, it presents a mixed SystemC/VHDL simulation environment.

[1] In Greek mythology, HERMES is the messenger of Gods, son of MAIA and ZEUS.

This work was partially funded by grant 550009/03-5, from CNPq, Brazil.

Another framework for NoC modeling and simulation is the *OCCN* [11]. *OCCN* enables the creation of NoC at different abstraction levels, protocol refinement, design space exploration, and NoC components development and verification based on a communication API [11].

Pestana presents in [12] a NoC simulator based on user-generated XML files that describe a NoC topology, the IP to NoC mapping and the interconnection details. The simulator also allows describing traffic generators to evaluate NoCs.

Finally, the *NOCIC* tool allows estimating performance and power for NoC structures using a set of design parameters simulated with HSPICE tools [13].

The MAIA framework has some features in common with NoCGEN, including automated traffic and network generation. One significant difference between MAIA and the previously reviewed works is that the latter are centered on the network itself, with little reference on how to map NIs to the rest of the SoC. An exception is [12]. MAIA automatically generates the NIs, using the OCP standard [14].

3. HERMES Network-on-Chip

HERMES is an infrastructure used to implement low area overhead packet-switching NoCs for different topologies, flit sizes, buffer size, routing algorithms, and flow control strategies [4]. It supports the implementation of the three lower OSI-RM layers, namely physical, data link and network. Initially, HERMES is based on NoCs using only wormhole routing.

The basic component of this infrastructure is the router. It contains a centralized control logic module, responsible for arbitration and routing, and up to five bi-directional ports (East, West, North, South, and Local). Each port has an input buffer for temporary storage of flits. The Local port connects the router and its local IP core.

Two flow control strategies are available: handshake protocol and credit based. When a 4-phase asynchronous handshake protocol is used, the external router interface is composed by six signals: *rx*, *ack_rx* and *data_in* for input and *tx*, *ack_tx* and *data_out* for output. When credit based flow control is used, a transmission clock is sent to the receiver and a *credit* signal is asserted from the receiver to the transmitter indicating available buffer space. Signal *ack_rx* does not exist in this case. This flow control algorithm enables implementing GALS networks.

Several instances of NoC-based systems were successfully prototyped in FPGAs [4]. An example system contains two simple 16-bit processors, an embedded memory and an interface IP core to provide communication with a host processor. It is estimated that a *5x5* HERMES-based system can be implemented in a 4-million gate device, with the same small 16-bit processor connected to the Local port of each router.

4. MAIA Features

The design of NoCs is an error prone process, even if routers and IP to NIs are assumed given. This is due to the large number of wires used to connect routers among them and IP cores to routers. Each router has a set of control and data signals that amount to more than a 100 wires. The primary function of MAIA is to build NoCs from parameterizable templates. Fig. 1 shows the MAIA design flow.

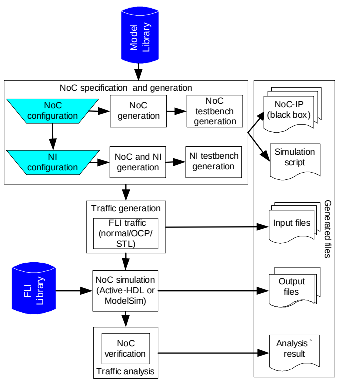


Fig. 1 – MAIA framework structure and execution flow.

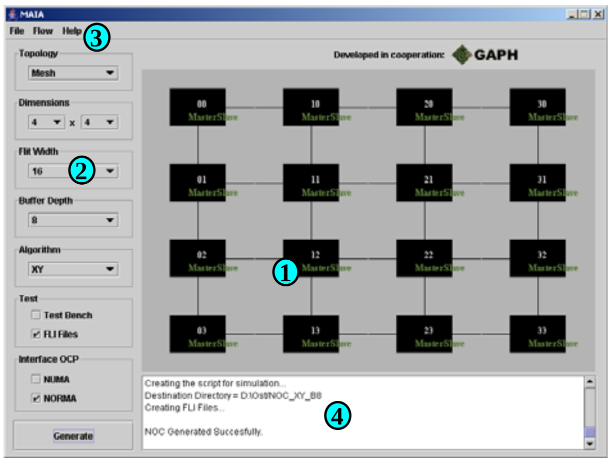
Three steps compose the design flow:

(i) *NoC specification and generation* comprise: (1) selection of the network parameters; (2) selection of the external interface – native or OCP; (3) selection of the communication model; (4) selection of the IP type, if OCP interface is selected. After parameter selection, all VHDL and C files are created. A UNIX script is created as well, to start simulation.

