Networks on Chips: from Research to Products

G. De Micheli[⋆], C. Seiculescu[⋆], S. Murali[§] [⋆], L. Benini[‡], F. Angiolini[§] , A. Pullini[§]

[⋆] LSI, EPFL, Lausanne, Switzerland,{ciprian.seiculescu, giovanni.demicheli}@epfl.ch
[§] iNoCs, Lausanne, Switzerland, {murali, angiolini, pullini}@inocs.com
[‡] DEIS, University of Bologna, Bologna, Italy, luca.benini@unibo.it

ABSTRACT

Research on *Networks on Chips* (NoCs) has spanned over a decade and its results are now visible in some products. Thus the seminal idea of using networking technology to address the chip-level interconnect problem has been shown to be correct. Moreover, as technology scales down in geometry and chips scale up in complexity, NoCs become the essential element to achieve the desired levels of performance and quality of service while curbing power consumption levels. Design and timing closure can only be achieved by a sophisticated set of tools that address NoC synthesis, optimization and validation.

Categories and Subject Descriptors

B.4.3 [**INPUT/OUTPUT AND DATA COMMUNICATIONS**]: Interconnections (Subsystems)—*topology*

General Terms

Design

Keywords

Network on Chip, NoC, System on Chip, SoC

1. INTRODUCTION

A key challenge in the design of multi-core chips is the choice of a scalable system-level interconnect. This issue surfaces in both general-purpose multi-cores, i.e., *Chip Multi-Processors (CMPs)*, as well as in heterogeneous, application-specific *Multi-Processor Systems on Chips (MPSoCs)*. A similar challenge affects the design of the next generation of *Field-Programmable Gate Arrays* (FPGAs).

The importance of interconnects for system performance is growing with technology scaling. An increasingly large number of on-chip cores with continuously improving performance can be found in designs such as TI's OMAP [30], Infineon XMM/X-Gold [31] or ST's Nomadik [29]: a mobile phone SoC nowadays comprises several tens to hundreds of components that need to be connected together. Thus, application bandwidth requirements have also been increasing steadily. Furthermore, at the physical level, with technology scaling, gate delays decrease while global wire delays do not. Thus, in current advanced technologies the delay on the wires has an increasingly significant impact on system performance.

For a long while, bus-based solutions have been widely used to connect components inside chips. As the number of components and their complexity scales up, the complexity of the bus system also increases. Bus architectures have evolved significantly, with designers migrating from a single shared-bus topology to bridged buses and to multilayer buses. Today a complex SoC can have several levels of bus hierarchy. The protocol complexity has also increased, with support for burst, outstanding, out-of order transactions, just to give some examples.

One of the most critical design issues with bus-based systems is that it is getting increasingly hard to achieve *design closure*. The use of several levels of bus hierarchies, interconnected by means of bridges and crossbars in an *ad hoc* manner is difficult to design and even more challenging to verify. Moreover, it is hard to predict the length and the delay of the wires of the bus within the architectural design phase. Accurate wire delay estimates are possible only at the end of the physical design (placement and routing) phase, and any delay violations lead to costly design iterations.

The use of networking principles to connect components alleviates some of the major issues with bus-based systems. There are several advantages in adopting a *Network on Chip* (NoC) paradigm for chip-level interconnects. First and foremost, the modularity of NoCs is a key asset in supporting scalability from the ground up, in particular in terms of performance. Physical-design-aware NoC components enable large-scale *System on Chip* (SoC) design with more predictable (and possibly guaranteed) performance. Second, NoCs can be tuned to support specific applications on SoCs. Whereas macroscopic networks (e.g., LANs, WANs) emphasize general-purpose communication schemes and modularity, in NoCs these constraints are less restrictive because, in most cases, on-chip communication is known at design time. Thus NoC implementations may be optimized, e.g., by merging or separating data and control traffic, or using *ad-hoc* bus widths and flow control schemes. This flexibility enables CAD tools to explore the power/performance design space and provide designers with effective solutions. Finally, the distributed nature of NoC infrastructures can be effectively leveraged to enhance system-level reliability. For example, NoCs can locally handle at run-time the correction of timing failures induced by variability and/or other signal integrity issues. Moreover, reconfigurable NoCs can support component redundancy in a transparent fashion, thus being an essential technology for designing highly-dependable systems. The key challenge in bringing networking means to silicon is to suitably adapt the principles to the chip medium, and to achieve a low-latency interconnect with low power consumption and area overhead.

