

Spidergon STNoC Design Flow

Florentine Dubois^{*†}, Jose Cano[‡], Marcello Coppola^{*}, Jose Flich[‡] and Frederic Petrot[†]

[†]TIMA laboratory, CNRS/Grenoble INP/UJF, Grenoble, France

Email: florentine.dubois@imag.fr, frederic.petrot@imag.fr

^{*}ST Microelectronics, Grenoble, France

Email: marcello.coppola@st.com

[‡]Universitat Politècnica de València, Spain

Email: jocare@gap.upv.es, jflich@disca.upv.es

Abstract—In this demonstration we present an enhanced version of the usual Spidergon STNoC design flow. In addition, we show the automatic generation of a simulation platform that can be used to perform early architecture exploration.

I. INTRODUCTION

The term System-on-Chip (SoC) refers to the integration of all the necessary electronic circuits with diverse functionality onto a single chip, in order to obtain a complete electronic system which performs a complex and useful final product functionality. Some advantages derived of using SoC technology are: a) higher performance and system reliability (since all the circuits are on a single chip) and b) lower bill of materials (BOM). SoC development is a complex task as it includes lots of disciplines ranging from circuit design to test engineering. In addition, a modular, component-based approach is required to both hardware and software design.

As technology advances, tens (and in the near future several hundreds) of elements need to be connected within the same chip, thus requiring an efficient on-chip interconnect. The Network-on-Chip (NoC) paradigm is emerging as the solution for interconnecting multiple cores into a single SoC [1]. NoCs are normally characterized by a vast number of architectural parameters, like topology, routing or traffic repartition; finding the optimum solution in term of latency, power or area in this huge set of possibilities is usually called design space exploration [2]. As it is impossible to test every alternative, the platform is modified in an optimization loop, where the designer improves iteratively the system configuration thanks to a set of evaluation methods [3]. However, the later the performance evaluation is performed in the design flow, the more complex the evaluation methods and design modifications become. In order to reduce the optimization loop, the designer can then use high-level network evaluation methods to early explore the design space at low-cost, thus decreasing the overall SoC's time-to-market.

This demonstration presents an enhanced version of the usual ST Microelectronics Spidergon STNoC hardware design methodology, which can be directly included in a more general SoC design flow. This flow guides in a fast and efficient way designer's choices in the design space thanks to several performance evaluations at different levels of abstraction. It is inspired of the well-known Y-chart methodology, which is an iterative design strategy based on the separation of the application from the architecture [4], [5].

The organization of the paper is the following: after presenting the NoC design flow's state of the art, we will give an overview of the Spidergon STNoC technology, before describing step by step our flow proposal and concluding.

II. STATE OF THE ART

Whereas mature SoC design tools already exist, NoCs design flow and their integration in SoC synthesis is still an open research area. An example of tool is the commercial one proposed by the company Arteris [6], which use a flow similar to our proposal. They explore the architectural possibilities and validate the network functionalities with both

abstract network models and low-level simulations. The main difference with our work is that their validation steps do not use pure mathematical models to provide early performance estimations at very low modeling cost. Goossens et al. [7] and Bolotin et al. [8] propose similar design flows based on three main steps: NoC generation (topology, mapping), NoC configuration (routing, QoS, etc) and finally NoC verification, based on mathematical models and/or simulations. However, neither of those works use intermediate abstract models to further explore the design space at lower cost than simulations and with more accuracy than pure theoretical models. Others Network-on-Chip generation flows can be found in [9], [10].

III. SPIDERGON STNoC TECHNOLOGY OVERVIEW

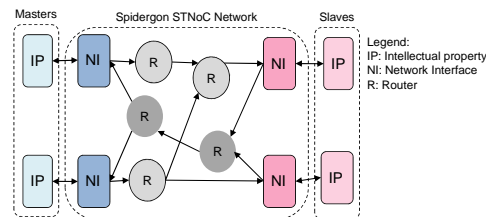


Fig. 1. Spidergon STNoC example

The Spidergon STNoC interconnect is a flexible Network-on-Chip technology developed by ST Microelectronics [11]; a system example is given in figure 1. It is composed of three different types of building blocks, appropriately interconnected to each other, and provides a set of customizable low-level platform services along with a set of communication primitives. The key building blocks are:

- The network interface (NI), which interfaces IP blocks and the network,
- The router, which is responsible for determining the next network point to which a packet/flit should be forwarded toward its destination;
- The physical link, which is a logical component responsible for the physical communications.

