Application Specific NoC Design

Luca Benini
DEIS Universitá di Bologna
lbenini@deis.unibo.it

Abstract

Scalable Networks on Chips (NoCs) are needed to match the ever-increasing communication demands of large-scale Multi-Processor Systems-on-chip (MPSoCs) for high-end wireless communications applications. The heterogeneous nature of on-chip cores, and the energy efficiency requirements typical of wireless communications call for application-specific NoCs which eliminate much of the overheads connected with general-purpose communication architectures. However, application-specific NoCs must be supported by adequate design flows to reduce design time and effort.

In this paper we survey the main challenges in application-specific NoC design, and we outline a complete NoC design flow and methodology. A case study on a high complexity SoC demonstrates that it is indeed possible to generate an application-specific NoC from a high level specification in a few hours. Comparison with a hand-tuned solution shows that the automatically generated one is very competitive from the area, performance and power viewpoint, while design time is reduced from days to hours.

Keywords: Systems on chip, networks on chip, application-specific integrated systems, design methodologies

1 Introduction

The number of processor, memory and accelerator cores on systems on chip for wireless communications is rapidly increasing to support evolving standards and new applications. Computation and communication complexity is skyrocketing, and scalability-centric design paradigms are critically needed. *Networks on Chip (NoCs)* have emerged as a viable option for designing scalable communication architectures for MPSoCs [5], [6]. In NoCs, on-chip micro-networks are used to interconnect the various cores. NoCs have better modularity and design predictability when compared to traditional bus-based systems, and they are viewed by many as a strategic technology for addressing communication issues in current and future nanometer-scale ICs.

The challenges encountered in NoC design differ considerably from those encountered in macro-networks. First, the communication patterns can be statically analyzed for many SoCs and the NoC can be tailored for the particular application behavior. Second, design objectives and constraints are different. As most wireless communication chips are meant to operate in energy-constrained environments, network energy efficiency is an important design objective. These systems are also subject to tight real-time requirements and must operate in a highly predictable fashion, hence quality-of-service oriented, predictable interconnects are highly desirable. Finally, the design process should also consider VLSI issues, such as the structure and wiring complexity of the resulting interconnect.

Some of the most important phases in designing the NoC are the design of the topology or structure of the network and setting of various design parameters (such as frequency of operation or link-width). Several early works favored the use of standard topologies under the assumption that the wires can be well structured in such topologies. These approaches are adequate for general-purpose systems where the traffic characteristics of the system cannot be predicted statically, as in homogeneous chip-multiprocessors [1]. However, most SoCs are heterogeneous, with each core having different size, functionality and communication requirements. Thus, standard topologies can have a structure that poorly matches application traffic. This leads to large wiring complexity after floorplanning, as well as significant power and area overhead. Moreover, for most SoCs the system is designed with static (or semi-static) mapping of tasks to processors and hardware cores and hence the communication traffic characteristics of the SoC are well characterized at design time. This is true for a number of SoC platforms for wireless and multimedia, such as the Philips Nexperia platform [2], ST Nomadik [3], TI OMAP [4], etc.

On the other hand, SoC platforms are designed under ever-increasing time-to-market pressure, and simply there is no time for handcrafting a system interconnect on every platform instantiation. What is ultimately needed is then

3-9810801-0-6/DATE06 © 2006 EDAA

a methodology to automatically design the best topology that is tailor-made for a specific application and satisfies the communication constraints of the design. The topology design process should support multiple objective functions, such as minimizing power or hop delay. The topology design process also needs to support constraints on several parameters such as design area and total wire-length. The topology synthesis process is should embed a floorplanner to estimate the design area and wire-lengths. The wire-length estimates from the floorplan can be used to evaluate whether the designed NoC satisfies the target frequency of operation and to compute the power consumption of the wires. As deadlock-free routing is critical for proper operation of custom topologies, topology generation must also integrate algorithms that guarantee deadlock-free deliver of packets.