Many SoCs have been realized with NoC-based interconnects in the recent years. Examples range from a multitude of designs based on the ARM AMBA Network Interconnect [32], which closely resembles a NoC due to their multi-layered nature, to Intel's Teraflop Research Chip (also called it Polaris), a multi-core processor [37], and the new Single-chip Cloud Computer [38]. Tilera markets the TILE-Gx [27], a 100 core processor, which is the commercial spin-off of research done on the RAW architecture [39] at MIT. Other research SoCs include LETI's FAUST [25] and KAIST's BONE series [41]. Whereas some of these chips will be reviewed in Section 5, it is important to stress that NoCs are present in commercial products and that top semiconductor manufacturers have invested in NoC design methods and tools.

Electronic Design Automation (EDA) tools for NoC design and optimization have been the object of intensive research in the last decade. A few providers of NoCs have shown to support a flow and to yield cost-effective designs. As NoCs become mainstream, the related tools will be an important value-added part to standard design flows. Indeed, one of the biggest advantages of NoCs is the ease of design, optimization and verification, and thus faster design times that NoC design automation can bring. With the use of appropriate tool flows, the NoC operating frequency can be predicted accurately already during architectural design, accounting for the impact of floorplan on wire lengths. Thus, fewer (or no) iterations are needed across the architectural and physical design phase for convergence. The NoC can be finely tuned and optimized, and its power consumption can be evaluated and reduced. Other advantages include scalability to newer SoC platforms, ability to support varied application *Quality-of-Service* (QoS) constraints, better support for voltage and frequency islands, *etc.*.

2. HISTORY OF NOCS

Design issues in macro-networks (LAN, WAN, Internet) have received broad attention in the last twenty five years. In the last decade, the design of chip-to-chip interconnection networks for parallel processing has also received considerable focus. However, the challenges encountered in the design of on-chip networks for SoCs is quite different from the design of such macroscopic networks. Some major differences are the following three. (1)The communication between the various cores can be statically analyzed for many SoCs, so that the NoC can be tailored for the particular application behavior. In the case of macroscopic networks, it is impossible to obtain an upfront knowledge of the traffic patterns of all the users. (2) The design objectives and constraints are different. As many SoCs are used in mobile and hand-held devices, having a network with minimum power consumption becomes an important design objective. Many SoCs also need to respond in real-time for certain inputs, for which the NoC has to support different criticality levels for the different traffic streams. Area and latency constraints are also much more stringent for NoCs. (3) The design process should also consider VLSI issues, such as the structure (floorplan requirements) and wiring complexity of the resulting interconnect.

The use of packet-switched networks to connect components inside a computing system was advocated first by Seitz and co-workers [1]. Nevertheless, the concept of Networks on Chip was pioneered by Greiner within the *Scalable Programmable Integrated Network (SPIN)* project [3] and elaborated in its various facets by Benini et al. [2], and Dally et al. [5], [7] in the early part of this decade. Thereafter, there has been a flurry of contributions addressing the different design issues of NoCs. We present a non-exhaustive summary of some important contributions, while referring the author to [4] for a detailed description of the literature.

Early examples of chip realizations involving NoCs or precursors include the following. The Maia heterogeneous signal processing architecture [6] is fully instance-specific and uses circuit switching to route data. The RAW architecture [39] uses a mesh topology to support general-purpose parallel multiprocessing and introduces the notion of exposing the wiring architecture to the compiler. The SPIN project described in [3] is an early example of a NoC architecture, with the use of a regular, fat-tree-based network.

Many NoC architectures have been proposed. A large fraction of these are natively synchronous, such as $\times$pipes [44], NOSTRUM [10], Spidergon [22]; others are conceived to support asynchronous operation, such as Mango [26], FAUST [25] and ANOC [23]. Some architectures were presented to achieve predictable QoS behavior, using special hardware mechanisms, such as Æthereal [21], QNoC [20]. We show some examples of NoC architectures in Section 3.

Several research groups have focused on design automation for NoCs. The issues include routing strategy development, topology synthesis, QoS achievement, buffer sizing. Initial works on topology design focused on mapping cores onto regular topologies [8], [9]. The research and development of $\times$pipesCompiler [45] and followup tools [11] addressed both the support of heterogeneous NoC topologies, tailored to the application traffic requirements, and the corresponding design tool flow. It strongly differentiated from earlier approaches that were targeting only standard topologies, such as meshes, as these do not map well to SoCs that are usually heterogeneous in nature. In this work, both a parametrized library $\times$pipes and a NoC hardware compiler were presented to address the problem of synthesis and optimization for heterogeneous NoCs by means of highly-configurable network building blocks, customizable at instantiation time for a specific application domain. This work showed the advantages of customizing NoC resources, such as switches, and the need to consider the floorplan of the chip when designing the NoC.