IV. SPIDERGON STNoC DESIGN FLOW

The enhanced Spidergon STNoC design flow proposed in this demonstration is schematized in Figure 2. It is a direct application of the Y-chart design methodology ([4], [5]) to the field of NoCs and can be easily included in a general SoC design strategy. The system is optimized iteratively; designer's choices are based on a set of evaluation steps, going in ascending modeling complexity order, thus lowering the speed of the evaluation but improving its accuracy. In high-level steps, the designer has a great freedom in the design space exploration, whereas low-level steps limits the optimization possibilities to neighbourhood configurations, due to design modification costs. The designer can then explore and greatly refine the possibilities during first steps, reducing the number of time-consuming low-level validations, resulting in a fast and effective design space exploration. For an overview of existing NoCs evaluation metrics, see [12]. The remainder of

this section is devoted to the description of the different steps, which are all included in the Spidergon STNoC design tool. The demonstration will focus on the high-level steps (steps A,B,C) and their associated model generations.

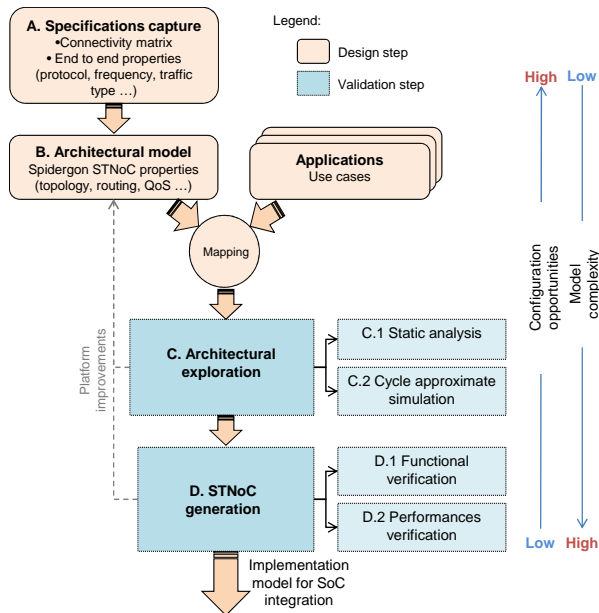


Fig. 2. Spidergon STNoC design flow

A. First step: Specifications capture

This step is responsible for the end-to-end characterization: only the IP properties are defined and the network is considered as a black-box. Properties chosen in this phase are:

- the IP communication properties, for example protocols (AXI, STBus, etc), working frequencies or data widths;
- The connectivity matrix, which defines which pairs of IPs communicate together and the associated flow properties: required bandwidths, traffic types (Real-time, Best-Effort), etc.

B. Second Step: Architectural model

This step is responsible for capturing the SoC requirements which have direct implications on different network features. The designer defines here the network topology, routing, power and frequency domains, etc.

C. Architecture Exploration

This evaluation step follows the mapping of a set of use-cases, based on the different target applications, on the current NoC (mapping step of figure 2), according to the Y-chart methodology main idea.

The architecture exploration step is composed of two high-level evaluation methods described below. They provide early estimations of network performances and give a good overview of the different configuration possibilities with a minimum amount of effort. All models required here are automatically generated by the Spidergon STNoC design tool.

1) *Static analysis*: Static analysis is based on analytical models of the network and provide pseudo-realistic performance evaluations at very low modeling cost. Even though the precision of those metrics depends on the mathematical methods used, they are very useful to explore quickly the design space. Moreover, those methods can also detect configuration errors to decrease the number of bugs at low-level steps, and thus greatly reduce the time-to-market.