Topology generation is only the first step of a complete design flow for parameter tuning (e.g. buffer depth, clock frequency), instantiation, synthesis, simulation/emulation and physical design of the NoC. Practical flows should also support manual intervention, if needed, at several levels (like manually setting up frequency, link-width). Even though NoCs have been investigated intensively for a relatively short time (approximatively five years), significant progress has been obtained, and a few automatic network design flow have reached a good level of maturity [9, 7].

In the following section we will first give an overview of current NoC synthesis research efforts. We then describe the $\times$pipes design flow, as an example of one of the first attempts to provide a complete application-to-layout flow for application-specific NoCs. Finally, we focus on a case study of interconnect design for a complex multi-core SoC, which provides quantitative evidence of the advantages of a complete synthesis based design flow.

2 NoC synthesis: the landscape

In recent years, a large body of research has focused on synthesizing and generating bus-based systems [10]-[16]. A floorplan aware point-to-point link design and bus design are presented in [17] and [16]. While some of the design issues in the NoCs are similar to bus based systems (such as link-width sizing), a large number of issues such as finding the number of switches required, sizing the switches, finding routes for packets, etc. are new in NoCs. A good overview of NoC synthesis problems is given by Marculescu et al. [13].

Methods to collect and analyze traffic information that can be fed as input to the bus and NoC design processes have been presented in [14] and [15]. Mappings of cores onto standard NoC topologies have been explored in [18]-[21]. In [19], [21] a floorplanner is used during the mapping process to get area and wire-length estimates. These works only select from a library of standard topologies, and cannot generate a fully customized topology. In [20], a unified approach to mapping, routing and resource reservation has been presented. However, the work does not explore the topology design process. Important research in macro-networks has considered the topology generation problem [22]. As the traffic patterns on these networks are difficult to predict, most approaches are tree-based (like spanning or Steiner trees) and only ensure connectivity with node degree constraints [22]. Hence, these techniques cannot be directly extended to address the NoC synthesis problem.

Application-specific custom topology design has been explored in [23]- [26]. The works from [23], [24], do not consider the floorplanning information during the topology design process. In [25], a physical planner is used during topology design to reduce power consumption on wires. However, it does not consider the area and power consumption of switches in the design. Also, the number and size of network partitions are manually fed. In [26], a slicing tree based floorplanner is used during the topology design process. This work assumes that the switches are located at the corners of the cores and it does not consider the network components (switches, network interfaces) during the floorplanning process. Also, the actual sizes of the cores are considered only after generating their relative positions. The resulting floorplan can be extremely area inefficient when compared to the standard floorplanning process. Also, deadlock free routing, which is critical for custom NoC designs is not supported. Moreover, a complete design space exploration, from architectural parameter setting to simulation is not presented. ored in [23]- [26]. The works from [23], [24], do not consider the floorplanning information during the topology design process. In [25], a physical planner is used during topology design to reduce power consumption on wires. However, it does not consider the area and power consumption of switches in the design. Also, the number and size of network partitions are manually fed. In [26], a slicing tree based floorplanner is used during the topology design process. This work assumes that the switches are located at the corners of the cores and it does not consider the network components (switches, network interfaces) during the floorplanning process. Also, the actual sizes of the cores are considered only after generating their relative positions. The resulting floorplan can be extremely area inefficient when compared to the standard floorplanning process. Also, deadlock free routing, which is critical for custom NoC designs is not supported. Moreover, a complete design space exploration, from architectural parameter setting to simulation is not presented.

Topology generation is the front end of a complete design flow that spans multiple abstraction levels, down to placement and routing. The back-end of the flow is more mature at the present time. In fact, several works have been