(ii) *Traffic generation* produces the packet files, according to chosen parameters as described in Section 4.3.

(iii) *Traffic analysis* creates traffic analysis reports.

Fig. 2 displays the primary graphical interface of MAIA. The visualization area (1) represents a 4x4 mesh network composed by routers with master-slave OCP NIs. The NI type is individually chosen, on a router by router basis. The second region (2) allows the user to select network parameters. These are used to configure the VHDL implementation files. The third region (3) contains menus used to start simulation, traffic generation and traffic analysis. The fourth region (4) presents messages resulting from execution of user operations.



Topology: Mesh
Dimensions: 4 x 4
Flit Width: 16
Buffer Depth: 8
Algorithm: XY
Test: Test Bench [✓ FLI Files]
Interface OCP: NUMA [✓ NORMA]

```
Creating the script for simulation...
Destination Directory = D:\Ost\NOC_XY_88
Creating FLI Files...

NOC Generated Succesfully.
```

Fig. 2 – MAIA framework graphical user interface.

The main contribution of the MAIA design flow is to relieve the user from worrying about the NoC structure and its internal functionality, i.e. enabling the use of NoCs as a generic communication IP core – a *NoC-IP*. It also simplifies the interconnection of IP cores to the NoC. The only requirement imposed on the MAIA user is to know in advance the structure of the transactions supported by the NIs at the network boundaries.

4.1. *Network Generation*

The NoC components in the Model Library can be described in RTL VHDL, RTL SystemC and Transaction Level (TL) SystemC. The integration of parameterizable SystemC components is an ongoing work. All results presented in this paper regard VHDL models only. Currently, the following parameters can be chosen by the MAIA user: (*i*) network topology, (*ii*) routing algorithm, (*iii*) flit width, (*iv*) buffer size, (*v*) network structure, and (*vi*) control flow strategy. To parameterize the NoC, a set of tokens are inserted in the VHDL code, such as: (*i*) \$*flit_size*\$, (*ii*) \$*buff_depth*\$, and (*iii*) \$*router_no*\$.

Network ports and associated buffers, can be automatically by MAIA, depending on the network topology and specific router position. For example, in a mesh topology some buffers of routers at the edges of the network are always eliminated. Consider the upper left router in a mesh network. The North and East buffers are suppressed, contributing to reduce the overall interconnect area. Today, MAIA supports three NoC topologies: mesh, torus and ring.

Two other parameters affect network performance: flow-control strategies (handshake or credit based protocols are supported) and routing algorithms. Deadlock free routing algorithms for torus and ring topologies are not yet available. Virtual channel support that can be used to build these is another ongoing work. For mesh topologies, four routing algorithms are available: pure XY, west-first, north-last and negative-first [3].

4.2. *Network Interfaces Generation*

A NoC is formed by two main components: routers and NIs. The NI is responsible for packet segmentation and reassembly. NIs should be designed taking into account the *communication model.*

Since a NoC-based SoC is similar to a parallel processing system in communication structure, it is possible to employ classical *communication models* from the latter domain. The Uniform Memory Access (UMA) model is not useful in the intra-chip domain. Accordingly, MAIA only supports two other communication models: non-uniform memory access (NUMA) and no-remote memory access (NORMA).

The NUMA communication model assumes all IP cores connected to the NoC share a single address space. Communication then takes place using memory-mapped I/O operations. This model is adequate for fast migration of legacy IP cores to NoC-based SoCs, once most bus architecture-based systems assume this communication model. The NORMA communication model does not assume the existence of a global address map. Communication is achieved directly, through the exchange of messages between IP cores. In a NORMA SoC, IP cores are independent, providing services to other IP cores. However, the NI may have to deal with the overhead of adapting its IP core communication paradigm.

The *IP core-NI* interface can be either proprietary or standard. A proprietary interface reduces the reusability of the network but may improve its performance. A standard interface has opposite characteristics. The user can choose the native HERMES interface or an OCP interface. OCP defines a point-to-point interface between two IP cores. One of these operates as master and the other as slave [14]. Only the master sends commands to initiate transactions. The slave answers to the commands, receiving data from or sending data to the master. Typical master IP cores are processors, while a memory is often a slave IP core.