While the need for custom topologies became evident at this point, synthesizing them automatically to meet application specifications considering physical design issues still remained a challenge. Topology synthesis needs to account for several, often conflicting, objectives such as reducing latency and power consumption. It should also consider a variety of network operational issues. For example, the synthesized topologies should be free of routing and message-dependent deadlocks. Moreover, the wire lengths should also be considered when calculating the power and delay values of topologies. This motivated the work of SunFloor [11], where a synthesis tool was presented to design application specific custom topologies. The tool has several key features, including a method to gradually drive the synthesis process to meet conflicting objectives and constraints and the ability to explore a large design space, producing several design points with different power-performance values. Another highlight of this work and subsequent developments [12] is the use of *incremental floorplanning*. The tool takes an early floorplan of the SoC (without the interconnect) as an input, which is used to guide the synthesis process. Once a topology is designed, the tool inserts the NoC components in the best positions in the floorplan, while marginally perturbing the initial floorplan input. This incremental floorplan is then fed as an input to standard placement and routing tools. This approach captures accurately wire delays and power values of the NoC during topology synthesis.

Around the same time, several other works also tackled different aspects of the application specific custom topology synthesis [13]-[19]. Some groups have established complete tool flow for NoCs, that include CAD tools to design the NoCs, covering from architecture to physical design issues. Æthereal [21] and Netchip [42] are examples of such complete tool flows.

3. NOC ARCHITECTURES

Many NoC architectures have been proposed in literature. In on-chip networks, the wiring limitations are less strict than in multiprocessor chip-to-chip networks. For this reason, simple switches with little buffering and reduced complexity are the most suitable. Complex switches that aim at maximizing link utilization are not needed for most NoCs and some of the initial NoC architectures overlooked this trend, which is dominant in recent implementations. The subject of NoC architecture is extensive and we refer the reader to [4] for a detailed coverage. We report here on modular NoC architectures that can be built out of a simple (parametrizable) library, and that can be thus be considered in connection with design automation.

A modular NoC architecture usually consists of at least three basic elements:

- Network Interfaces (NIs)
- Switches
- Links

The main role of the *Network Interfaces* is to convert the bus protocol that is used by the *Processing Elements* (PEs) to the network protocol used by the switches. An NI is needed to connect each IP core to the NoC. NIs convert transaction requests/responses into packets and vice versa. Packets are then serialized into a sequence of *FLow control unITS (flits)* before transmission, to decrease the physical wire parallelism requirements. While there are no standards NoC protocols for intra-network communication, many NoCs support standard protocols (e.g., OCP, AHB, AXI, Wishbone, OPB, PLB) at the outer edge, to connect the PEs to NIs. This enables existing IPs to be connected easily to the network and provides greater flexibility in connecting IPs using different protocols to the same system interconnect.

Switches are the backbone of the network. Their main function is to route packets from source to destination. They provide arbitrary connectivity between several inputs and several outputs and allow for implementation of different topologies in order to provide connectivity for many PE. Switches provide buffering resources to lower congestion and improve performance. The buffers could be placed at the input ports (*input-queued router*), output ports (*output-queued router*) or at both places. In many NoCs with regular topologies where one or few cores are connected to a switch, the functionality of the NI is integrated in the switch itself. Links abstract the connectivity between NIs and switches and between the switches themselves. Links can represent more than just physical wires as they can provide pipelining in order to achieve the required timing.

As an example, a brief description of the basic NoC library components, based on ×pipes [44], is presented in Figure 1. This library incorporates features that have been successful in many NoC designs. In ×pipes, two separate NIs are defined, an *initiator* and a *target* one, respectively associated with system masters and system slaves. A master/slave device will require an NI of each type to be attached to it. The interface among IP cores and NIs is point-to-point as defined by the *Open Core Protocol OCP 2.0* [46] specification, guaranteeing maximum re-usability. NI *Look-Up Tables* (LUTs) specify the path that packets will follow in the network to reach their destination (source routing). ×pipes switches use wormhole switching, as most common in NoCs, but support two variations of flow control. If *ACK/NACK* flow control is used then output buffers are required, as flits have to be retransmitted until the downstream router has sufficient capacity to store and accept them.

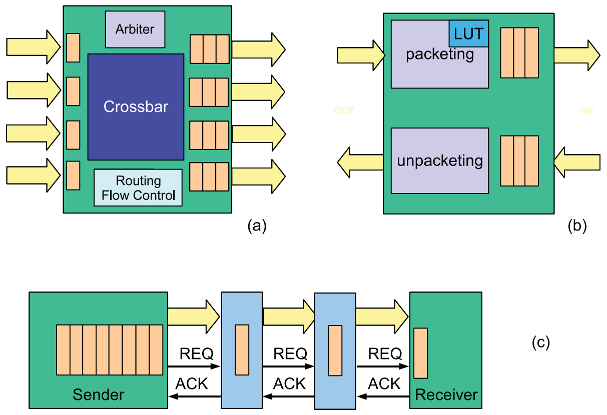

Figure 1: ×pipes building blocks: (a) switch, (b) NI, (c) link

If *ON/OFF* flow control is used, backpressure from the downstream switch stalls the transmission until the there is sufficient buffering capacity. In this case, output buffers can be omitted. In any case, the arbiter is required to resolve conflicts between packets when they require access to the same physical link.