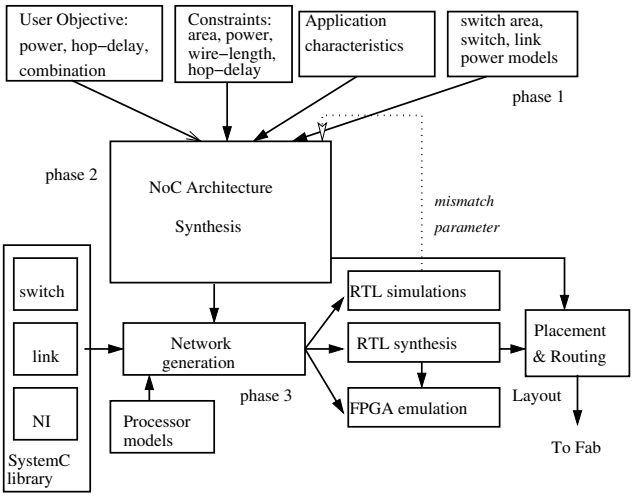


Figure 1. NoC Design Flow

published on automatically generating the *Register Transfer Level (RTL)* code of a designed topology for simulation and synthesis [27]-[29]. Building area, power models for on-chip networks have been addressed in [30]-[33]. These approaches have rapidly reached the industrial practice. Several companies are today productizing synthesizable NoC architectures [34, 35, 36]. At the research and development frontier, the focus is now on bridging the gap between high-level analysis of communication requirements and synthesis of a NoC implementation for a given topology.

3 ×pipes NoC Synthesis Flow

×pipes is an example of a complete flow for designing NoCs, as shown in Figure 1. In the first phase, the user specifies the objectives and constraints that should be satisfied by the designed NoC. The application traffic characteristics, size of the cores, and the area and power models for the network components are also obtained.

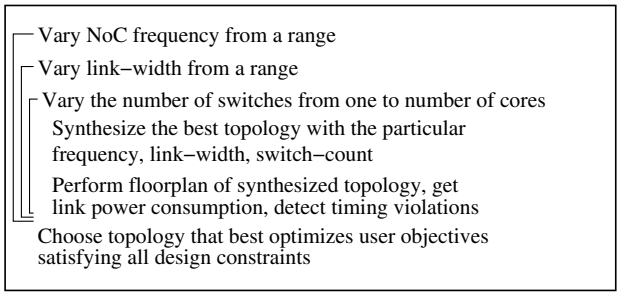


Figure 2. Topology synthesis iterations

In the second phase of the flow, the NoC architecture that optimizes the user objectives and satisfies the design constraints is automatically synthesized. The different steps in this phase are presented in Figure 2. In the outer iterations, the key NoC architectural parameters: NoC frequency of operation and link-width are varied from a range of suitable values. The bandwidth available on each NoC link is the product of the NoC frequency and link-width. During the topology synthesis, it is ensured that the traffic on each link is less than or equal to its available bandwidth value.

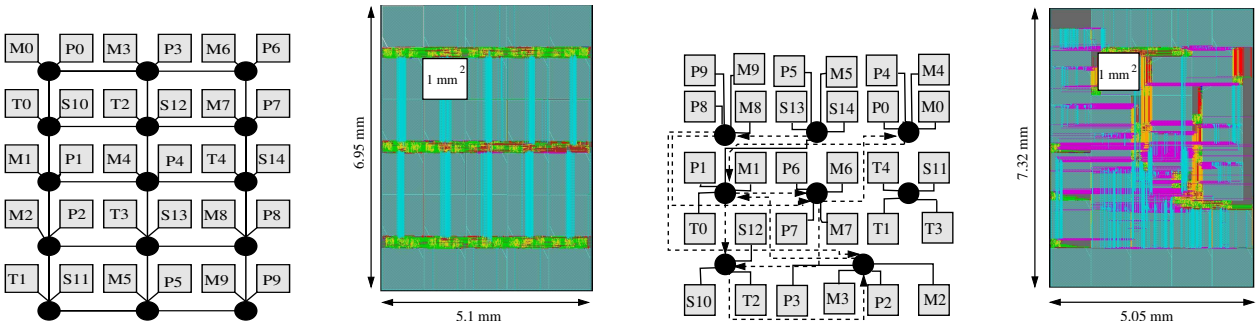
The synthesis step is performed once for each set of the architectural parameters. In this step, several topologies with different number of switches, starting from a topology where all the cores are connected to one switch, to the topology where each core is connected to a separate switch are explored. The synthesis of each topology includes finding the size of the switches, establishing the connectivity between the switches and connectivity with the cores and finding deadlock-free routes for the different traffic flows (using the approach outlined in [37]).

