Available online at www.sciencedirect.com

ScienceDirect

INTEGRATION, the VLSI journal 41 (2008) 340–359

INTEGRATION the VLSI journal

www.elsevier.com/locate/vlsi

Invited paper

Network-on-Chip design and synthesis outlook

David Atienza[a,b,*,1], Federico Angiolini[c,a], Srinivasan Murali[a], Antonio Pullini[d], Luca Benini[c], Giovanni De Micheli[a]

[a]*LSI-EPFL, Station 14, 1015 Lausanne, Switzerland*
[b]*DACYA-Complutense University, Avda. Complutense s/n, 28040 Madrid, Spain*
[c]*DEIS-University of Bologna, viale Risorgimento 2, 40136 Bologna, Italy*
[d]*DAUIN-Politecnico di Torino, corso Duca degli Abruzzi 24, 10129 Torino, Italy*

Received 26 November 2007; received in revised form 23 December 2007; accepted 27 December 2007

Abstract

With the growing complexity in consumer embedded products, new tendencies forecast heterogeneous *Multi-Processor Systems-On-Chip* (MPSoCs) consisting of complex integrated components communicating with each other at very high-speed rates. Intercommunication requirements of MPSoCs made of hundreds of cores will not be feasible using a single shared bus or a hierarchy of buses due to their poor scalability with system size, their shared bandwidth between all the attached cores and the energy efficiency requirements of final products.

To overcome these problems of scalability and complexity, *Networks-On-Chip* (NoCs) have been proposed as a promising replacement to eliminate many of the overheads of buses and MPSoCs connected by means of general-purpose communication architectures. However, the development of application-specific NoCs for MPSoCs is a complex engineering process that involves the definition of suitable protocols and topologies of switches, and which demands adequate design flows to minimize design time and effort. In fact, the development of suitable high-level design and synthesis tools for NoC-based interconnects is a key element to benefit from NoC-based interconnect design in nanometer-scale CMOS technologies.

In this article we overview the benefits of state-of-the-art NoCs using a complete NoC synthesis flow, and a detailed scalability analysis of different NoC implementations for the latest nanometer-scale technology nodes. We present NoC-based solutions for the on-chip interconnects of MPSoCs that illustrate the benefits of competitive application-specific NoCs with respect to more regular NoC topologies regarding performance, area and power. Moreover, we show that it is currently feasible to synthesize in an automatic way a complete custom NoC interconnect from a high-level specification in few hours. Finally, we summarize future research challenges in the area of NoC interconnect design automation.

Keywords: Network-on-chip; System-on-chip; Design automation; Wires

1. Introduction

In the last years there has been an increase in computation requirements for embedded systems due to the increasing complexity of new communication and multimedia standards. This has fostered the development of high-performance embedded platforms that can handle the computational requirements of recent complex algorithms, which cannot be executed in traditional embedded mono-processor architectures.

In addition, the continuous time-to-market pressure for consumer embedded devices has made it impossible for a

*Corresponding author at: LSI-EPFL, Station 14, 1015 Lausanne, Switzerland. Tel.: +41 21 69 30 915; fax: +41 21 69 30 909.

E-mail addresses: david.atienza@epfl.ch, datienza@dacya.ucm.es (D. Atienza), federico.angiolini@epfl.ch, federico.angiolini@unibo.it (F. Angiolini), srinivasan.murali@epfl.ch (S. Murali), antonio.pullini@polito.it (A. Pullini), luca.benini@unibo.it (L. Benini), giovanni.demicheli@epfl.ch (G. De Micheli).

[1]This work is partially supported by the Swiss FNS Research Grant 20021-109450/1, the Spanish Government Research Grant TIN2005-5619, and a grant by STMicroelectronics for University of Bologna.

0167-9260/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.vlsi.2007.12.002

design group to perform a complete redesign each time a new product needs to be developed. Due to all these requirements, *Multi-Processor System-on-Chip* (MPSoC) architectures have become a very attractive solution for the new consumer multimedia embedded market [1]. As a matter of fact, some platforms from the major semiconductor vendors (e.g., Philips Nexperia [2], TI OMAP [3] or ST Nomadik [4]) are already available today exemplifying these paradigms in heterogeneous platforms.

Although MPSoCs promise to significantly improve the processing capabilities and versatility of embedded systems, one major problem in their current and future design is the effectiveness of the interconnection mechanisms between the internal components, as the amount of components grows with each new technological node. Bus-based designs are not able to cope with the heterogeneous and demanding communication requirements of MPSoCs. Moreover, as the semiconductor industry reaches deep sub-micron technologies [5], power density and process variations become critical design concerns for embedded systems as well; thus, predictability in the design of on-chip interconnects is becoming as important as the provided bandwidth. Hence, new paradigms and methodologies that can design power-effective and reliable interconnects for MPSoCs are a must nowadays.

Networks-on-Chip (NoCs) have been suggested as a promising solution to the aforementioned scalability problem of forthcoming MPSoCs [6,7]. NoCs build on top of the latest evolutions of bus architectures in terms of advanced protocols and topology design, and, by bringing packet-based communication paradigms to the on-chip domain, they address many of the upcoming issues of interconnect fabric design better than buses [8]. For example, wire lengths can be controlled by matching network topology with physical constraints; bandwidth can be boosted simply by increasing the number of links and switches. Furthermore, compared to irregular, bridge-based assemblies of clusters of processing elements, NoCs also help in tackling design complexity and verification issues [9,10].

Using NoCs the interconnect structure and wiring complexity can be controlled well. When the interconnect is structured, the number of timing violations that occur during the physical design (floorplanning and wire routing) phase are minimal. Such design predictability is critical for today's MPSoCs to achieve timing closure. It leads to faster design cycle, reduction in the number of design re-spins and faster time-to-market. As the wire delay as a fraction of gate delay is increasing with each technological generation, having shorter wires is even more important for future MPSoCs. Early works on NoC topology design assumed that using regular topologies, such as meshes, like those that have been used in macro-networks, would lead to regular and predictable layouts [10,11]. While this may be true for designs with homogeneous processing cores and memories, it is not true for most MPSoCs as they are typically composed of heterogeneous cores and regular topologies result in poor performance, with large power and area overhead. This is due to the fact that the core sizes of the MPSoC are highly non-uniform and the floorplan of the design does not match the regular, tile-based floorplan of standard topologies [10]. Moreover, for most state-of-the-art MPSoCs (like the Cell-Playstation III [12], Philips Nexperia [13] or TI OMAP [3]) 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 MPSoC can be obtained statically. Thus, an application-specific NoC with a custom topology, which satisfies the design objectives and constraints, is critical to have efficient on-chip interconnects for MPSoCs.

When designing an efficient NoC architecture, satisfying the application performance constraints is a complex process. The design issues span several abstraction levels, ranging from the high-level application modeling to the physical layout-level implementation. Some of the most important phases in designing NoCs include modeling the application traffic characteristics, synthesizing the topology or structure of the network, setting various design parameters (such as frequency of operation or link width), generating the *Register Transfer Level* (RTL) code for the network components and performing the physical design. 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, which is not an easy challenge and opens many commercial opportunities. In fact, three companies already exist in this space that have started bringing the NoC technology to commercial product, namely, Arteris [14], Silistix [15] and iNoCs [16], and more start-ups are likely to appear in the growing market of MPSoC on-chip interconnect design.

This paper tries to provide an overview of the currently available NoC-based interconnects for next generations of MPSoC platforms. Thus, to take into account as many key effects as possible in our studies of NoC interconnects, we establish a flow in this work that takes our MPSoC test platforms down to placed&routed layouts. This flow allows us to derive final frequency, area and power figures for the NoC blocks to perform complete studies of different overall NoC interconnects. In our analyses and study of on-chip interconnects we cover NoCs implemented with the proposed design flow using three different technology libraries (130, 90 and 65 nm), such that we can provide conclusions for a very representative part of the design spectrum. Finally, according to our conclusions we suggest a number of future research challenges in the area of NoC technologies and interconnect design automation for forthcoming MPSoC embedded platforms.

The remainder of the paper is structured in the following way. In Section 2, we present an overview of state-of-the-art in the area of on-chip interconnect architectures and NoC design methods for MPSoCs. Then, Section 3 presents the architectural foundations of NoCs. Next, in

Section 4 we describe the proposed design flow and exploration front-end to build reliable NoC-based interconnects for MPSoCs and Section 5 covers in detail the back-end process necessary for suitable NoC synthesis in latest process technologies. After this, in Section 6 we review the characteristics of state-of-the-art NoCs using the proposed NoC synthesis flow, and perform a complete scalability analysis of different NoC implementations for several real-life MPSoC case studies using the latest nanometer-scale technology nodes. Then, in Section 7 we present an overview of possible future research challenges that still need to be addressed in NoC interconnects to be valid for future MPSoC designs and 3D integration technologies. Finally, Section 8 summarizes the major conclusions obtained from this work.

2. Related work

A significant number of different communication fabrics have been described in the literature. The different versions of the ARM *Advanced Microcontroller Bus Architecture* (AMBA) [17], including the latest AXI [18] standard, and the ST Microelectronics STBus [19] and Sonics MicroNetworks [20] are examples of advanced buses that attempt to overcome the limitations of classical shared bus architectures and bridge-based assemblies of clusters of processing elements by different methods. For instance, multiple STBus channels can be deployed, leading to crossbars in the extreme solution. In fact, most of these bus solutions are getting very similar in their parallel multi-channel architectures for shared on-chip communication to the packet-based interconnects promoted by the NoC paradigm. Thus, making the differentiation between both paradigms for on-chip interconnects, traditionally differentiated, become almost inexistent.

Up to today several researchers have motivated the need for NoC-based designs as a way to tackle design complexity issues [6,7,9,21,22]. Research on NoC architectures has fostered the proposal of different flow control protocols [10,21,23], *Quality of Service* (QoS) provisions [24–26], and asynchronous implementations of NoCs [27,28]. Moreover, several start-ups already exist today that are bringing NoC technologies into commercial products [14–16], thus, the application of NoCs into real-life MPSoCs is already becoming a reality.

In addition, various works in the literature exist that explore the implications of the NoC paradigm at different design levels trying to propose efficient synthesis methods for NoC-based interconnects and to provide coherent comparisons with bus-based MPSoCs [8,11,29]. NoC layouts for *Application Specific Integrated Circuits* (ASICs) are presented in [30–33], a test chip is shown in [34], and different implementations of NoCs on *Field Programmable Gate Arrays* (FPGAs) are provided by [35,36].

Moreover, the design of application-specific NoC architectures for known communication patterns has been addressed in [37–41]. Also, several researchers have developed tool chains for designing application-specific NoCs [21,42], while a complete *Computer-Aided Design* (CAD) tool for NoC instantiation and optimization for both regular and custom topologies can be found in [43]. Synthesis and layout results for the ×pipes library of component blocks that we leverage upon for our study of NoC architectures are shown in [31].

3. Basic building blocks of NoC architectures

Several architectures have been proposed in the NoC literature. However, all NoCs have three fundamental building blocks, namely, *switches* (also called *routers*), *Network Interfaces* (NIs) (also called *network adapters*) and *links* [7,9,10]. The NoC is instantiated by deploying a set of these components to form a topology and by configuring them in terms of buffer depth, etc. The backbone of the NoC consists of switches, whose main function is to route packets from sources to destinations. Some NoCs rely on specific topological connectivity, such as octagon [44] or ring [45], to simplify the control logic, while others allow for arbitrary connectivity [31], providing more flexible matching to the target application. NoCs can be based on circuit or packet switching, or a mix of both; the former is aimed at providing hard QoS guarantees, while the latter optimizes the efficiency for the average case. When packet switching is chosen, switches provide buffering resources to lower congestion and improve performance. They also handle flow control [10] issues, and resolve conflicts among packets when they overlap in requesting access to the same physical links. Two of the most usual flow control protocols involve switch-to-switch communication and are retransmission-based (i.e., packets are optimistically sent but a copy of them is also stored by the sender, and, if the receiver is busy, a feedback wire to request retransmission is raised) or credit-based (i.e., the receiver constantly informs the sender about its ability to accept data, and data are only sent when resources are certainly available). End-to-end flow control schemes [10], where peripheral NIs directly exchange flow control information with each other, are more rarely used because of their buffering requirements; the most common usage scenario involves NoCs that implement circuit-switching [21].