Using a standard interface does not change the way IP cores are developed, since they will still be exchanging the same information with its environment, as predicted by its specification. The form how this exchange occurs is by means of a standard industry-accepted procedure, as occurs with the PCI standard for microcomputer manufacturers. Thereby, IP cores reusability is higher and design time can be reduced, since IP core integration occurs at higher levels of abstraction, becoming a simpler process.

NIs generated from MAIA templates (master, slave and master-slave) have been certified using the *CoreCreator* tool [14], and prototyped in Xilinx FPGAs.

4.3. *Traffic Generation and Analysis*

The MAIA framework *Traffic generation* is responsible for testbench and traffic files generation. Traffic files contain packets, which are read by the IPs connected to the NoC. Files are generated in one of four formats: (*i*) VHDL testbenches for the native interface; (*ii*) C/VHDL testbenches for the native interface; (*iii*) C/VHDL testbenches for the OCP interface; and (*iv*) STL (*Sonics Transaction Language*) format, to be used in the *CoreCreator* tool [14].

The following parameters define a traffic: (*i*) network load [2]; (*ii*) number of packets each IP core sends; (*iii*) number of flits in each packet; (*iv*) flit size; (*v*) the target IP core, which can be random or fixed. According to these parameters, a set of input files is created. During simulation, another set of files is generated, containing the received packets and their respective time stamps (time spent by a packet to be transmitted from source to destination).

The files generated during simulation are read by the *Traffic analysis* module, which produces a report file. This report file presents some traffic analysis results, such as: (*i*) total number of received packets; (*ii*) average time to deliver the packets, in clock cycles; (*iii*) total time to deliver all packets, in clock cycles; (*iv*) the average, minimal, maximal and standard deviation time to deliver a packet, in clock cycles; and (*v*) the total simulation time, in seconds.

5. NoC Design Space Exploration

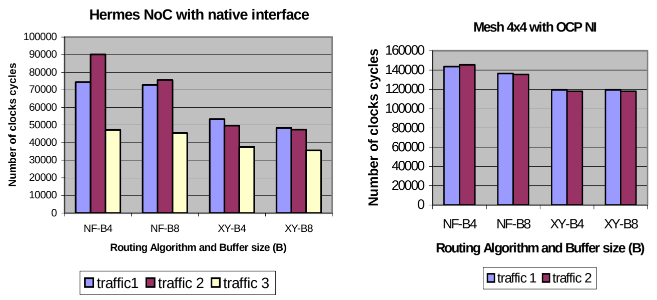
Using the MAIA framework, the performance of routing algorithms, the effect of buffer sizes and of external interfaces was evaluated.

All generated NoCs case studies have a 4 *x* 4 mesh topology, with a flit size equal to 16. The following structural parameters were varied: routing algorithm (XY and negative first, NF); buffer size (4 and 8 flits); external interface (native and OCP). Three randomly generated traffics were used, each with a fixed load of 70%. Table 1 presents these traffic characteristics. All examples employ a NORMA communication model.

Fig. 3 (a) illustrates the performance evaluation for different buffer sizes and routing algorithms, for different traffic conditions, using the HERMES native interface. On the other hand, Fig. 3 (b) reproduces the results obtained when using OCP interfaces.

Table 1 – Case study traffic characteristics (4 *x* 4 mesh).

	# of packets per router	# of flits per packet	Total # of flits
Traffic 1	100	100	160,000
Traffic 2	20	500	160,000
Traffic 3	500	10	80,000



(a) using HERMES NI. (b) using OCP interface.

Fig. 3 - Performance evaluation for different buffer sizes and routing algorithms, for different traffic conditions.

Some conclusions of these results are:

(i) *routing algorithms:* the performance of the XY routing algorithm was consistently superior to the NF algorithm.

(ii) *buffer size:* a very small advantage is observed when buffer size is equal to 8. Since NoC area is dominated by the buffer size, in this experiment, buffers with size 8 are oversized and unnecessary [4].

(iii) *traffic conditions*: in adaptive algorithms (NF), the use of smaller packets (traffic 1) improves the performance over larger ones (traffic 2), since packets can explore alternative paths in the network when blocking conditions arrive. An opposite behavior is observed in the deterministic algorithm (XY), where the performance of large packets is superior. The expected time to deliver all packets of traffic 3 would be half of that for traffic 1 or 2, since the number of flits to transmit is reduced to half. The overhead due to frequent routing/arbitration operations (small packets) reduces this performance.