In the next step, to have an accurate estimate of the design area and wire-lengths, the floorplanning of each synthesized topology is automatically performed (using the tool described in [38]). The floorplanning process finds the 2D position of the cores and network components used in the design. Based on the frequency point and the obtained wire-lengths, the timing violations on the wires are detected and power consumption on the links is obtained. In the last step, from the set of all synthesized topologies and architectural parameter design points, the topology and the architectural point that best optimizes the user's objectives, satisfying all the design constraints is chosen. Thus, the output of phase 2 is the best application-specific NoC topology, its frequency of operation and the width of each link in the NoC.

In the last phase of the design (phase 3 in Figure 1), the RTL (SystemC) code of the switches, network interfaces and links for the designed topology are automatically generated. For this, we use ×pipesLite [9], a library of soft macros for the network components and the associated tool [27] to interconnect the network elements with the cores. At this phase, we also obtain synthesizable RTL design (that can also be emulated on FPGA). From the floorplan specification of the designed topology, the synthesis engine automatically generates the inputs for placement&routing. The placement&routing of the design is performed using SoC Encounter [39] for obtaining the layout, including the global and detailed routing of wires. The output of this phase is a complete layout of the NoC design that can be sent to fabrication.

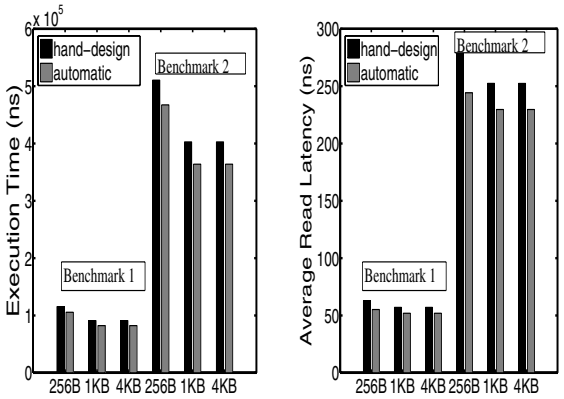
4 Case Study

In this section we test the effectiveness of the topology synthesis approach outlined in the previous section against a



(a) Hand-designed topology (b) Layout (c) Automatically synthesized (d) Layout

Figure 4. (a), (b) Hand-designed topology and its layout. M: ARM7 processors, T: traffic generators, P, S: private and shared slaves (c), (d) Automatically synthesized topology and its layout. In Figure (c), bi-directional links are dark and uni-directional links are dotted.



(a) Execution time (b) Average read latency

Figure 3. Performance vs. cache sizes

hand-designed NoC architecture for a SoC design that runs multi-media application benchmarks. The design consists of 30 cores: 10 ARM7 processors with caches, 10 private memories (a separate memory for each processor), 5 custom traffic generators, 5 shared memories and devices to support interprocessor communication. The hand-designed NoC has 15 switches connected in a 5x3 mesh network, shown in Figure 4(a). The design is highly optimized, with the private memories being connected to the processors using a single switch and the shared memories distributed around the switches. The layout of the design (presented in Figure 4(b)) was performed using SoC Encounter and the mesh structure was maintained in the layout. Each of the cores has an area of 1 mm^2 in the design. The entire process, from topology specification to layout generation took several weeks. The post-layout NoC could support a maximum frequency of operation of 885 MHz and the power consumption of the topology for functional traffic is 368.08 mW.

We apply the topology synthesis process with the objective of minimizing power consumption, to automatically design the NoC for this SoC design. We set the design constraints and the required frequency of operation to be the same (885 MHz) as that of the hand-designed topology. The designed NoC topology and the layout obtained using SoC Encounter are presented in Figures 4(c) and 4(d). The synthesized topology has fewer switches (8 switches) than the hand-designed topology. But, the switch-to-switch wires are longer (up to $2\times$ longer) in the synthesized topology. However, it could support the same maximum frequency of operation (885 MHz) as that of the hand-designed topology, without any timing violations on the wires. As we considered the wire-lengths during the synthesis process to estimate the frequency that can be supported, we could synthesize the most power efficient topology that would still meet the target frequency. To arrive at such a design point manually would require several iterations of topology design and place&route phases, which is a very time consuming process.