An NI is needed to connect each core to the NoC. NIs convert transaction requests/responses into packets and vice versa. Packets are then split into a sequence of *FLow control unITS* (FLITS) before transmission, to decrease the physical wire parallelism requirements. NIs are associated in NoCs to system masters and system slaves. Many current NoC solutions leverage static source routing, which means that dedicated NI *Look-Up Tables* (LUTs) specify the path that packets will follow in the network to reach their final destination. This type of routing minimizes the complexity of the routing logic in the NoC. As an alternative, routing can be performed within the topology itself, normally in an adaptive manner; however, performance

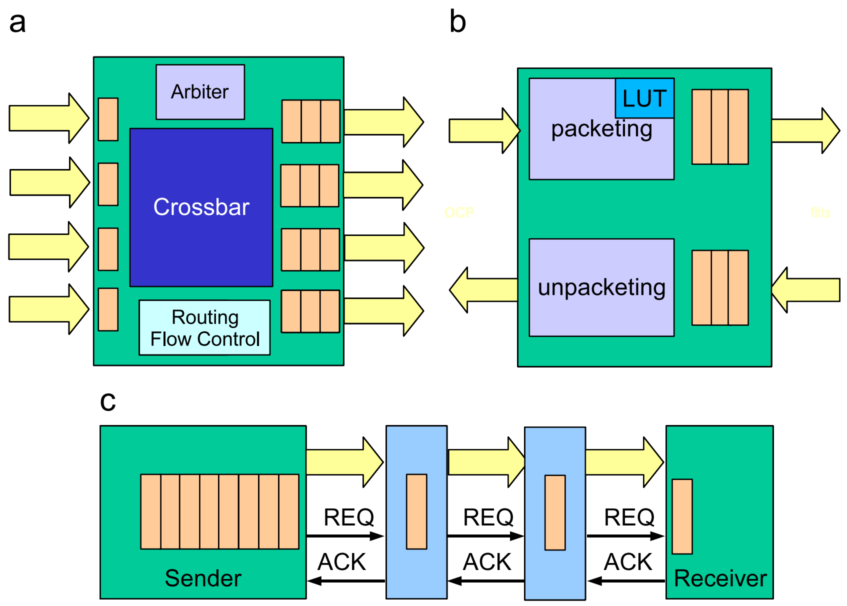


Fig. 1. × pipes NoC architectural blocks: switch (a), NI (b) and pipelined link (c).

advantages, in-order delivery and deadlock/livelock freedom are still issues to be studied in the latter case.

In general, two different clock signals can be attached to NIs: the first one drives the NI front-end, the side to which the external core is attached, and the second one drives the NI back-end, the internal NoC side. These clocks can, in general, be independent. This arrangement enables the NoC to run at a different (and potentially faster) clock than the attached cores, which is crucial to keep transaction latency low.

In this work we employ the × pipes NoC library, as a state-of-the-art NoC solution that incorporates most of the features and effective architectural solutions that have been proposed in NoC designs; thus, it is representative of various reasonable design points. The × pipes NoC [31] is an example of a highly flexible library of component blocks (see Fig. 1). The × pipes NoC can employ either ACK/NACK (retransmission-based) or STALL/GO (credit-based) flow control protocols, using output or input buffering, respectively, for maximum efficiency. Links can be pipelined and no virtual channels are implemented, as this allows for a much leaner implementation. Deadlocks are avoided by construction in the definition of the routing tables included in the NIs. Two separate NIs are defined, i.e., an initiator one (for the master cores) and a target one (for the slave cores); a master/slave device requires an NI of each type to be attached to it. The interface among cores and NIs is point-to-point as defined by the *Open Core Protocol* (OCP) 2.0 [46] specification used as public interface of the NoC, guaranteeing maximum reusability for different cores and MPSoCs. × pipes NIs support two different clock signals, one for the OCP interface and another one for the × pipes internal interface; the × pipes clock frequency must be an integer multiple of the OCP one, to greatly simplify the hardware and performance overhead of clock synchronization. Since each core can run at a different divider of the × pipes frequency, mixed-clock platforms are possible, which provides large flexibility.

4. NoC-based interconnect design flow

As explained in the previous section, a NoC consists of three main blocks (switches, NIs and links). Then, leveraging the × pipes NoC architecture, we propose in this section a complete design flow to instantiate these blocks and generate complete NoC topologies that can allow us to study the large NoC implementation spectrum.

The proposed complete flow for designing NoCs is presented in Fig. 2. The tool flow has three main phases and several tools integrated together. In the first phase or Front-End Phase, several key NoC architectural features are determined, such as the interconnect structure (or topology), routing tables or path widths for each traffic flow. We have developed the SunFloor tool to automate this phase. In the intermediate phase or Architectural Design Phase, the RTL code of the NoC architecture is instantiated. Finally, in the Back-End Phase, the NoC is implemented on FPGA or ASIC back-ends, and simulated or emulated accordingly. These three phases are explained in detail in the subsequent sections.

Before going into the methods used in the proposed flow, we present some background on NoC topology synthesis, deadlock issues and area-power modeling aspects.

4.1. Background on NoC topology synthesis

The standard topologies (mesh, torus, etc.) that have been used in macro-networks can result in poor performance and have large power and area overhead when used for heterogeneous MPSoCs. As we illustrate in this section,

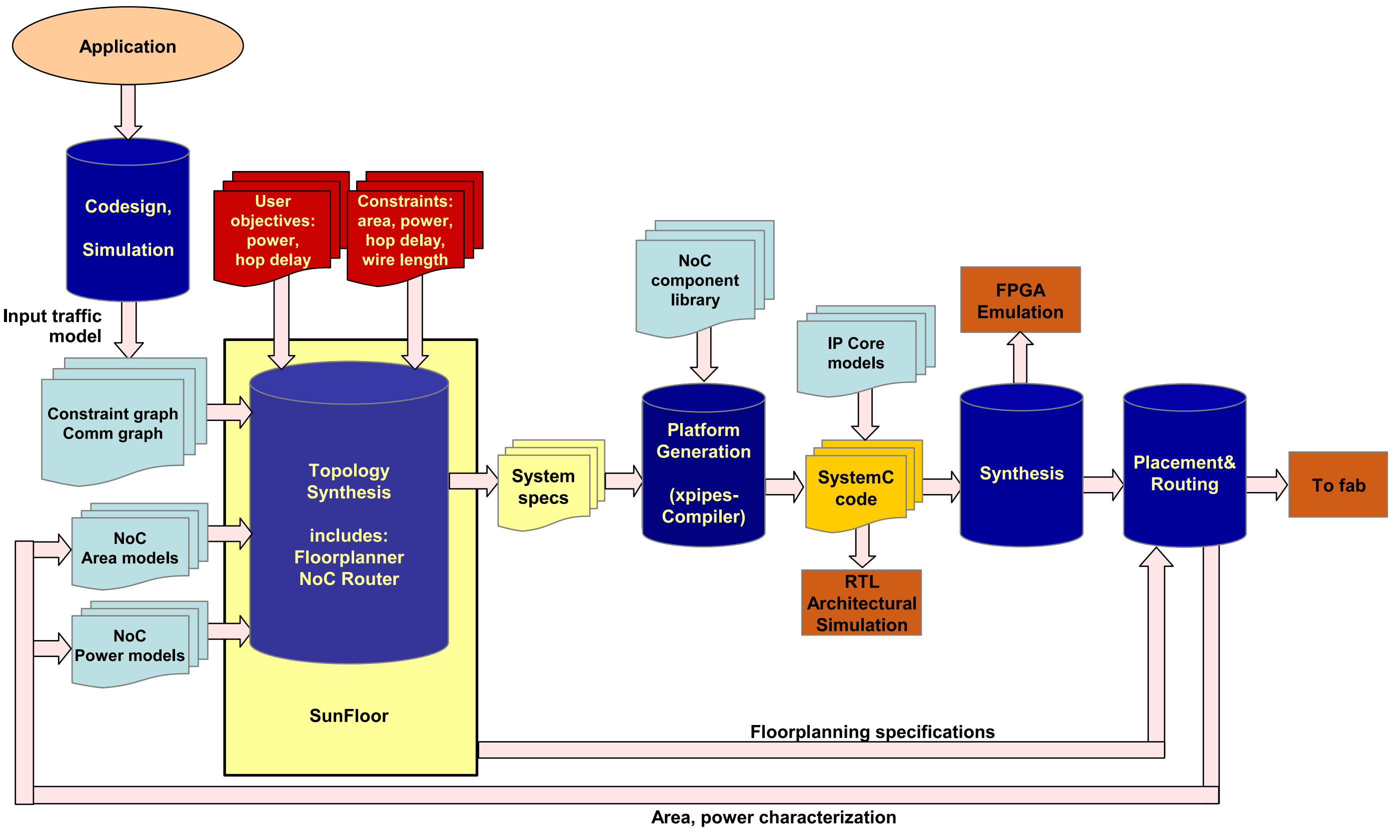

Application
Codesign, Simulation
Input traffic model
User objectives: power, hop delay
Constraints: area, power, hop delay, wire length
Constraint graph Comm graph
NoC Area models
NoC Power models
Topology Synthesis includes: Floorplanner NoC Router
SunFloor
System specs
NoC component library
Platform Generation (xpipes-Compiler)
IP Core models
SystemC code
RTL Architectural Simulation
FPGA Emulation
Synthesis
Placement& Routing
To fab
Floorplanning specifications
Area, power characterization

Fig. 2. Our complete NoC design flow.

when the interconnect can be designed in a structured manner, like in regular topologies, the number of timing violations that occur during the physical design (floorplanning and wire routing) phase tend to be reduced (see Section 5). However, the actual features of the switches (e.g., input/output connectivity or target frequency) are key elements to define the final topology, and very important additional benefits in power savings and performance can be achieved by customizing the NoCs for the actual application patterns of each final MPSoC. These additional power savings and performance improvements are critical to achieve suitable interconnects for today's and forthcoming MPSoCs.

As a motivating example, the network power consumption (switch and link power consumption), hop count, wire length and design area of two different NoC topologies for a video processor MPSoC with 42 cores [43] are presented in Table 1. The first topology is a standard mesh, while the second is a custom topology generated using the methodology presented in this paper. The wire lengths and design area are obtained from floorplanning of the NoC designs. The detailed explanation of the topologies and the floorplanning process are described later in Section 5. The custom topology leads to a 74% reduction in network power consumption, a 35% reduction in average hop count, a 1.28× reduction in total length of wires, and a 1% reduction in design area when compared to the mesh, illustrating the potential benefits of application-specific NoC designs with respect to regular topologies.

Table 1
Topology comparisons

Parameter	Mesh	Application-specific	Gain
Power (mW)	301.78	79.64	74%
Hop count	2.58	1.67	35%
Total wire length (mm)	185.72	145.37	1.28×
Design area (mm^2)	51.0	47.68	1%

4.2. *Deadlock-free NoC design*

The deadlocks that can occur in NoCs can be broadly categorized into two classes: *routing-dependent* deadlocks and *message-dependent* deadlocks [10,38,47]. Routing-dependent deadlocks occur when there is a cyclic dependency of resources created by the packets on the various paths in the network.

Message-dependent deadlocks occur when interactions and dependencies are created between different message types (e.g., requests and responses) at network endpoints, when they share resources in the network. Even when the underlying network is designed to be free from routing-dependent deadlocks, the message-level deadlocks can block the network indefinitely, thereby affecting the proper system operation.

For proper system operation, it is critical to remove both routing and message-dependent deadlocks in the network. It is also important to achieve deadlock freedom with minimum NoC area and power overhead. In the proposed topology synthesis process (Section 4.4), we integrate methods to find paths that are free from both routing and message-dependent deadlocks.