2) *Cycle approximate simulation*: This second method is based on the free simulation library OMNeT++ [13], and uses an abstract model to compute cycle-approximate simulations of the network. The designer can here evaluate very quickly network latencies and bandwidths with more precision than with static analysis and a small modeling effort.

D. Fourth step: STNoC generation

The flow's final step is low-level explorations and verifications that provides highly accurate but computationally expensive network performance estimations. Few simulations are required here thanks to previous steps and the designer can thus quickly choose the final configuration inside the reduced design space. This step is composed of two parts described below and its output is an implementation model of the platform (RTL).

1) *Functional verification*: This step is responsible for checking functional properties of the different actors in the system. It is based on a set of general predefined tests which validates the components behaviour at RTL level.

2) *Performance verification*: Platform performance is evaluated in an IPTG environment [14], where the SoC architecture is modeled either at the transactional or at RTL level. The designer can then optimize the design until it meets the initial constraints with correct performance, before generating the final NoC's model.

V. CONCLUSION

An enhanced Spidergon STNoC design flow was presented. This flow is based on the Y-chart methodology and can be included in a general SoC design method. It is based on two evaluation steps which guide the designer in the optimization loop; those evaluations are first located at a high level of abstraction so the exploration areas can be greatly refined before computing time-consuming simulations to choose the final configuration.

ACKNOWLEDGMENT

This work was supported by the Spanish MEC and MICINN, as well as European Commission FEDER funds, under Grant CSD2006-00046. It was also partly supported by the COMCAS project (CA501), a project labelled within the framework of CATRENE, the EUREKA cluster for Application and Technology Research in Europe on NanoElectronics.

REFERENCES

- [1] L. Benini *et al.*, "Networks on chips: A new soc paradigm," *Computer*, vol. 35, pp. 70–78, 2002.
- [2] P. P. Pande *et al.*, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *Computers, IEEE Transactions on*, vol. 54, no. 8, pp. 1025 – 1040, Aug. 2005.
- [3] U. Ogras *et al.*, "Analytical router modeling for networks-on-chip performance analysis," in *Design, Automation Test in Europe Conference Exhibition, 2007. DATE '07, 2007*, pp. 1–6.
- [4] A. Jantsch *et al.*, *Networks on chip*. Kluwer Academic Publishers, 2003.
- [5] S. Khan *et al.*, "From y-chart to seamless integration of application design and performance simulation," in *System on Chip (SoC), 2010 International Symposium on*, 2010, pp. 18–25.
- [6] Arteris. [Online]. Available: <http://www.arteris.com>
- [7] K. Goossens *et al.*, "A design flow for application-specific networks on chip with guaranteed performance to accelerate soc design and verification," in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 2, ser. DATE '05, 2005*, pp. 1182–1187.
- [8] E. Bolotin *et al.*, "Qnoc: Qos architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, pp. 105–128, February 2004.
- [9] J. Chan *et al.*, "Noegen: a template based reuse methodology for networks on chip architecture," in *VLSI Design, 2004. Proceedings. 17th International Conference on*, 2004, pp. 717–720.
- [10] S. Kumar *et al.*, "A network on chip architecture and design methodology," *VLSI, IEEE Computer Society Annual Symposium on*, 2002.
- [11] M. Coppola *et al.*, *Design of Cost-Efficient Interconnect Processing Units: Spidergon STNoC*. CRC Press, Inc., 2008.
- [12] E. Salmien *et al.*, "On network-on-chip comparison," in *Digital System Design Architectures, Methods and Tools, 2007. DSD 2007. 10th Euromicro Conference on*, 2007, pp. 503–510.
- [13] Omnet++. [Online]. Available: <http://www.omnetpp.org/>
- [14] A. Perrin *et al.*, "Architecture analysis and system debugging," in *Transaction Level Modeling with SystemC*. Springer US, 2005, pp. 207–240.