Layout level power consumption calculations on functional traffic shows that the synthesized topology has 277.08 mW power consumption, which is $1.33\times$ lower than the hand-designed topology. Given the fact that the hand-designed topology is highly optimized, with much of the communicating traffic (which is between the ARM cores and their private memories) traversing only one switch, this savings is achieved entirely from efficiently spreading the shared memories around the different switches. The layout of the hand-designed NoC was manually optimized to a large extent (by moving switches, network interfaces) to reduce the area of the design. The layout of the synthesized topology is obtained completely automatically, and still the

area of the design is close to that of the manual design (only a marginal 4.3% increase in area).

We perform cycle-accurate simulations of the hand-designed and the synthesized NoCs for two multimedia benchmarks. The total application time for the benchmarks (including computation time) and the average packet latencies for read transactions for the topologies are presented in Figures 3(a) and 3(b). The custom topology not only matches the performance of the hand-designed topology, but provides an average of 10% reduction in total execution time and 11.3% reduction in average packet latency.

5 Conclusions

Synthesizing application-specific NoC architectures is a non-trivial task, given the large design space that needs to be explored. In this work, we have outlined a methodology that automates the process, generating efficient NoCs that satisfy the design constraints of the application. We have presented a case study with the comparison between a hand-crafted and a synthesized topology. Our results (fully validated post-placement and routing) demonstrate that NoC synthesis can achieve competitive quality at a fraction of the design time: manual topology design took a couple of weeks while the automatic flow required less than four hours to complete.

6 Acknowledgments

The author wishes to thank his co-workers at Stanford University (Srinivasan Murali), University of Bologna (Federico Angiolini, Davide Bertozzi), University of Cagliari (Salvatore Carta, Paolo Meloni, Luigi Raffo), EPFL (Giovanni De Micheli, David Atienza). The financial support of STMicrolectronics and SRC (contract 1188) is gratefully acknowledged.

References

[1] M. Taylor, W. Lee, S. Amarasinghe, A. Agarwal, "Scalar operand networks ", IEEE Transactions on Parallel and Distributed Systems, vol. 16, no. 2, pp. 145-162, Feb 2005.

[2] S. Dutta et al., "Viper: A Multiprocessor SOC for Advanced Set-Top Box and Digital TV Systems", IEEE Design and Test of Computers,Sep/Oct 2001, pp. 21-31.

[3] A. Artieri et al., "Nomadik Open Multimedia Platform for Next-generation Mobile Devices", STMicroelectronics Technical Article TA305, 2003, available at http://www.st.com

[4] J. Helmig, "Developing core software technologies for TI's OMAPTM platform", Texas Instruments, 2002. Available at http://www.ti.com.

[5] L.Benini and G.De Micheli, "Networks on Chips: A New SoC Paradigm", IEEE Computers, pp. 70-78, Jan. 2002.

[6] D.Wingard,"MicroNetwork-Based Integration for SoCs", Design Automation Conference DAC 2001, pp. 673-677, Jun 2001.

[7] K. Goossens et al., "A Design Flow for Application-Specific Networks on Chip with Guaranteed Performance to Accelerate SOC Design and Verification", DATE 2005.

[8] D. Bertozzi et al., "NoC Synthesis Flow for Customized Domain Specific Multi-Processor Systems-on-Chip", IEEE Transactions on Parallel and Distributed Systems, Feb 2005.

[9] S. Stergiou et al., "×pipesLite: a Synthesis Oriented Design Library for Networks on Chips", pp. 1188-1193, Proc. DATE 2005.

[10] J. Daveau et al., "Synthesis of system-level communication by an allocation based approach", Proc. ISSS, pp. 150-155, Sept. 1995.