4.3. Area, power models for NoC components

We have built accurate analytical models for calculating the power consumption, area and delay of the $\times$pipes network components [48]. To get an accurate estimate of these parameters, they are extracted from real placed&routed NoC designs. Then, the switching activity in the network components is varied by injecting functional traffic. The capacitance, resistance and switching activity report are combined to estimate power consumption using Synopsys PrimeTime [49] (see Section 5 for further details).

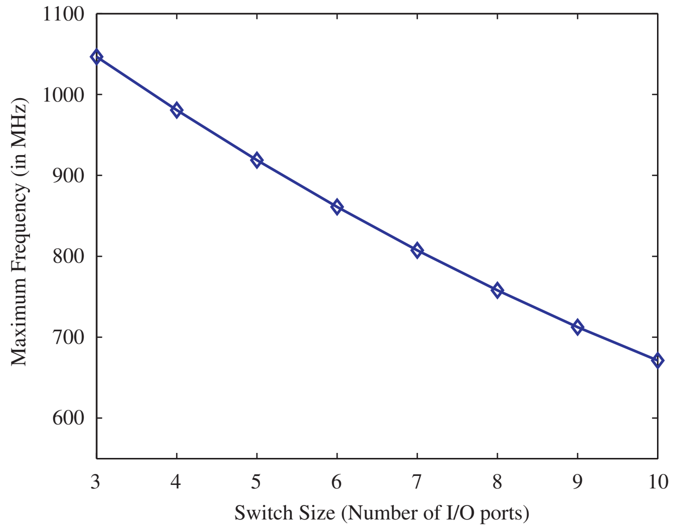


Fig. 3. Maximum frequency variation with switch size.

A large number of implementation runs were performed, varying several parameters, such as, the number of input/output ports, link width and amount of switching activity at the layout level for the NoC switches. When the size of a NoC switch increases, the size of the arbiter and the crossbar matrix inside the switch also increases, thereby increasing the critical path of the switch. To have accurate delay estimates of the switches, we model the maximum frequency that can be supported by the switches, as a function of the switch size, presented in Fig. 3 for 130 nm process technology.

We use linear regression to build analytical models for the area and power consumption of the components as a function of these parameters. Due to the intrinsic modularity and symmetry of NoC components, the models are very accurate (with maximum and mean error of less than 7% and 5%, respectively) when compared to the actual values. Power consumption on the wires is also obtained at the layout level. As in the $\times$pipes architecture each core is connected to a separate NI [50], we consider the power consumption of the NI to be part of the power consumption of the core.

The impact of the target frequency of operation on the area and energy consumption of an example 5×5 switch obtained from layout-level estimates for 130 nm is presented in Fig. 4. Note that we plot the energy values (in power/MHz) instead of the total power, so that the inherent increase in power consumption due to an increase in frequency is observed in the plot. When the targeted frequency of operation is below a certain frequency, referred to as the nominal operating frequency (around 250 MHz in the plots), the area and energy values for the switch remains the same because the switch implementation with minimal area and energy supports operation until the nominal operating frequency. However, as the targeted frequency increases beyond the nominal frequency, the area and energy values are modeled as increasing linearly with frequency. This is because the synthesis tool (such as Synopsys DC [51]) tries to match the desired high operating frequency by utilizing faster components that have large area and energy overheads. When performing the area and power estimates, we also

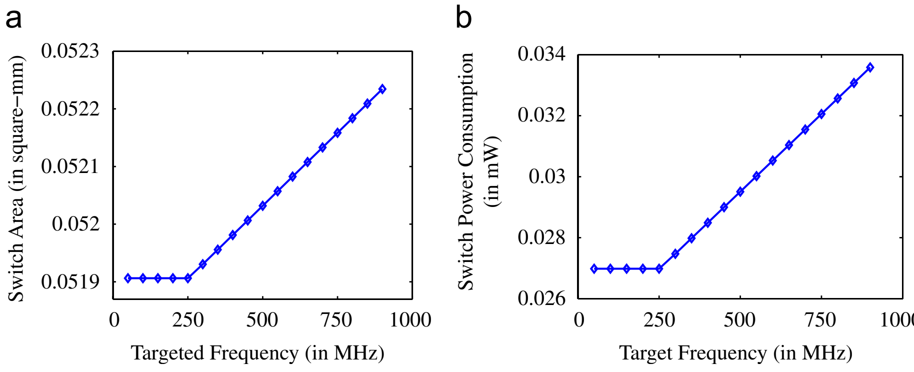


Fig. 4. Impact of frequency on area and energy of a 5×5 switch for 130 nm. (a) Impact on switch area and (b) impact on switch energy.

- Vary NoC frequency from a range
  - Vary link–width from a range
    - Vary the number of switches from one to number of cores
      - Synthesize the best topology with the particular frequency, link–width, switch–count
      - Perform floorplan of synthesized topology, get link power consumption, detect timing violations
- Choose topology that best optimizes user objectives satisfying all design constraints

Fig. 5. NoC architecture synthesis (phase 2 of design flow).

model this impact of desired operating frequency on the switch area, power consumption.

4.4. Front-end: the SunFloor tool and platform simulation

The SunFloor tool handles the front-end design of NoCs in the proposed flow (see Fig. 2). In the first phase of the flow, the user specifies the objectives and constraints that should be satisfied by the NoC. The application traffic characteristics and the area and power models for the NoC components are also taken as inputs. The SunFloor tool automatically derives the NoC architecture that optimizes the user objectives while satisfying the design constraints. The different steps in this phase are presented in detail in Fig. 5. In the outer iterations, the key NoC architectural parameters (frequency of operation and link width) are varied within a set of suitable values. The bandwidth available on each NoC link is the product of the NoC frequency and the link width. During the topology generation step, the algorithm ensures that the traffic on each link is less than or equal to its available bandwidth. The topology generation step is performed once for each set of architectural parameters of the target design space. Several topologies with different numbers of switches are explored, starting from a topology where all the cores are connected to one switch, to one where each core is connected to a separate switch. The analysis 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. Subsequently, to have an accurate estimate of the design area and wire lengths, the floorplanning of each candidate topology is automatically performed, based on the NoC area models and user-specified values for the area demands of the other cores in the design. The floorplanning process thus determines the 2D position of the cores and network components. For this purpose, we use Parquet [52], a fast and accurate floorplanner. Based on the frequency point and the obtained wire lengths, any timing violation on the wires is detected and the power consumption on the links is obtained. Eventually, from the set of all synthesized topologies and architectural parameter design points, the topology and the architectural configuration that optimize the user's objectives, while satisfying all the design constraints, are chosen.

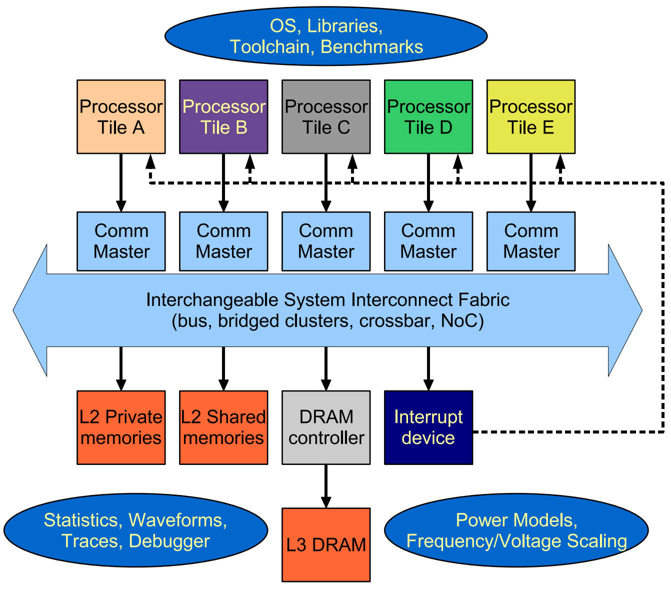


Fig. 6. The MPARM SystemC virtual platform.

As shown in Fig. 2, the next step of the flow is based on the ×pipesCompiler tool [50]. ×pipesCompiler takes care of generating the RTL SystemC code of the complete platform, by configuring and interconnecting the ×pipes soft macros based on the specifications of SunFloor. This includes the automatic generation of routing tables, based on the actual communication flows required by the application.

Once the SystemC code is available, it can be used in multiple ways. To get accurate simulation in a flexible environment, we integrate the NoC in MPARM (Fig. 6), a SystemC cycle-accurate virtual platform [53]. MPARM allows for accurate injection of functional traffic patterns as generated by real cores (processors, DMA engines, etc.) during a benchmark run. Further, it provides facilities for debugging, statistics collection and tracing.

4.5. Achieving design closure

The flow outlined in Fig. 2 is composed of several steps. Therefore, quickly achieving the design closure is not trivial. We tackle the problem in several ways. First, we try to make sure as early as possible, i.e., during the topology generation phase itself, that the timing constraints after the place&route phase will not be violated. This is a key property of SunFloor and a must in other similar tools developed to bring NoCs to state-of-the-art MPSoC design flows. The use of accurate area, power and timing models for the NoC components further bridges the gap between the topology design phase and the back-end physical-design phase.

In addition, to bridge the gap between the initial traffic models (application task graph) and the actual observed traffic after simulating the designed NoC, we introduce the use of a *mismatch* parameter. If the performance constraints are not met during simulation, the input traffic models can be scaled using this parameter and the design process is repeated. This parameter is read as part of the input specifications by the topology selection engine. The user can manually tune the parameter; conservative values will guarantee very fast design closure, while aggressive settings may need redesign iterations but will very closely tailor the NoC to the actual traffic characteristics (see Section 6.4). Several other options are also supported by the topology generation engine, such as support for cores with fixed locations in the layout (due to pin/pad constraints).

5. Back-end of a NoC implementation flow

Due to the quick pace of lithographic miniaturization, it is nowadays well known that a number of physical-level process issues related to deep sub-micron fabrication (such as wire delays and leakage power) are affecting designs. Understanding these issues is clearly key to tackling them, for example, by compensating for them at the architectural and tooling level.

In the case of NoCs, the relationship among back-end flows and architectural design is even stricter due to several factors. First, NoCs are intended to be large structures, spread across a whole chip. As such, several design issues, such as clock tree distribution, wire delays and variability, play a key role in NoCs. Second, NoCs are also designed to interconnect a large number of heterogeneous components and devices, each of which could come as a pre-built, pre-characterized core macro. Thus, it is key to be able to leverage standard back-end industrial toolchains for NoC design; otherwise, the effort of developing customized infrastructure would be impossible to afford.

In the following, we present an outline of the proposed back-end flow for NoCs, subsequently focusing our attention on specific portions of the flow that have particular relevance to tackle NoC synthesis in latest technology nodes, and the insights we gain from our study of NoC implementations.

5.1. The ×pipes back-end infrastructure

In the proposed NoC design and synthesis framework for ×pipes, we provide a complete back-end flow (see Fig. 7). As a main assumption of the NoC designs studied in this paper, without any loss of generality in our conclusions, we focus on standard cell-based physical implementations. In fact, although full custom design does certainly improve results, it does also greatly decrease flexibility and largely increases design time; thus, it is not a desirable practise for the design of current (and specially forthcoming) MPSoC interconnects. First, we perform logic synthesis by utilizing standard Synopsys tools; however, this step must be augmented with placement awareness, as will be discussed in Section 5.2. We support this procedure both on 90 and 65 nm technology libraries by a partner foundry, tuned for different performance/power trade-offs, with different threshold and supply voltages.