Other arbitration and routing schemes have been developed in order to offer support for predictable communication behavior. The *Æthereal* NoC design framework presented in [21] aims at providing a complete infrastructure for developing heterogeneous NoC with end-to-end quality of service guarantees. The network supports *guaranteed throughput* (GT) for real time applications and *best effort* (BE) traffic for timing unconstrained applications. The architecture offers so-called GT connections which provide bandwidth and latency guarantees on that connection. In order to provide bandwidth and latency guarantees, it uses a *Time Division Multiple Access* (TDMA) mechanism to divide time in multiple time slots, and then assigns each GT connection a number of slots. The result is a slot-table in each NI, stating which GT connection is allowed to enter the network at which time-slot. For traffic that has no real-time requirements, Æthereal implements Best-Effort connections.

4. PHYSICAL IMPLEMENTATION

One of the main strengths of NoCs is their promise to simplify back-end design closure. For this reason, it is essential to assess and optimize the interplay of NoC technology with physical design processes.

4.1 Structured Wiring

NoCs have been improving on-chip wiring in two distinct ways. First, the packetization paradigm enables easily the implementation of communication serialization. A typical on-chip bus requires around 100 to 200 wires: 32 or 64 bits of write data, 32 or 64 bits of read data, 32 bits of address, plus control signals. On the other hand, a NoC sends packets, and can do so by splitting them over multiple cycles in flits. Therefore, it does not, in principle, have constraints over how many wires need to be deployed in parallel. By deploying highly serialized links, routing can be simplified, while area and crosstalk can be minimized. In practice, a lower bound is set by performance needs. Published NoC research shows that some implementations [21] have gone for a fixed flit width and packet structure, whereby the number of wires is much more manageable than in buses, e.g. 32; some others [44] even allow for

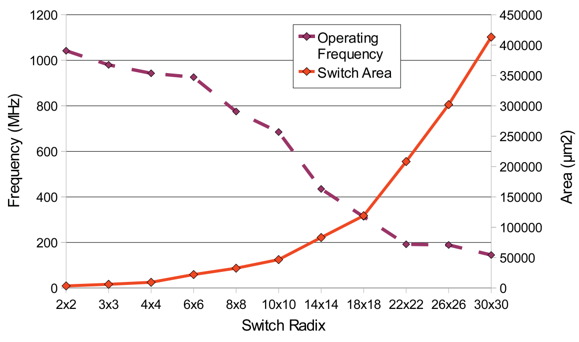

Figure 2: Study on 65nm, 32-bit switch scalability. Routers up to 10x10: 85% row utilization or more; 14x14 to 22x22: 70% to 50% row utilization; 26x26 and above: DRC violations to tackle manually even at 50% row utilization

complete flexibility, letting designers choose their favorite performance/wiring tradeoff.

A second contribution of NoCs to a tidier wiring implementation is through wire segmentation. As NoC wires are laid point-to-point, as opposed to being multipoint nets in buses, it is possible to optimize NoC topologies to constrain maximum wire lengths. This is done either by choosing highly regular topologies, e.g. meshes, or by suitable NoC topology synthesis. Furthermore, as shown in Section 3, links can be explicitly segmented to further break critical paths. This is simpler on a NoC than on a bus, where most specifications, including AMBA AHB [33], implicitly assume single-cycle communication among masters and slaves.

4.2 Routability

Bus-based architectures have been extended with components such as crossbars, as e.g. in Multilayer AHB [33], whereby fully-connected data lanes allow for parallel communication among a plurality of masters and slaves. Crossbars are successful at providing non-blocking access and minimizing arbitration delays. Unfortunately, if the inputs and outputs of the crossbars are 100- to 200-wires wide as in buses, crossbars may exhibit serious physical wire routability issues. Due to this, commercial tools [34] often constrain the maximum crossbar size to 8x8 or less. NoCs permit wire serialization, largely obviating the issue. Figure 2, based on [43], shows that NoC switches of radix 10x10 can be efficiently designed, and even much larger switches are still feasible, though at an area and frequency cost. Alternatively, smaller NoC routers can be chosen, completely solving routability concerns.

4.3 Synchronization Schemes

To tackle the increasing challenges of global clock distribution in large chips, including the power cost and variability concerns, a variety of *Globally Asynchronous Locally Synchronous* (GALS) chip design paradigms have been proposed. NoCs offer a natural backbone for the implementation of such approaches. This is because packet-switching networks (i) are distributed, (ii) natively provide ways to tackle heterogeneity, including in timing, and (iii) natively decouple transaction injection and transaction transport times. Among others, fully asynchronous communication [35] and pausible clocking [24] have been proposed and demonstrated. By incorporating all necessary timing adaptation features natively in the on-chip communication framework, designs can converge more quickly and easily, strengthening the "plug&play" view of system composability.