[11] M. Gasteier, M. Glesner, "Bus-based communication synthesis on system level", ACM TODAES, vol.4, no.1, pp. 1-11, 1999.

[12] K. Ryu, V. Mooney, "Automated Bus Generation for Multiprocessor SoC Design", Proc. DATE, pp. 282-287, March 2003.

[13] U. Ogras, J. Hu, R. Marculescu, "Key research problems in NoC design: a holistic perspective", IEEE CODES+ISSS pp. 69-74, 2005.

[14] K.Lahiri et al., "Design Space Exploration for Optimizing On-Chip Communication Architectures", IEEE TCAD, vol.23, no.6, pp. 952- 961, June 2004.

[15] S. Murali, G. De Micheli, "An Application-Specific Design Methodology for STbus Crossbar Generation", pp. 1176-1181, Proc. DATE '05.

[16] S. Pasricha et al., "Floorplan-aware automated synthesis of bus-based communication architectures", Proc. DAC '05.

[17] J. Hu et al., "System-Level Point-to-Point Communication Synthesis Using Floorplanning Information", Proc. ASPDAC '02.

[18] J. Hu, R. Marculescu, 'Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures', Proc. DATE, March 2003.

[19] S. Murali, G. De Micheli, "SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs", Proc. DAC 2004.

[20] A. Hansson et al., "A unified approach to constrained mapping and routing on network-on-chip architectures", pp. 75-80, Proc. ISSS 2005.

[21] S. Murali et al., "Mapping and Physical Planning of Networks on Chip Architectures with Quality-of-Service Guarantees", Proc. ASPDAC 2005.

[22] R. Ravi et al., "Approximation algorithms for degree-constrained minimum-cost network design problems", Algorithmica, 31(1): 58-78, 2001.

[23] A.Pinto et al., "Efficient Synthesis of Networks on Chip", ICCD 2003, pp. 146-150, Oct 2003.

[24] W.H.Ho, T.M.Pinkston, "A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns", HPCA 2003, pp. 377-388, Feb 2003.

[25] T. Ahonen et al. "Topology Optimization for Application Specific Networks on Chip", Proc. SLIP 04.

[26] K. Srinivasan et al., "An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks", Proc. ICCAD '05.

[27] A. Jalabert et al., "×pipesCompiler: A tool for instantiating application specific networks-on-chip", pp. 884-889, Proc. DATE 2005.

[28] D.Siguenza-Tortosa, J. Nurmi, "Proteo: A New Approach to Network-on-Chip", in CSN 02, Sep. 2002.

[29] X.Zhu, S.Malik, "A Hierarchical Modeling Framework for On-Chip Communication Architectures", ICCD 2002, pp. 663-671, Nov 2002.

[30] T. T. Ye et al., "Analysis of power consumption on switch fabrics in network routers", Proc. DAC '03.

[31] H-S Wang et al., "Orion: A Power-Performance Simulator for Interconnection Network", Proc. International Symposium on Microarchitecture, , Nov 2002.

[32] N. Banerjee et al., "A power and performance model for network-on-chip architectures", Proc. DATE '04.

[33] G. Palemoro, C. Silvano, "PIRATE: A Framework for Power/Performance Exploration of Network-On-Chip Architectures", PATMOS 2004

[34] www.arteris.net

[35] M. Coppola et al., "Spidergon: a novel on-chip communication network", ISSOC pp. 16-18, 2004.

[36] J. Bainbridge, S. Furber, "Chain: a delay-insensitive chip area interconnect", IEEE Micro, vol. 22, no. 5, pp. 16-23, Sept.-Oct. 2002.

[37] D. Starobinksi et al., "Application of network calculus to general topologies using turn-prohibition", IEEE/ACM Transactions on Networking, Vol. 11, Issue 3, pp. 411-421, June 2003.

[38] S. N. Adya, I. L. Markov, "Fixed-outline Floorplanning : Enabling Hierarchical Design", IEEE Trans. on VLSI Systems, vol 11(6), pp. 1120-1135, Dec 2003. URL: http://vlsicad.eecs.umich.edu/BK/parquet/

[39] www.cadence.com