During synthesis, we can optionally instruct the tools to save power when buffers are inactive by applying clock gating to NoC blocks. The gating logic can be instantiated

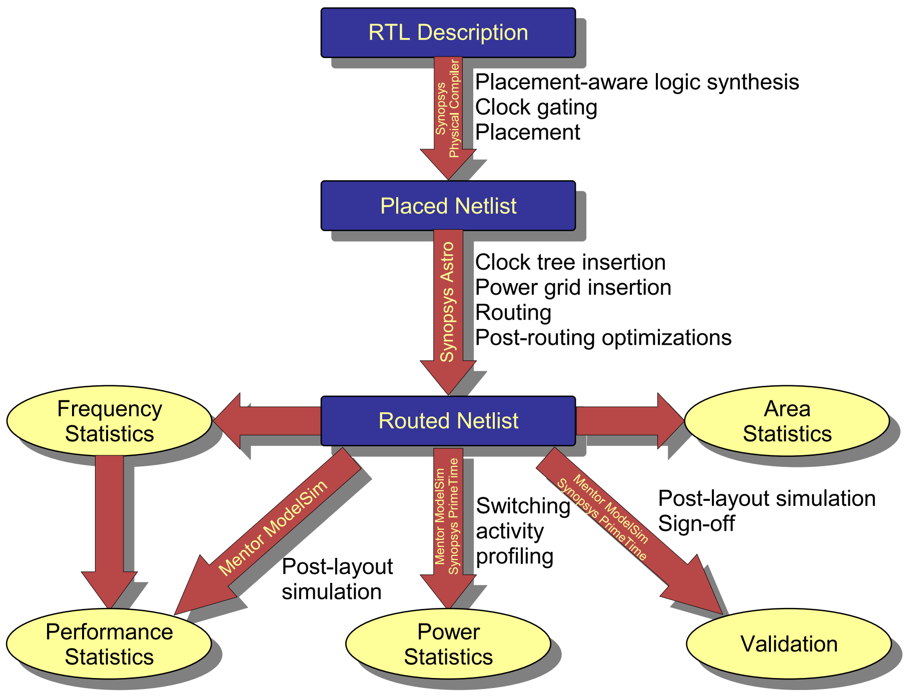


Fig. 7. The synthesis flow for ×pipes.

only for sequential cells that feature an *input enable* pin, which are a large majority of the datapath flip–flops of $\times$pipes.

We subsequently perform the detailed placement&routing step within Synopsys Astro [54]. First, we feed Astro with a coarse floorplan, generated either manually or by SunFloor. This floorplan contains *hard macros* and *soft macros*, separated by fences. The hard macros represent cores and memories and are modeled as black boxes. Hard macros are defined with a *Library Exchange Format* (LEF) file and a Verilog Interface Logical Model, and obstruct an area of choice. These boxes also obstruct some of the metal layers laying directly above; the exact number of obstructed levels is configurable, depending on how many metal layers the cores are supposed to require and on whether over-the-cell routing should be allowed for the NoC wires vs. between-the-cell. Soft macros are also boxes; they enclose the modules of $\times$pipes, and the placement tool is allowed to operate within them as long as the fences are not trespassed. By constraining the placement tool to operate on a "tile" at a time, the solution space is dramatically pruned, and relatively fast runtimes can be achieved. For proper results, however, it becomes necessary to specify rough timing constraints at the soft macro boundaries; we achieve this by pre-characterization of the links (Section 5.4).

The next step in the flow is clock tree insertion. We instantiate a clock tree within each soft macro, to minimize the memory requirements and runtime of this operation; the clock trees are then attached to a common source and balanced at the global level. The clock tree can leverage *clock borrowing* algorithms in the tools. In other words, instead of trying to fully erase clock skews (an impossible task anyway), the skews are exploited to accommodate the delay properties of the circuits, by supplying wider clock periods where the logic paths are most critical. Once the clock tree has been generated, its wires are kept untouched within the tool, to prevent further skews from appearing.

At this point, the power supply nets are added. Two main schemes are available. Traditionally, *power rings* (metal lines carrying the power supply voltages) are laid around the die; as an alternative, a *power grid* can be laid across the chip in the topmost metal layers. The latter choice requires more metal resources, but minimizes *IR drops* (voltage drops and fluctuations due to resistive effects in the supply networks and to the current draw). Therefore, we choose power grids, so as to maximize voltage stability.

Finally, the routing tool begins to route the logic wires. An initial heuristic mapping lays the wires; this initial solution is semi-random and almost certainly violates essential constraints, such as that of not shorting different wires. Therefore, *Search&Repair* (SR) loops are executed to fix any violations, including those regarding excessive propagation delays.

As a final step, post-routing optimizations are performed. This stage includes crosstalk minimization, antenna effect minimization, and insertion of filler cells. A *sign-off* procedure can be run by using Synopsys PrimeTime [49] to accurately validate the timing properties of the resulting design.

Post-layout verification and power estimation is achieved as follows. First, the netlist representing the final placed&routed topology, including accurate delay models, is simulated by injecting functional traffic through the OCP ports of the NIs. This simulation is aimed both at verifying the functionality of the placed fabric and at collecting a switching activity report. At this point, accurate wire capacitance and resistance information, as back-annotated from the placed&routed layout, is combined with the switching activity report using Synopsys PrimeTime [49]. The output is a layout-aware power/energy estimation of the simulation.

5.2. Wireload models and placement-aware logic synthesis

The traditional flow for standard cell design features logic synthesis and placement as two clearly decoupled stages. While our in-house experience [8] shows that this flow achieves reasonable results for 130 and 90 nm NoC designs, we have found the situation to be substantially different at the 65 nm node.

The origin of the problem lies in the decoupling of the two steps. Synthesis and placement could be considered as independent when wire delays were negligible; this is unfortunately not the case anymore [55]. Since wire delays can be comparable to logic delays, if not larger, it is crucial to be able to estimate wire delays already during synthesis. Since wire delays depend directly on wire length, it is clear that placement algorithms are also unfortunately affecting the solution space of synthesis algorithms.

To alleviate the problem, *wireload models* have been introduced. Wireload models are pre-characterized equations, supplied within technology libraries, that attempt to predict the capacitive load that a gate will have to drive based on its fan-out and on the overall design area. Unfortunately, wireload models remain a statistical representation of the physical reality, and are therefore an inaccurate tool to predict delays on a single net basis, given that each net could exhibit a different behavior. In our 65 nm tests, we experience unacceptable performance degradation due to either under- or over-estimations of wire loads. Even when synthesizing single NoC modules (i.e., even without considering long links), the logic synthesis tools generate a netlist with the expectation of some operating frequency; however, after placement, the actually reachable frequency is often up to 30% worse (and even lower after the routing phase). Furthermore, sometimes placement and routing tools simply do not converge towards any solution at all, trying in vain to match the expectations set by the logic synthesis step.

To address this issue NoC synthesis in 65 nm requires *placement-aware* logic synthesis tools, such as Synopsys Physical Compiler [56]. Therefore, in the proposed NoC back-end flow, after a very quick initial logic synthesis based on wireload models, the tool internally attempts a coarse placement of the current netlist. Next, it iteratively optimizes the netlist and the placement, based on the actual wire loads implied by the current candidate placement. The outcome is a placed netlist that is optimized also accounting for wire delays.

We also observe in our study of NoC synthesis that other issues may arise when placing gates into soft macros. For example, in our test designs, placement tools perform poorly when modules have to be placed within fences which are either too small or too wide. While the former case is clearly understandable, we attribute the unexpected latter effect to the placement heuristics, which are probably performing worse when the solution space becomes very large. The problem must be solved by proper tuning of the spacing among the soft macro fences and, consequently, accurate area models of the NoC modules are required to avoid very time-consuming iterations.

5.3. 65 nm technology libraries and their degrees of freedom

Fig. 8 shows how the power, speed and area of a reference ×pipes NoC switch vary, when synthesized based on different technology libraries. The experiment utilizes two 65 nm and two 90 nm libraries, labeled LP-HVT and LP-LVT; while all of these libraries belong to the *Low Power* (LP) family, the *High* V_T (HVT) variant strives for absolute minimum consumption, while the *Low* V_T (LVT) variant offers a more performance-oriented setup. The switches are fully placed&routed, including the addition of a clock tree.

A first observation is that, as hoped, synthesis in 65 nm technologies indeed offers huge benefits compared to 90 nm; both area and power experience savings around 50% among comparable libraries, while the frequency of the 65 nm design is higher (at least if the LP-LVT library variant is chosen).

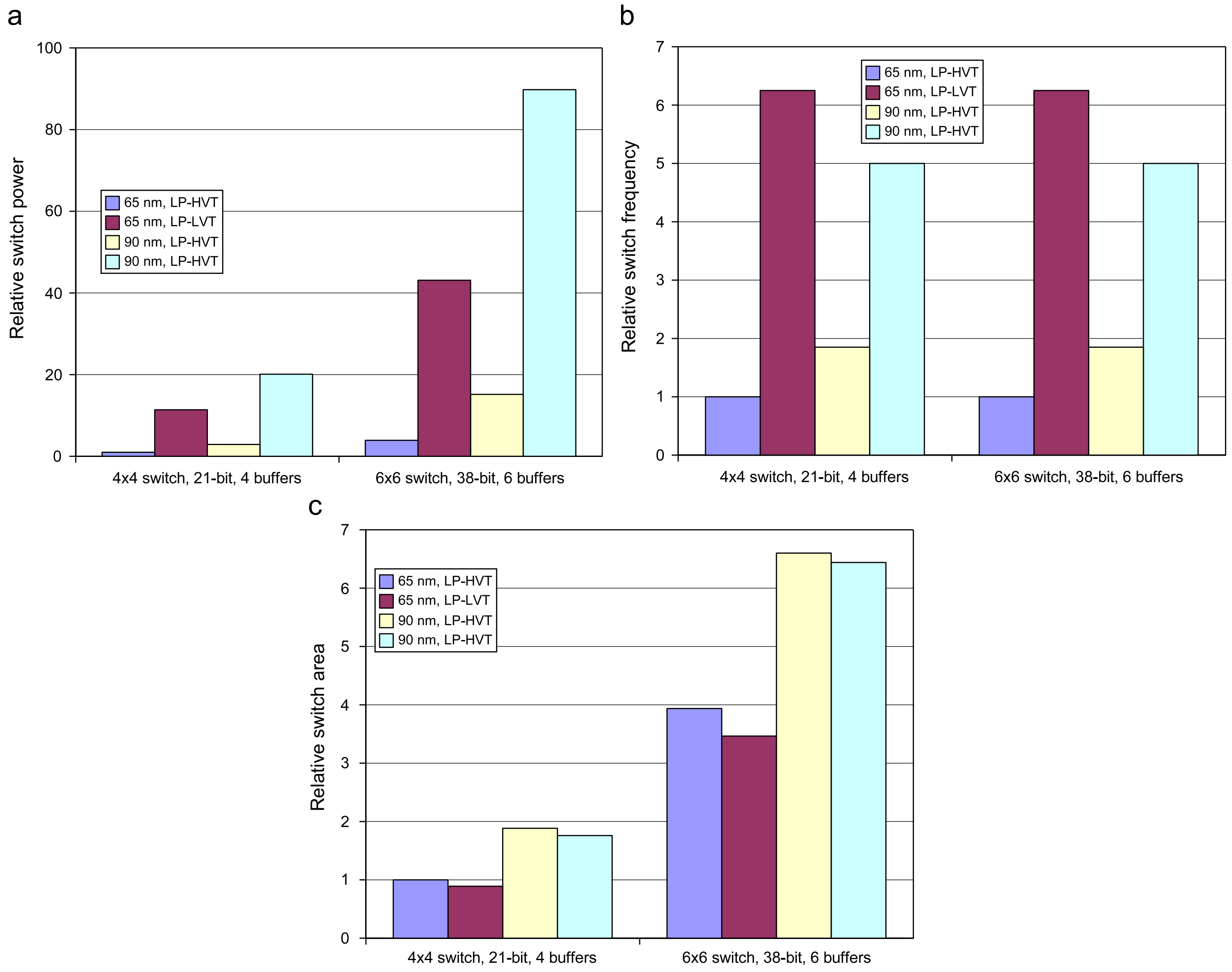


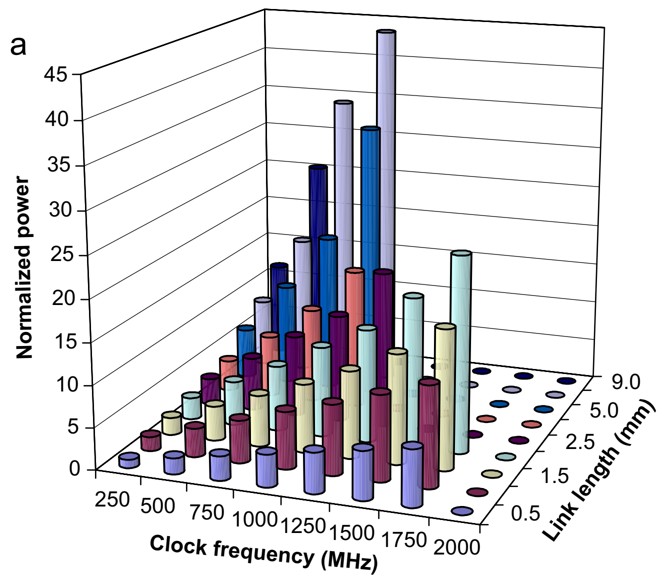
Fig. 8. Analysis of two representative ×pipes switches in different technology libraries. Figures normalized to the 4×4 switch in the LP-HVT library. (a) Power, (b) operating frequency, and (c) area.