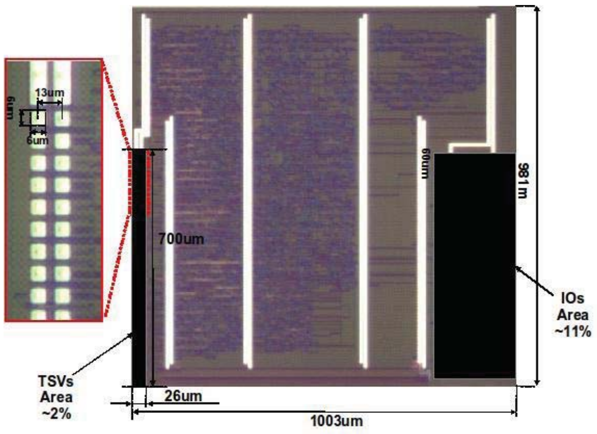

Figure 3: 3D IC with NoC for communication

4.4 3D Integration Extensions

3D chip stacking is increasingly touted [47] as a way to pursue Moore's Law and "More than Moore" visions of heterogeneous, multi-functional products including MEMS and bio-interfaces. Nevertheless, to be successful, 3D integration still has to solve some shortcomings, such as the yield of vertical connections, the area overhead, and the complexity of system design and verification.

NoCs are an ideal fit to 3D design paradigms because they represent a flexible, scalable, distributed backbone. Figure 3 shows a chip where iNoCs [36] technology has successfully met the constraints of 3D design. For example, area and yield have been optimized by suitably serializing vertical links, to minimize the number of required vertical vias. Verification has been automated by leveraging built-in link testing facilities. 3D system integration has also been made easier by the flexibility of NoC routing tables, easily enabling either 2D-only operation (in testing mode) or 3D-capable communication. The NoC also hides the 3D clocking challenges by natively handling synchronization among layers.

5. NOC CHIP EXAMPLES

Many SoCs for multimedia and wireless connectivity applications fabricated in $45nm$ technology have either a proprietary NoC or NoC IP from a third party IP provider. In this section, we show some chip-level implementations of NoCs. We provide small case-studies on the use of NoCs and their implementations.

ARM is commercializing the *AMBA Network Interconnect* [32]. The IP library provides crossbars and bridges that can be assembled to construct a hierarchical bus interconnect that closely resembles an NoC, because the topology of the interconnect can be designed to match the floorplan and buffering can be added to provide stalls.

NoCs are being actively used in CMPs, both in academia and in real products. The Tilera TILE-Gx processor [27] has 100 cores integrated onto a chip, with the cores connected by a 2D mesh network. A NoC-based interconnect from Arteris [28] has been implemented to connect the components on the TI OMAP platform. Several research prototypes for CMPs, such as the TRIPS processor [48], Smart Memories [49], use a NoC to connect the cores together. The Intel Teraflops [37], a prototype 80-core processor, also uses a mesh network to connect the cores. A block diagram depicting a core and the 5-port router is shown in Figure 4. Each core consists of two programmable floating point units and a five-port router. The routers are connected in a 2D mesh topology. In order

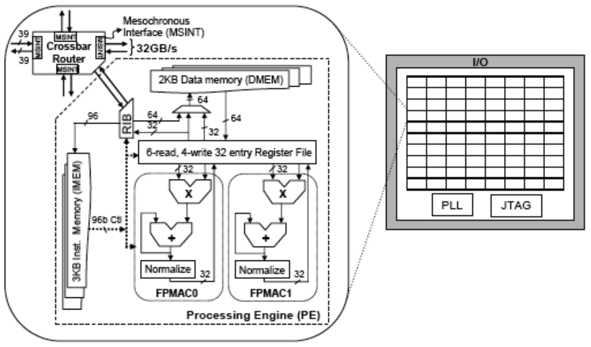

Figure 4: A single core and a 5-port router of the Intel 80 core processor

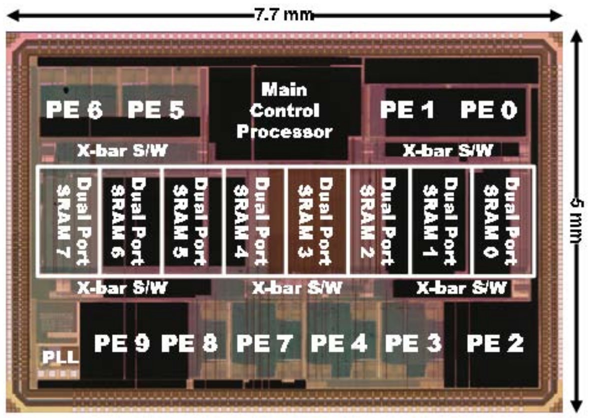

Figure 5: A multicore NoC using the BONE architecture

to avoid the communication overhead in maintaining coherency, the system does not use cache coherency and instead, data is transferred using message passing. The aggregate bandwidth supported by the chip at 3.16 GHz operating speed is around 1.62 Terabits/s.