A more extensive set of experiments showed that, in terms of total clock cycles to deliver all packets, deterministic XY routing is consistently faster than the other three partially adaptive algorithms. The latter can potentially speed up the time to deliver individual packets. However, globally, results point out to performance poorer than that of the XY algorithm. Glass and Ni [3], suggest that reducing the number of turns that a message takes may reduce blocking and hence improve performance. This is justified because adaptive routing tends to concentrate traffic in the center of the network, increasing the number of blocked paths. The *North-last* algorithm presents a small advantage over the XY algorithm for 30% traffic load and small packets (10 and 100 flits). This situation leads to a reduced number of blocked paths and the availability of idle time between packets. As the XY algorithm cannot explore different paths, even when they are available, adaptive algorithms have a potential advantage in this case.

Fig. 3 (b) illustrates the performance when using the OCP interface. This Figure also showed that buffer size 8 is oversized and that the XY algorithm showed superior performance over NF. The most import result concerning Fig. 3 (b) is the cost of adding a standard NI. As already mentioned, standard interfaces have the advantage of improving reuse (plug-and-play feature). On the other hand, performance is penalized. Comparing Fig. 3 (a) to Fig. 3 (b), packets are delivered to their targets almost 50% faster in the NoC without OCP NI. Currently, OCP interfaces are attached and adapted to the native NoC interface using a dedicated finite state machine. The OCP interface cost could be significantly reduced if the NoC Local port directly implemented the OCP protocol, suppressing the native NoC interface. Therefore, router adaptation is not necessary. This option simplifies the design process and can reduce the NI area overhead. This can be justified because fewer states are necessary to packet segmentation and reassembly.

6. Conclusions and Future works

The MAIA CAD framework is useful to generate and evaluate NoCs with varying architectural parameters.

Ongoing work includes: (*i*) SystemC modeling; (*ii*) virtual channels implementation to give support to QoS and deadlock free algorithm in torus topologies; (*iii*) more elaborate statistical traffic models generation.

Future versions of the MAIA framework will include: (*i*) a library of IP blocks that can be configured by the user to enable SoC generation and simulation; (*ii*) a module for enabling more thorough traffic analyses using real traffic, such as video streaming; (*iii*) new NoC architectures, such as irregular meshes of multiple IPs per router.

7. References

[1] Benini L. et al. **Networks on chips: a new SoC paradigm**. IEEE Computer, 35(**1**). 2002, pp. 70-78.

[2] Andriahantenaina, A. et al. **SPIN: a Scalable, Packet Switched, On-Chip Micro-network.** In. DATE'03. 2003, pp. 70-73.

[3] Glass, C. et al. **The Turn Model for Adaptive Routing**. Journal of the Association for Computing Machinery, v. 41(**5**). 1994, pp. 874-902.

[4] Moraes, F. et al. **HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip.** Integration VLSI Journal, 2004. Accepted for publication, in print.

[5] Whelihan, D. **The NOCsim Simulator Users Guide**. 2003. http://www.ece.cmu.edu/~djw2/NOCsim/NOCsim2.3.pdf.

[6] http://www.opnet.com

[7] Xu, J. et al. **A Case Study in Networks-on-Chip Design for Embedded Video**. In: DATE'04. 2004, pp. 770-775.

[8] Sun, Y-R. et al. **Simulation and Evaluation for a Network on Chip Architecture Using Ns-2**, In: 20th IEEE Norchip Conference. 2002.

[9] Kogel, T. et al. **A Modular Simulation Framework for Architectural Exploration of On-Chip Interconnection Networks**. In: CODES/ISSS. 2003, pp. 7- 12.

[10] Chan, J. et al. **NoCGEN: A Template Based Reuse Methodology for Networks on Chip Architecture**. In: VLSI'04. 2004, pp 717-720.

[11] Coppola, M. et al. **OCCN: A Network-On-Chip Modeling and Simulation Framework**. In: DATE'04. 2004, pp. 174-179.

[12] Pestana, S. et al. **Cost-Performance Trade-offs in Networks on Chip:A Simulation-Based Approach**. In: DATE'04. 2004, pp. 764-769.

[13] Venkatraman, V. et al. **NoCIC: A Spice-based Interconnect Planning Tool Emphasizing Aggressive On-Chip Interconnect Circuit Methods**. In: SLIP'04 2004, pp.69- 75.

[14] OCP-IP. **Open Core Protocol Specification – Version 2.0**. Available at: http://www.ocpip.org/.