In addition, it is also relevant to observe that, considering for example the power results, already in 90 nm technology, there is a factor of six difference among the power consumption of the LP-LVT and LP-HVT implementations; in 65 nm, this gap increases to 11×. Similarly for frequency, a gap of 3× in 90 nm becomes a gap of 6× in 65 nm. Therefore, when designing for next-generation technologies, it is in fact impossible to identify a single technological target. In fact, a very large set of trade-offs is available, where, by several metrics, results can be up to one order of magnitude different from one another. Then, it is the designer's responsibility to identify the best set of technological choices in NoC synthesis for the given project.

5.4. Link delay and power

To assess the impact of global wires, we have studied 65 nm NoC links in isolation from the NoC modules. An overview of the results is shown in Fig. 9. Several factors have to be considered in link design, including obviously length and desired clock frequency. Short or slow-clocked links do not pose problems. However, as either length or target frequency is increased, an undesired rise of power consumption is also observed. The reason is that when links are pushed for high performance, back-end tools automatically insert large amounts of buffering gates, dramatically increasing the energy cost of the links. If frequency or length are pushed even further, the links become infeasible, either because of timing violations or because of crosstalk concerns, i.e., the added buffers would be too large to be deployed without affecting nearby wires. This kind of trade-off among link performance, feasibility and power consumption is crucial to the NoC designer.

Another extremely important dependency we observe is on the specific technology library used. As seen in Section 5.3, especially at the 65 nm node, a single "technology library" is no longer realistic for NoC designs based on standard cells. In fact, manufacturing technologies are spreading across a variety of processes optimized for specific uses, such as, LP or high performance, with several intermediate levels featuring, for example, different threshold voltage values. In this case, if very low-power libraries are used, the size and speed of the buffers interleaved along wires become dramatically inferior, which results in much tighter constraints on frequency of operation or length. Fig. 9(a) reports power consumption for the 65 nm LP-LVT library, while Fig. 9(b) describes the LP-HVT variant. These results show that NoC links implemented using the LP-HVT library are substantially more power-effective, but impose much tighter constraints on link feasibility. Hence, the availability of floorplan-aware and technology-aware high-level design automation tools becomes key to pruning the NoC-based design space and to identifying the best libraries for each design according to its particular constraints.



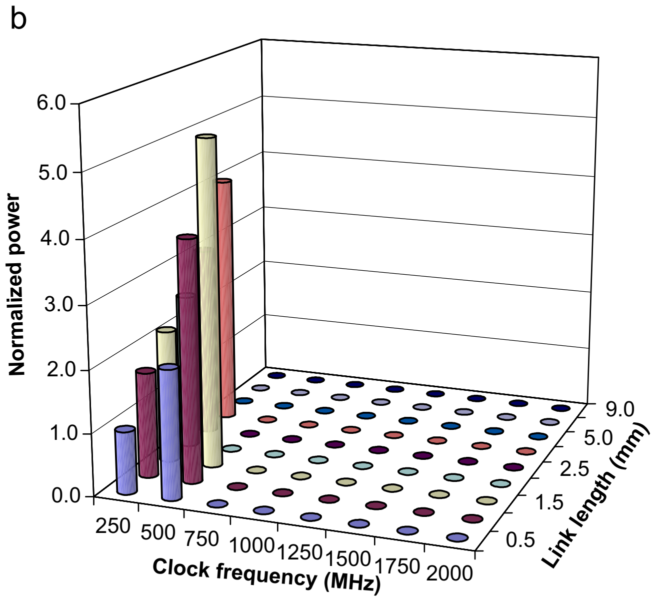


Fig. 9. Power consumption of 38-bit links of varying lengths at different operating frequencies. Values normalized to shortest link at slowest frequency for confidentiality reasons. Missing columns represent infeasible length/frequency combinations. (a) Performance/power oriented 65 nm library (LP-LVT) and (b) very low-power 65 nm library (LP-HVT).

A way to tackle the timing violations on long NoC links, other than just inserting electrical buffers, is link pipelining. Pipeline stages are clocked registers interleaved along the links. By providing one or more extra clock periods to traverse long distances, they solve the link infeasibility problem at a much lower cost than, e.g., that of deploying whole NoC switches in the middle of the links. In some cases, pipelining may even produce more power-effective solutions than regular wire buffering along particularly critical links. However, it incurs a performance cost of one extra cycle of latency. Another major drawback is that NoC flow control must be extended to account for the fact that feedback signals are now coming back after multiple clock cycles instead of in the same clock period. This can be tackled by either deploying deeper buffers at the link endpoints, and using plain registers as pipeline elements, or by pipelining the link with flow control-aware elements, without touching the buffers and logic at the endpoints. The latter approach proves better in our experience [23]. In all cases, since link pipelining affects both the RTL description of the architecture and its latency, the need for higher-level (but technology-aware) CAD tools able to pro-actively accounting for them arises clearly.

5.5. *Wire routability issues in NoCs*

All the issues (e.g., crosstalk) applying to global wires in NoCs also apply, to a smaller extent, to local wires. This means that local wires are increasingly critical too in latest and forthcoming technology nodes. As a result, in wire-intensive components, such as, NoC switches, which are essentially crossbars, it becomes difficult to simultaneously achieve signal integrity, timing closure, and routability (i.e., finding a wire layout in such a way that design rules are respected). As tools automatically try to make wires as straight and short as possible to improve timing, and insert spacing among them to avoid crosstalk, a number of *Design Rule Check* (DRC) violations may occur, including overlapping/shorted wires. Routing tools automatically try to remove DRC violations, for example, by means of SR iterations; the design is virtually split into sub-blocks, and the tools begin trying to resolve routing violations one block at a time. If many violations occur, it is unlikely that all will be automatically fixed in the NoC synthesis flow, so designers have to resort to alternate ways, including:

- Manual intervention on the layout, as in full custom design. Of course this is extremely time-consuming and non-reusable, and is normally only undertaken when the violations in the NoC design are very few.
- Decreasing the row utilization, i.e., spreading the module out into a larger area. Ideally this leaves more space for wire routing, but since it may also affect the output of placement (possibly causing the placement algorithm to diverge from timing closure, as discussed above), this alternative must be experimentally explored in future research. In any case, this approach implies at least an area cost.
- Decreasing the target frequency. Wires are allowed to take less straight paths to their destinations without violating timing constraints, and crosstalk is less of an issue, allowing for tighter wire packing. This strategy is very effective in removing DRC violations in NoC synthesis, but its obvious cost is lower performance.
- Hierarchical floorplanning. This approach tries to better direct the algorithms of the routing tool, by allowing for pre-optimizations and by splitting the problem complexity. Our experience shows that its effectiveness in NoC synthesis depends on the specific module at hand, and must be weighted against the extra design effort at the tool scripting level (usually considerable). Furthermore, hierarchical floorplanning prevents several optimizations that tools can perform on flattened designs. Thus, in the case of NoC switches, this strategy seems to be of limited use in our experience. In fact, if the designer has to manually position even the sub-blocks of switches, just deploying more, smaller switches would require much less effort.

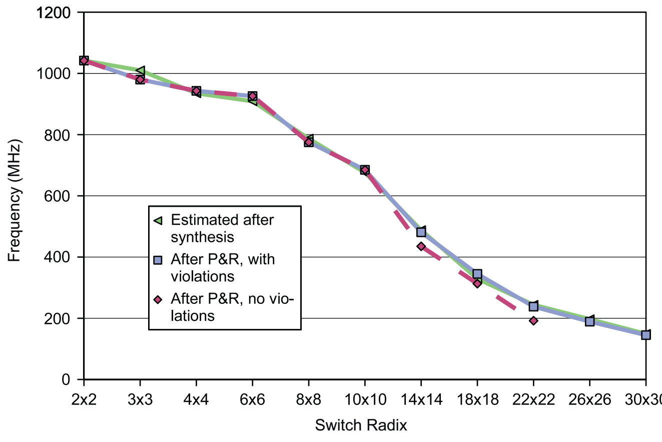


Fig. 10. Frequency achievable by switches of increasing cardinality.

In Fig. 10 we show how frequency scales when implementing ×pipes switches of increasing cardinality in a 65 nm LP-MVT (medium V_T) technology.

The first observation we can draw is that placement-aware synthesis is working as expected for the smaller switches; there are no significant gaps among the timing predictions of Physical Compiler and the timing actually reached by Astro after placement&routing. The most interesting result that we observe, however, concerns switches larger than 10×10 (in 65 nm process technology). The logic synthesis tools are now aware of placement, but not yet of routing. Starting from 14×14, the wire density in the switch crossbars becomes just too high to simultaneously comply with timing objectives, guarantee crosstalk freedom, and resolve DRC violations. Due to the goal priorities we set in our scripts for ×pipes switches, we achieve the former two, but get an increasing amount of the third, ranging from hundreds (14×14) to tens of thousands (30×30). This number of DRC violations in NoC synthesis is clearly unacceptable for manual fixing, and must be tackled automatically. The two possible options for fixing are increasing the switch area or decreasing the switch frequency. The former option proves ineffective; for example, in the 30×30 case, the violations are not fixed even with a final row utilization of 60% (i.e., by leaving 40% of the switch floorplan empty), which is much below typical industrial rates of 85% [1,10]. The latter option gives somewhat better results, making 14×14 and 18×18 switches routable at a 25–30% frequency cost, but still fails on larger switches, even after more than halving the frequency targets. Therefore, our results show the unfeasibility of such NoC switches in latest technology nodes.

As a consequence, even though the DRC violations for some switches of intermediate radix can be fixed, our studies suggest that avoiding large switches altogether may be the best option for NoC implementations in latest technology nodes. This is also due to system-level effects that would result from using large centralized blocks. For example, since many components (NIs or other switches) would be connected to such large switches, floorplan-level congestion would arise: some, if not all, of the other entities would then need to be placed far away from the high-radix switch. In turn, this would require several long links, which would demand aggressive buffering or even pipelining, as discussed above, and therefore bring further performance, area and power costs.

6. Experiments and case studies

In this section we illustrate our analyses of NoC architectures through the application of the proposed NoC-based design flow to several case studies that represent modern multimedia MPSoC architectures. In addition, we compare custom NoC architectures with respect to regular NoC-based solutions, such as different types of mesh-based topologies.

6.1. Experiments on MPSoC benchmarks

We have applied the proposed topology design procedure to six different MPSoC benchmarks, namely, a *Video PROCessor* (VPROC) of 42 cores, an *MPEG4 decoder* (MPEG4) of 12 cores, a *Video Object Plane Decoder* (VOPD) of 12 cores, a *Multi-Window Display* (MWD) application of 12 cores, a *Picture-in-Picture* (PIP) application of 8 cores and an *IMage Processing* (IMP) application of 23 cores. We refer the readers to [43] for the communication characteristics of some of these benchmarks.