The GALS based ANoC and the multi-synchronous DSPIN NoC have been implemented in two demonstrator chips as system interconnect for the FAUST application [25]. The FAUST (Flexible Architecture of Unified Systems for Telecom) demonstrator is a SoC platform with advanced telecommunication capabilities. The implemented topology is a quasi-mesh as on some routers connect more than one core. In the receiver matrix - which consists of only of 10 cores - the aggregate required bandwidth is 10.6 Gbits/s to maintain real time communication.

Several chips have been fabricated by the BONE NoC group, including the design of a NoC based parallel processor for visual attention engine [40] and an object recognition processor [41]. The chip layout of a memory centric NoC for a homogeneous MPSoC designed by the team is shown in Figure 5. The design consists of 8 dual port memories, crossbar switches and ten RISC processors. They are connected in a hierarchical star topology. The dual-port SRAMs are assigned dynamically to the RISC processors that are exchanging data. The crossbars act as a non-blocking medium to connect the RISC processors and the SRAMs. The architecture supports flexible mapping of tasks to processors, thereby providing better performance than a conventional 2D mesh-based CMP.

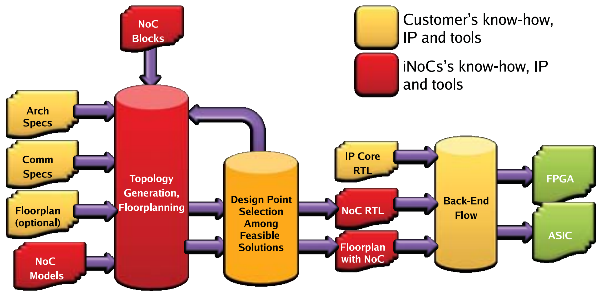
NoC Blocks
Customer's know-how, IP and tools
iNoCs's know-how, IP and tools
Arch Specs
Comm Specs
Floorplan (optional)
NoC Models
Topology Generation, Floorplanning
Design Point Selection Among Feasible Solutions
IP Core RTL
NoC RTL
Floorplan with NoC
Back-End Flow
FPGA
ASIC

Figure 6: A NoC design flow from iNoCs

6. NOC DESIGN TOOL FLOW

Designing an efficient NoC architecture, while satisfying the application performance constraints, is a complex process. The design issues include topology synthesis, finding routes for traffic flows, setting architectural parameters (such as frequency of operation, link width), verifying performance, building simulation and emulation models [42]. In order to handle the design complexity and meet the tight time-to-market constraints, it is important to automate most of these NoC design phases. To achieve design closure, the different phases should also be integrated in a seamless manner.

Some companies like: Arteris [28], iNoCs [36], Silistix [35] are providing design automation tools for NoCs . As an example, we describe here the tool flow from iNoCs, also shown in Figure 6 [36]. The tool flow takes the application architecture and application constraints as inputs. The architecture specifications include the type of core (master or slave), the kind of protocol supported. The application communication constraints include the average bandwidth of communication between the different cores, average latency constraints, hard QoS constraints on bandwidth and latency, type of transaction, traffic shape. This information is obtained by application profiling or from the designer's estimates. The tool flow also optionally takes the floorplan of the SoC without the interconnect as an input. The floorplan is an estimate of the position of each core, depending on the I/O constraints and the communication among cores. Instead of a floorplan, a simpler metric can be used, such as the relative distance between the blocks. Finally, the NoC components are characterized with the target technology library to compute the area, power and maximum operating frequency of the routers, NIs and links. All this information is fed into the design toolchain. Based on the specifications, the topology synthesis tool builds several topologies with different switch counts and architectural parameters. The topologies are designed to meet the application and architecture constraints, with each design point having different power, area and performance values. From the set of all Pareto optimal points, the designer can then choose a NoC instance. Then, the RTL of the topology is automatically generated. The tools also generate simulation models (high level as well as RTL) with traffic generators that can be used to validate the runtime behavior of the system.

If an input floorplan is provided, the synthesis tool can use the knowledge of the positions of the cores to design topologies with shorter wire lengths. Moreover, accurate delay and power consumption of the wires can be obtained and used during the synthesis process itself. The tool also produces an output floorplan for the topology point, with the NoC components placed at the ideal locations, in order to minimize power consumption and delay. Then, the RTL and simulation models of the topology are generated. The tool flow supports several features, such as providing hard real-time guarantees if needed, without additional hardware support. It also supports the concept of voltage islands, where cores in an island operate at the same frequency and voltage, while cores in different islands can operate at different frequencies and voltages.