For comparison, we generated both custom and mesh topologies for the benchmarks, by modifying the design procedure to synthesize NoCs based on mesh structure. To obtain mesh topologies, we generated a design with each core connected to a single switch and restrict the switch sizes to have 5 input/output ports. We also generated a variant of the basic mesh topology: *optimized mesh* (*opt-mesh*), where those ports and links that are unused by the traffic flows are removed.

The core graph and the floorplan for the custom topology synthesized by our tool for one of the benchmarks (VOPD) are shown in Fig. 11. The network power consumption (power consumption across the switches and links), average hop count and design area results for the different benchmarks are also presented in Table 2. Note that the average hop count is the same for mesh and opt-mesh, as in the opt-mesh only the unused ports and links of the mesh have been removed and the rest of the connections are maintained. The custom topology results

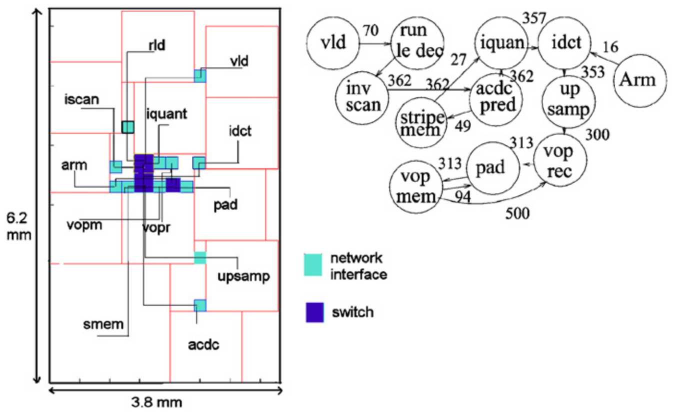


Fig. 11. VOPD custom topology floorplan and core graph.

Table 2
Comparisons with standard topologies

Application	Topologies	Power (mW)	Avg. hops	Area (mm^2)	Time (min)
VPROC	Custom	79.64	1.67	47.68	68.45
	Mesh	301.8	2.58	51.0	
	Opt-mesh	136.1	2.58	50.51	
MPEG4	Custom	27.24	1.5	13.49	4.04
	Mesh	96.82	2.17	15	
	Opt-mesh	60.97	2.17	15.01	
VOPD	Custom	30.0	1.33	23.56	4.47
	Mesh	95.94	2.0	23.85	
	Opt-mesh	46.48	2.0	23.79	
MWD	Custom	20.53	1.15	15	3.21
	Mesh	90.17	2.0	13.6	
	Opt-mesh	38.60	2.0	13.8	
PIP	Custom	11.71	1	8.95	2.07
	Mesh	59.87	2.0	9.6	
	Opt-mesh	24.53	2.0	9.3	
IMP	Custom	52.13	1.44	29.66	31.52
	Mesh	198.9	2.11	29.4	
	Opt-mesh	80.15	2.11	29.4	

in an average of 2.78× improvement in power consumption and 1.59× improvement in hop count when compared to the standard mesh topologies. The area of the designs with the different topologies is similar, thanks to efficient floorplanning of the designs. It can be seen from Fig. 11 that only very little slack area is left in the floorplan. This is because we consider the area of the network elements during the floorplanning process, and not after the floorplanning of blocks. The total runtime of the topology synthesis and architectural parameter setting process for the different benchmarks is presented in Table 2. Given the large problem sizes and very large solution space that is explored (8 different frequency steps, 4 different link widths, 42 cores for VPROC and several calls to the floorplanner) and the fact that the NoC parameter setting and topology synthesis are important phases, the runtime of the engine is not large. This is mainly due to the use of hierarchical tools for partitioning and floorplanning, and

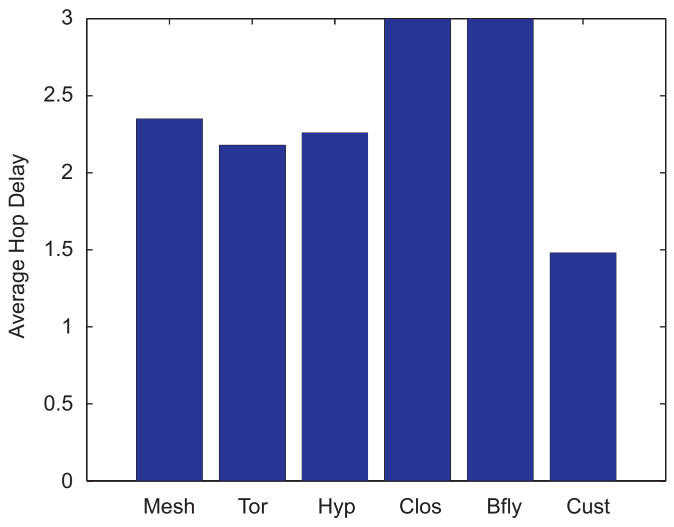


Fig. 12. Performance comparisons of different NoC topologies.

our development of fast heuristics to synthesize the topology in the proposed NoC design flow. Hence, our results illustrate that optimized application-specific NoC synthesis for MPSoCs can be performed in a limited time.

In addition, we have performed comparisons of synthesized topologies against several other standard topologies. For mapping the cores onto the standard topologies, we used the tool from [57]. We optimized the topologies for performance, subject to the design and timing constraints of the target application. The comparisons against 5 standard topologies (mesh, torus, hypercube, Clos and butterfly) for an IMP benchmark with 25 cores are presented in Fig. 12. The custom topology synthesized by our method shows large performance improvements (an average of 1.73×) over the standard topologies.

As an interesting observation, we found that prohibiting certain turns to avoid deadlocks during routing has a negligible impact on the power and performance results for all of the benchmarks. This is because, even if some turns are avoided, the path computation procedure could easily find other paths with low cost, as several alternative low cost paths exist between each source and destination.

6.2. Layout-level comparisons

Using the 130 nm technology library, we had earlier manually developed a NoC design for an MPSoC that runs multimedia benchmarks [8]. 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 inter-processor communication. The hand-designed NoC has 15 switches connected in a 5×3 quasi-mesh network (2 cores connected to each switch), shown in Fig. 13(a). The design is highly optimized, with the private memories being connected to the processors across a single switch and the shared memories distributed around the switches. The layout of the design is presented in Fig. 13(b); as shown, the mesh structure was maintained in the layout. Each of the cores has an area of 1 mm^2 [8] in the design. The entire process, from topology specification to layout generation took weeks because of several feedback loops required to fix several timing and area violations, and to optimize the final design according to the required performance by target multimedia MPSoC. The post-layout NoC could support a maximum frequency of operation of 885 MHz, which is determined by the critical path in the switch pipeline. The power consumption of the topology for functional traffic has been evaluated to be 368 mW.

We applied our topology synthesis process with the objective of minimizing power consumption, to automatically synthesize the NoC for this application. We set the design constraints and the required frequency of operation to be the same (885 MHz) as those of the hand-designed topology. The synthesized NoC topology and the layout are presented in Figs. 13(c) and (d). The synthesized topology has fewer switches (8 switches) than the hand-designed topology. It can support the same maximum frequency of operation (885 MHz), without any timing violation on the wires. As we had taken into account the wire lengths during the synthesis process to estimate the frequency that could be supported, we could synthesize the most power efficient topology that would still meet the target frequency. Moreover, to reach such a design point manually would require several iterations of topology design and place&route phases, which would have been a very time-consuming process.

Layout-level power consumption calculations on functional traffic show that the synthesized topology has 277 mW power consumption, which is 1.33× 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, these savings are 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, NIs) 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 performed 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 Figs. 14(a) and (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 of 11.3% in packet latency. Thus, these results prove that suitable CAD and tooling support can effectively enable competitive NoC synthesis for latest MPSoCs.

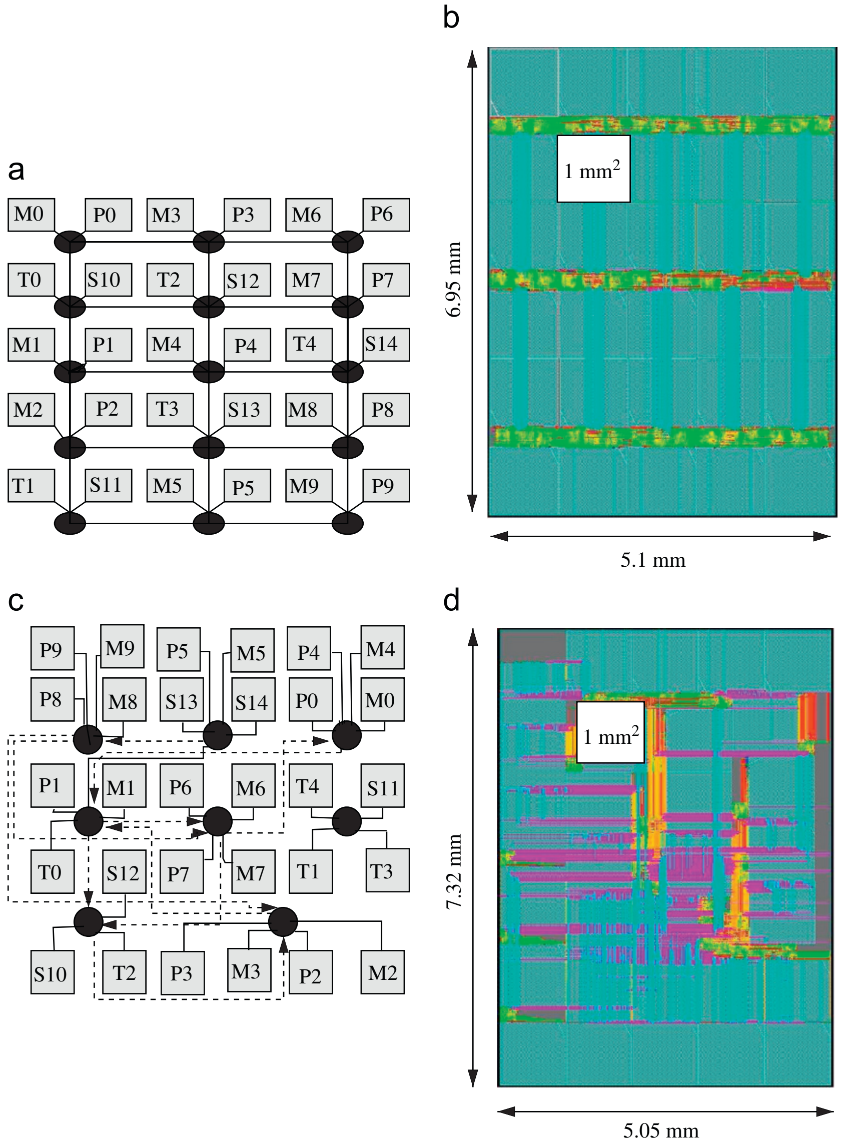


Fig. 13. (a), (b) Hand-designed topology and layout. M: ARM7 processors, T: traffic generators, P, S: private and shared slaves. (c), (d) Automatically synthesized topology and layout. In (c), bi-directional links are solid and uni-directional links are dotted.

6.3. *Impact of frequency constraints*

The maximum frequency of operation that can be supported by the NoC switches depends on the number of switch I/O ports and process technology, as outlined earlier in Fig. 3 for 130 nm process technology and Fig. 10 for 65 nm process technology. Thus, as the required NoC operating frequency increases and the process technology shrinks, the synthesized topologies tend to have switches of lower cardinality. However, this trend needs to be experimentally validated. Thus, in another set of experiments we have studied the impact of the required NoC frequency on the topology synthesis process. We have considered the multimedia MPSoC considered in Section 6.2 and applied the SunFloor tool to synthesize the most power-efficient topology for different operating frequency constraints. The number of switches used in the synthesized topologies for different NoC frequencies is presented in Fig. 15. From this plot we can infer that at low operating frequencies, a topology with few, but large switches results in the most power optimal design. This is due to the fact that the increase in power consumption is mostly linear with the increase in switch size [48]. Thus, in a design with fewer switches, the traffic flows traverse shorter paths, thereby leading to designs with better power consumption. Nevertheless, as the required NoC operating frequency increases, the timing delay constraints cannot be met by large switches (i.e., larger than 10×10), as shown in Fig. 10 (especially for 65 nm process technology), thereby the optimal design point moves to topologies with more switches, with fewer ports. As the proposed tool flow automatically considers the frequency constraint of the switches as well, we are able to prune the infeasible design points, violating timing constraints early in the design process.

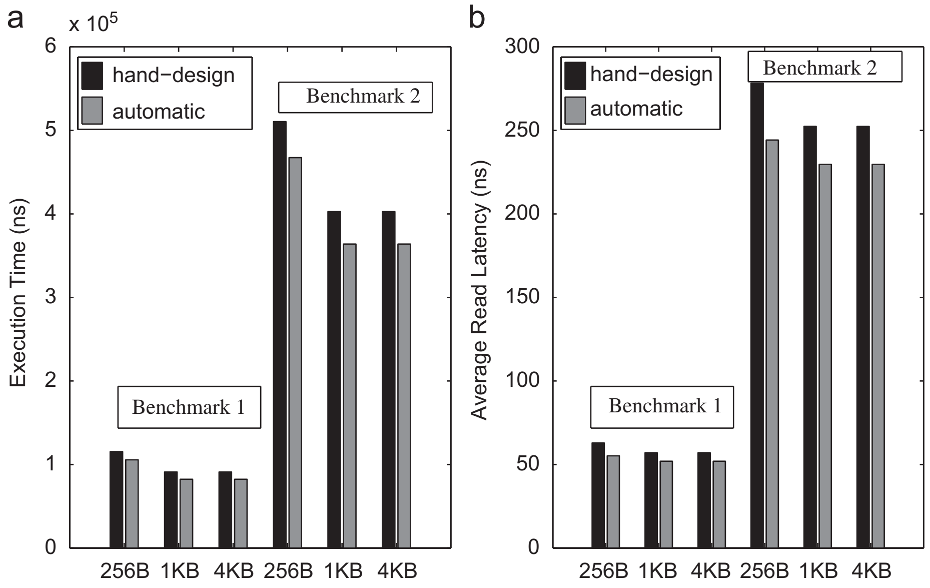


Fig. 14. Runtime and latency of two multimedia benchmarks for different cache sizes. (a) Execution time and (b) average read latency.

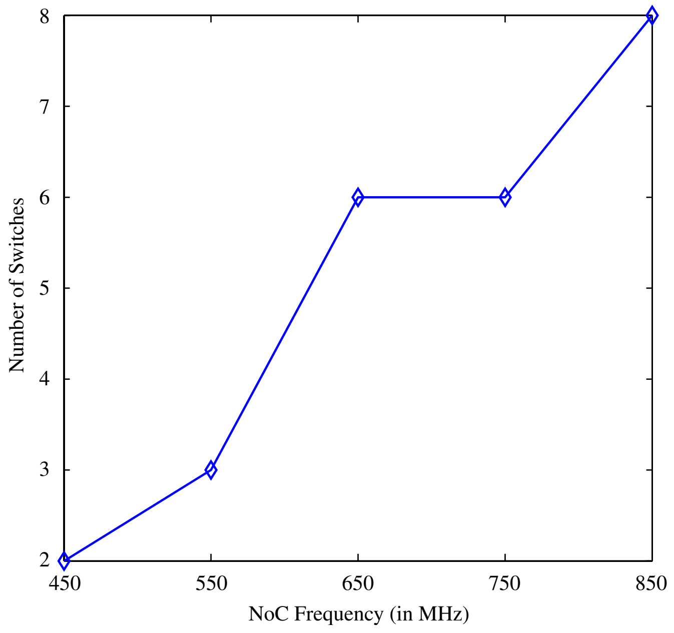


Fig. 15. Topology size variations with NoC frequency.

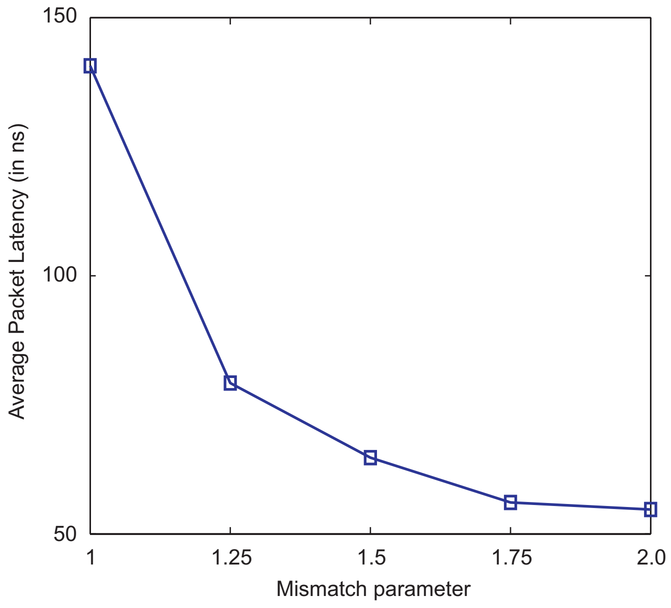


Fig. 16. Impact on latency of the adustment of the mismatch parameter to cope with dynamically-observed congestion.

6.4. Handling dynamic effects

When the designed NoC is simulated, there can be some mismatch between the observed traffic patterns and the initial traffic estimates. This may be either because of inaccurate traffic models or because of dynamic effects, such as congestion. Note that it would be too time consuming to simulate each topology during the synthesis process. Thus, in case the on-chip interconnect requirements are loosely known, to bridge the gap between topology synthesis and simulation, we use the concept of a mismatch parameter in the NoC design flow (see Section 4.5 for more details); in this case, the input traffic rates are multiplied by the value of this parameter. The parameter can be fed as an input to the synthesis engine by the designer. In fact, it is initially set to 1 and the user can manually tune the parameter and re-design the NoC if necessary, until the simulations satisfy the required performance level. The effect of increasing the parameter on performance for the MPEG4 NoC is presented in Fig. 16. These results illustrate that inaccurate estimations of NoC traffic in final MPSoCs can significantly affect the average peak latency of the NoC (variations up to 2.5×). Thus, extensions and suitable use of this new concept to handle localized congestion effects in the NoC are a very challenging problem in future NoC research.

7. Future challenges in NoC design

Despite the large body of research devoted to NoCs in the last years, specially to achieve a mature CAD level to integrate NoC-based interconnects in state-of-the-art design flows for MPSoCs, much remains to be done. Many of the open challenges relate to the seamless integration of physical-design considerations, architectural developments and comprehensive tooling support. For example, several synchronous, mesochronous and asynchronous implementations have been proposed in NoC literature. There is a consensus on the system-level need of integrating blocks at different operating frequencies, which is also often called a *Globally Asynchronous*, *Locally Synchronous* (GALS) paradigm [9]. However, as of today, the dilemma of which NoC alternative to choose in this context has not been fully cleared: synchronous choices (interspersed with a minimum amount of clock converters) have been claiming rapid development times and minimum overhead, while the alternatives claim, for example, superior robustness to process variance [21,27,28]. Furthermore, to the best of our knowledge, none of the contending approaches has yet comprehensively tackled the issue of dynamic frequency (and possibly also voltage) scaling [1,10]; some open questions in this research area include how to best devise mechanisms for NoC partitioning, how to best control the operating parameters of each NoC partition statically or at runtime, etc.

A completely new research field revolves around 3D NoCs [58–60], designed as the backbone for next-generation stacked chips. In this case, much work remains to be done. Moreover, since the manufacturing technology is not fully mature, many crucial design parameters (such as the attainable density, yield and speed of vertical interconnects) remain unclear. Depending on these parameters, many different NoC architectures and topologies can be envisioned. Some of the key upcoming challenges in this field include the potential need for pervasive fault tolerance, the design of a new generation of CAD tools for NoC topology exploration, and the issue of clock domain synchronization (if not even of bridging among different signaling methods) across stack layers.

NoC reconfiguration is also a broad topic [10]. While several approaches have been published to reconfiguring NoCs (either completely, on FPGAs, or just with new routing tables, in ASICs) [36,61], our impression is that many links are missing before fully reconfigurable, hazard- and deadlock-free, low-resource-overhead NoCs can be available. Furthermore, even then, the larger problem of efficiently deciding how to reconfigure NoCs at runtime, based on changing application demands, will represent a challenging problem to be solved.

8. Conclusions

Emerging consumer applications demand a very high level of performance in the next generation of embedded devices. Therefore, new techniques and interconnection mechanisms that can provide solutions for an efficient design of these complex forthcoming embedded architectures are greatly needed. NoCs have emerged as a promising structured way of realizing interconnections on silicon, overcoming the limitations of bus-based solutions. In this paper, we have performed a thorough study of the current state-of-the-art of NoC implementations using a design flow targeting the new trends imposed by deep submicron manufacturing processes. Also, we have presented a comparative analysis of different NoC fabrics ranging from regular topologies to highly tuned custom NoCs.

In this regard, we have illustrated that to have fewer design re-spins and faster time-to-market, design flows for NoC interconnects need to integrate the architectural models with back-end physical design models, thereby bridging a big design gap in NoC synthesis and creating on-chip interconnects free from deadlocks. Moreover, to handle the wiring complexity issue in NoC synthesis, accurate estimations of the interconnect delay and power consumption of the basic building blocks of NoCs early in the design phase are key to produce suitable interconnects for latest MPSoCs. We have also shown that custom NoC topology design, where a NoC is tailored to fit the target application, has noticeable potential benefits with respect to regular topologies, such as, mesh-based NoCs. All in all, the current NoC architectural blocks and design flows for custom NoC topology design have reached a level of maturity comparable to traditional bus-based on-chip interconnects. In fact, the presented design flow is currently able to synthesize a complete NoC-based custom interconnect for a certain MPSoC architecture, starting from a high-level specification, in few hours.

Nevertheless, several research directions still exist to make feasible efficient designs of NoC interconnects for new nano-scale devices and technologies. First, efficient support at the NoC level to handle blocks of MPSoCs working at different operating frequencies or GALS is still an open question. Also, mechanisms for NoC partitioning to provide QoS for new downloaded applications is a very challenging problem. Additionally, the possible benefits of applying runtime dynamic control to NoCs (e.g., dynamic routing schemes) to improve the efficiency of design-time MPSoC configurations is an open question. All these extensions to current NoC designs would require changes in both the topology generation algorithm and architectural implementations. Finally, a novel and very interesting research avenue is the design of 3D NoCs, which target next-generation stacked chips. In this regard, according to the yield and speed of vertical interconnects, many different NoC architectures and topologies can be envisioned, as well as the possible inclusion of various fault tolerant schemes, and the definition of suitable choices for each case are still to be explored. Moreover, the CAD support in this case would become even more critical and we expect a large set of new research challenges for NoC design in this area.

References

[1] W. Wolf, The future of multiprocessor systems-on-chips, in: Proceedings of the 41st Design Automation Conference (DAC'04), June 2004, pp. 681–685.

[2] Philips nexperia—highly integrated programmable system-on-chip (mpsoc), 2004 ⟨http://www.semiconductors.philips.com/products/nexperia⟩.

[3] Texas instruments, TI's omap platform, 2004 ⟨http://focus.ti.com/omap/docs/⟩.

[4] ST Microelectronics, ST nomadik multimedia processor, 2004 ⟨http://www.st.com/stonline/prodpres/dedicate/proc/proc.htm⟩.

[5] N.S. Kim, T. Austin, D. Blaauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, N. Vijaykrishnan, Leakage current: Moore's law meets static power, Computer 36 (12) (2003) 65–77.