7. CONCLUSIONS

Network on Chip technology has been established as the preferred way of realizing system-level interconnect for most (if not all) high-end SoC products for nomadic and multimedia applications fabricated with the 45nm node. These SoCs embody either a proprietary NoC or a NoC from a third-party IP provider. While looking at current and future designs of complex SoCs in 45nm technology and beyond, NoCs are essential components to achieve performance and design closure. At the same time, there is a growing interest of using NoCs in complex FPGA designs as well.

The EDA infrastructure for NoC design is provided by either inhouse design groups or by a few commercial companies. This infrastructure comprises both IP blocks and tools for optimizing and integrating the IP with the rest of the design. Needless to say, the EDA infrastructure has shown to be essential for realizing existing large-scale SoCs with NoC-based interconnects.

Future directions of NoC research and applications lead along two avenues. The former is related to 3-Dimensional chip stacking, with 3D NoCs providing a modular and flexible interconnect means that can also obviate for vertical connection failures and engineering changes. The second avenue deals with other structured interconnect means, including optical NoCs as well as networks integrating chips and the environment with RF means. In all these cases, key features are the separation of computation and communication via a NoC as well as the NoC modularity and support for scalability.

8. ACKNOWLEDGMENT

We would like to acknowledge the financial contribution of CTI under project 10046.2 PFNM-NM, the ARTIST-DESIGN Network of Excellence and the EU FP7 project NaNoC (248972).

9. REFERENCES

[1] C. Seitz, “Let’s Route Packets Instead of Wires.” Advanced Research in VLSI: Proceedings of the Sixth MIT Conference, 1990, pp. 133-138.
[2] L.Benini and G.De Micheli, “Networks on Chips: A New SoC Paradigm”, IEEE Computers, pp. 70-78, Jan. 2002.
[3] P.Guerrier, A.Greiner,”A generic architecture for on-chip packet switched interconnections”, Proc. DATE, pp. 250-256, March 2000.
[4] G. De Micheli, L. Benini, “Networks on Chips: Technology and Tools”, Morgan Kaufmann, First Edition, July, 2006.
[5] W. Dally, B. Towles, “Route Packets, not Wires: On-Chip Interconnection Networks”, Proc. DAC, pp. 684-689, June 2001.
[6] H.Zhang et. al., "A 1V Heterogeneous Reconfigurable DSP IC for Wireless Baseband Digital Signal Processing", IEEE Journal of Solid State Circuits, Vol.35, No.11, pp. 1697-1704, Nov 2000.
[7] W.J.Dally, S.Lacy, "VLSI Architecture: Past, Present and Future", Conf. Adv. Research in VLSI, pp. 232-241, 1999.
[8] R. Marculescu, “Networks-on-chip: the quest for on-chip fault-tolerant communication”, Proc. IEEE ISVLSI, pp. 8-12, Feb 2003.
[9] S. Murali, G. De Micheli, “SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs”, Proc. DAC 2004.
[10] Shashi Kumar, Axel Jantsch, Mikael Millberg, Johny Oberg, Juha-Pekka Soininen, Martti Forsell, Kari Tiensyrja, Ahmed Hemani, "A network on chip architecture and design methodology”, ISVLSI 2002, pp.105-112, 2002.
[11] S. Murali et al., “Designing Application-Specific Networks on Chips with Floorplan Information”, pp. 355-362, ICCAD 2006.
[12] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli. SunFloor 3D: A Tool for Networks on Chip Topology Synthesis for 3D Systems on Chip. In DATE 2009, pages 9-14, 2009
[13] J. Xu et al., “A design methodology for application-specific networks-on-chip”, ACM TECS, 2006.
[14] A.Pinto, L. Carloni, A. Sangiovanni-Vincentelli, “Efficient Synthesis of Networks on Chip”, Proc. ICCD, pp. 146-150, Oct 2003.
[15] T. Ahonen, D. Signza-Tortosa, H. Bin, J. Nurmi, “Topology Optimization for Application Specific Networks on Chip”, Proc. SLIP, pp. 53-60, Feb 2004.
[16] K. Srinivasan, K. Chatha, G. Konjevod, “An Automated Technique for Topology and Route Generation of Application Specific On-Chip Interconnection Networks”, Proc. ICCAD, pp. 231-237, Nov 2005.
[17] W.H.Ho, T.M.Pinkston, “A Methodology for Designing Efficient On-Chip Interconnects on Well-Behaved Communication Patterns”, Proc. HPCA, pp. 377-388, Feb 2003.
[18] X.Zhu, S.Malik, "A Hierarchical Modeling Framework for On-Chip Communication Architectures", ICCD 2002, pp. 663-671, Nov 2002.
[19] I.Saastamoinen, D.Siguenza-Tortosa, J. Nurmi, “Interconnect IP node for future system-on-chip designs", Proc. of The First IEEE International Workshop on Electronic Design, Test and Applications, pp. 116-120, Jan. 2002.
[20] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, “QNoC: QoS architecture and design process for Network on Chip”, The Journal of Systems Architecture, pp. 105-128, Vol. 50, Issue: 2-3, Feb 2004.
[21] E. Rijpkema, K. Goossens, A. Radulescu, J. Dielissen, J. van Meerbergen, P. Wielage, E. Waterlander, "Trade-offs in the design of a router with both guaranteed and best-effort services for networks on chip", Proc. DATE, pp. 350-355, Mar 2003.
[22] M. Coppola et al., “Spidergon: a novel on-chip communication network”, Proc. IEEE International Symposium on System-on-Chip 2004. 16–18, p. 15, Nov. 2004.
[23] E. Beigne et al., “An Asynchronous NoC Architecture Providing Low Latency Service and its Multi-Level Design framework”, ASYNC’2005, pp. 54-63, March 2005.
[24] E. Beigne et al., “Dynamic voltage and frequency scaling architecture for units integration with a GALS NoC”, NOCS, pp. 129ï£¡138, 2008.
[25] Miro-Panades, I. et al., “Physical Implementation of the DSPIN Network-on-Chip in the FAUST Architecture", NoC Symposium , 2008.
[26] Bjerregaard, T. et al. “An OCP Compliant Network Adapter for GALS-based SoC Design Using the MANGO Network-on-Chip", Proc. SoC 2005
[27] http://www.tilera.com/products/processors.php
[28] http://www.arteris.com/flex_noc.php
[29] A. Artieri, V. D Alto, R. Chesson, M. Hopkins, M. C. Rossi, “Nomadik Open Multimedia Platform for Next-generation Mobile Devices”, STMicroelectronics Technical Article TA305, 2003, available at http://www.st.com
[30] J. Helmig, “Developing core software technologies for TI’s OMAPTM platform”, Texas Instruments, 2002. Available at http://www.ti.com.
[31] http://www.infineon.com/cms/en/corporate/press/news/releases/2006/170186.html
[32] http://www.arm.com/products/system-ip/interconnect/axi/index.php
[33] http://www.arm.com/products/system-ip/amba/amba-open-specifications.php
[34] http://www.synopsys.com/dw/ipdir.php?ds=core_consultant
[35] http://www.silistix.com/
[36] http://www.inocs.com/
[37] S. R. Vangal, et al. “An 80-Tile sub-100-w teraflops processor in 65-nm cmos”, Solid-State Circuits, IEEE Journal of, 43(1):29–41, 2008.
[38] J. Howard et al, "A 48-Core IA-32 Message-Passing Processor with DVFS in 45nm CMOS", ISSCC 2010.
[39] M. Taylor et al., “The Raw Microprocessor: A Computational Fabric for Software Circuits and General Purpose Programs”, IEEE Micro, April 2002.
[40] K. Kim et al., “A 125GOPS 583mW network-on-chip based parallel processor with bio-inspired visual attention engine”, IEEE ISSCC 2008.
[41] K. Kim et al., “A 201.4GOPS 496mW Real-Time Multi-Object Recognition Processor with Bio-Inspired Neural Perception Engine”, IEEE ISSCC 2009.
[42] D. Bertozzi et al., “NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip”, IEEE Transactions On Parallel And Distributed Systems, pp. 113-129, Vol. 16, No. 2, Feb 2005.
[43] A. Pullini, F. Angiolini, S. Murali, D. Atienza, G. De Micheli and L. Benini, Bringing nocs to 65 nm, IEEE Micro 27 (5) (2007), pp. 75ï£¡85.
[44] S. Stergiou et al., “×pipesLite: a Synthesis Oriented Design Library for Networks on Chips”, pp. 1188-1193, Proc. DATE 2005.
[45] A.Jalabert et. al, “×`pipesCompiler`: A Tool For Instantiating Application Specific Networks on Chips”, Proc. DATE 2004.
[46] www.ocpip.org
[47] K. Banerjee et al., “3-D ICs: A Novel Chip Design for Deep-Submicrometer Interconnect Performance and Systems-on-Chip Integration”, Proc. of the IEEE, vol. 89(5), pp. 602, 2001.
[48] K. Sankaralingam et al., “Exploiting ILP, TLP, and DLP with the Polymorphous TRIPS Architecture”, IEEE Micro, Nov/Dec 2003.
[49] R. Ho, K. Mai, M. Horowitz, “Efficient On-Chip Global Interconnects”, IEEE Symposium on VLSI Circuits, June 2003.