[6] L. Benini, G. De Micheli, Networks on chip: a new SoC paradigm, IEEE Comput. 35 (1) (2002) 70–78.

[7] W.J. Dally, B. Towles, Route packets not wires: on-chip interconnection networks, in: Proceedings of Design Automation Conference (DAC'01), ACM, IEEE Press, New York, June 2001, pp. 648–689.

[8] F. Angiolini, P. Meloni, S. Carta, L. Benini, L. Raffo, Contrasting a NoC and a traditional interconnect fabric with layout awareness, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'06), Munich, Germany, ACM, IEEE Press, New York, 2006, pp. 124–129.

[9] A. Jantsch, H. Tenhunen (Eds.), Networks on Chip, Kluwer Academic Publishers, Hingham, MA, USA, 2003.

[10] L. Benini, G. De Micheli (Eds.), Networks on Chips: Technology and Tools, Morgan Kaufmann Publishers, San Francisco, CA, USA, 2006.

[11] J. Hu, R. Marculescu, Exploiting the routing flexibility for energy/performance aware mapping of regular noc architectures, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'03), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2003, p. 10688.

[12] O. Takahashi, S.R. Cottier, S.H. Dhong, B.K. Flachs, J. Silberman, Power-conscious design of the cell processor's synergistic processor element, IEEE Micro 25 (5) (2005) 10–18.

[13] Nexperia media processing, 2003 ⟨http://www.trimedia.com/products/briefs/soc_arch.html⟩.

[14] Arteris, The on Chip Company, Arteris: designing efficient and scalable SoC interconnects, 2005 ⟨http://www.arteris.com⟩.

[15] Silistix, Silistix: chainworks, 2006 ⟨http://www.silistix.com⟩.

[16] iNoCs 2007 ⟨http://www.inocs.com⟩.

[17] ARM Inc. Advanced Microcontroller Bus Architecture (AMBA), AMBA specification, May 1999 ⟨www.arm.com/products/solutions/AMBAHomePage.html⟩.

[18] Amba 3 ahi overview, 2005 ⟨http://www.arm.com/products/solutions/AMBA3AXI.html⟩.

[19] ST Microelectronics, The STBus interconnect, 2001 ⟨http://www.st.com⟩.

[20] D. Wingard, Micronetwork-based integration for socs: 673, in: Proceedings of the 38th Design Automation Conference (DAC'01), New York, NY, USA, ACM, New York, 2001, p. 677.

[21] K. Goossens, J. Dielissen, A. Radulescu, Aethereal network on chip: concepts, architectures, and implementations, IEEE Design Test Comput. 22 (5) (2005) 414–421.

[22] H.G. Lee, N. Chang, U.Y. Ogras, R. Marculescu, On-chip communication architecture exploration: a quantitative evaluation of point-to-point, bus, and network-on-chip approaches, ACM Trans. Des. Autom. Electron. Syst. 12 (3) (2007) 23.

[23] A. Pullini, F. Angiolini, D. Bertozzi, L. Benini, Fault tolerance overhead in network-on-chip flow control schemes, in: Proceedings of the 18th Annual Symposium on Integrated Circuits and System Design (SBCCI'05), New York, NY, USA, ACM, New York, 2005, pp. 224–229.

[24] M.A.A. Faruque, G. Weiss, J. Henkel, Bounded arbitration algorithm for qos-supported on-chip communication, in: Proceedings of the 4th International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS'06), New York, USA, ACM, IEEE Press, New York, 2006, pp. 76–81.

[25] E. Bolotin, I. Cidon, R. Ginosar, A. Kolodny, QNoC: QoS architecture and design process for network on chip, J. Syst. Archit. (2004) 117.

[26] D. Wiklund, D. Liu, Socbus: switched network on chip for hard real time embedded systems, in: Proceedings of the 17th International Symposium on Parallel and Distributed Processing (IPDPS'03), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2003, p. 78.1.

[27] T. Bjerregaard, J. Sparso, A scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip, in: Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'05), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2005, pp. 34–43.

[28] A. Sheibanyrad, I.M. Panades, A. Greiner, Systematic comparison between the asynchronous and the multi-synchronous implementations of a network on chip architecture, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'07), San Jose, CA, USA, EDA Consortium, 2007, pp. 1090–1095.

[29] S. Murali, L. Benini, G. De Micheli, Mapping and physical planning of networks-on-chip architectures with quality-of-service guarantees, in: Proceedings of the 2005 Conference on Asia South Pacific Design Automation (ASP-DAC '05), New York, NY, USA, ACM Press, New York, 2005, pp. 27–32.

[30] A. Pullini, F. Angiolini, S. Murali, D. Atienza, G. De Micheli, L. Benini, Bringing nocs to 65 nm, IEEE Micro 27 (5) (2007) 75–85.

[31] F. Angiolini, P. Meloni, S. Carta, L. Raffo, L. Benini, A layout-aware analysis of networks-on-chip and traditional interconnects for mpsocs, IEEE Trans. Comput.-Aided Design Integrated Circuits Syst. 26 (3) (2007) 421–434.

[32] A. Radulescu, J. Dielissen, K. Goossens, E. Rijpkema, P. Wielage, An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network configuration, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'04), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2004, p. 20878.

[33] A. Andriahantenaina, A. Greiner, Micro-network for soc: implementation of a 32-port spin network, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'03), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2003, p. 11128.

[34] K. Lee, S.J. Lee, S.-E. Kim, H.-M. Choi, D. Kim, S. Kim, M.-W. Lee, H.-J. Yoo, 51 MW 1.6 GHz on-chip network for low-power heterogeneous SoC platform, in: Digest of Technical Papers of the IEEE International Solid-State Circuits Conference (ISSC'04), IEEE Computer Society, Silver Spring, MD, 2004, pp. 152–158.

[35] C. Zeferino, A. Susin, Socin: a parametric and scalable network-on-chip, in: Proceedings of the 16th Symposium on Integrated Circuits and Systems Design (SBCCI), September 2003, pp. 169–174.

[36] T. Marescaux, J.-I. Mignolet, A. Bartic, W. Moffat, D. Verkest, S. Vernalde, R. Lauwereins, Networks on chip as hardware components of an os for reconfigurable systems, in: Proceedings of Field-Programmable Logic and Applications Conference (FPL'03), IEEE Press, New York, September 2003.

[37] A. Pinto, L. Carloni, A. Sangiovanni-Vincentelly, Efficient synthesis of networks on chip, in: Proceedings of 21st International Conference in Computer Design (ICCAD'03), October 2003, pp. 146–150.

[38] W.H. Ho, T.M. Pinkston, A methodology for designing efficient on-chip interconnects on well-behaved communication patterns, in: Proceedings of the 9th International Symposium on High-Performance Computer Architecture (HPCA '03), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2003, p. 377.
[39] A. Hansson, K. Goossens, A. Radulescu, A unified approach to constrained mapping and routing on network-on-chip architectures, in: Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES + ISSS '05), New York, USA, ACM Press, New York, 2005, pp. 75–80.
[40] K. Srinivasan, K.S. Chatha, G. Konjevod, An automated technique for topology and route generation of application specific on-chip interconnection networks, in: Proceedings of the 2005 IEEE/ACM International Conference on Computer-aided Design (ICCAD '05), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2005, pp. 231–237.
[41] T. Ahonen, D.A. Siguenza-Tortosa, H. Bin, J. Nurmi, Topology optimization for application-specific networks-on-chip, in: Proceedings of the 2004 International Workshop on System Level Interconnect Prediction (SLIP '04), New York, USA, ACM Press, New York, 2004, pp. 53–60.
[42] S. Kolson, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, A. Hemani, A network on chip architecture and design methodology, in: Proceedings of IEEE Annual Symposium on VLSI (ISVLSI'02), IEEE Computer Society, Silver Spring, MD, April 2002, pp. 105–112.
[43] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, G. De Micheli, Noc synthesis flow for customized domain specific multiprocessor systems-on-chip, IEEE Trans. Parallel Distrib. Syst. 16 (2) (2005) 113–129.
[44] G. Palermo, C. Silvano, G. Mariani, R. Locatelli, M. Coppola, Application-specific topology design customization for stnoc, in: Proceedings of the 10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD'0707), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2007, pp. 547–550.
[45] D. Siguenza-Tortosa, J. Nurmi, Vhdl-based simulation environment for proteo noc, in: Proceedings of High-Level Design Validation and Test Workshop (HLDVT'02), October 2002, pp. 1–6.
[46] OCP International Partnership (OCP-IP), Open core protocol standard, 2003 ⟨http://www.ocpip.org/home⟩.
[47] G.-M. Chiu, The odd–even turn model for adaptive routing, IEEE Trans. Parallel Distrib. Syst. 11 (7) (2000) 729–738.
[48] P. Meloni, S. Carta, R. Argiolas, L. Raffo, F. Angiolini, Area and power modeling methodologies for networks-on-chip, in: Proceedings of 1st International Conference on Nano-Networks and Workshops, 2006 (NanoNet'06), IEEE Press, New York, September 2006.
[49] Synopsys, PrimeTime ⟨http://www.synopsys.com⟩.
[50] A. Jalabert, S. Murali, L. Benini, G. De Micheli, × pipescompiler: a tool for instantiating application specific networks on chip, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'04), vol. 4, IEEE, New York, February 2004, pp. 1–6.
[51] Synopsys, Design compiler ⟨http://www.synopsys.com⟩.
[52] I.M. Saurabh Adya, Fixed-outline floorplanning: enabling hierarchical design, IEEE Trans. VLSI 11 (6) (2003) 1120–1135.
[53] L. Benini, D. Bertozzi, A. Bogliolo, F. Menichelli, M. Olivieri, Mparm: exploring the multi-processor SoC design space with SystemC, J. VLSI Signal Process. 41 (2) (2005) 169–182.
[54] Synopsys, Astro ⟨http://www.synopsys.com⟩.
[55] S.S.I. Association, The international technology roadmap for semiconductors, Technical Report, 2002 ⟨http://public.itrs.net/⟩.
[56] Synopsys, Physical compiler ⟨http://www.synopsys.com⟩.
[57] S. Murali, G. De Micheli, Sunmap: a tool for automatic topology selection and generation for nocs, in: Proceedings of the 41st Design Automation Conference (DAC'04), New York, USA, ACM Press, New York, 2004, pp. 914–919.
[58] J. Kim, C. Nicopoulos, D. Park, R. Das, Y. Xie, V. Narayanan, M.S. Yousif, C.R. Das, A novel dimensionally-decomposed router for on-chip communication in 3d architectures, in: Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA'07), New York, USA, ACM Press, New York, 2007, pp. 138–149.
[59] F. Li, C. Nicopoulos, T. Richardson, Y. Xie, V. Narayanan, M. Kandemir, Design and management of 3d chip multiprocessors using network-in-memory, in: Proceedings of the 33rd Annual International Symposium on Computer Architecture (ISCA'06), Washington, DC, USA, IEEE Computer Society, Silver Spring, MD, 2006, pp. 130–141.
[60] T. Kgil, S. D'Souza, A. Saidi, N. Binkert, R. Dreslinski, T. Mudge, S. Reinhardt, K. Flautner, Picoserver: using 3d stacking technology to enable a compact energy efficient chip multiprocessor, ACM SIGPLAN Not. 41 (11) (2006) 117–128.
[61] G.M. Link, N. Vijaykrishnan, Hotspot prevention through runtime reconfiguration in network-on-chip, in: Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE'05), vol. 01, Los Alamitos, CA, USA, IEEE Computer Society, Silver Spring, MD, 2005, pp. 648–649.

David Atienza is Associate Professor in the Computer Architecture and Automation Department of Complutense University of Madrid (UCM), Spain, and Post-Doctoral Research Associate at the Integrated Systems Laboratory in EPFL, Switzerland. He received his M.Sc. and Ph.D. degrees in Computer Science from Complutense University of Madrid (UCM), Spain, and Inter-University Micro-Electronics Center (IMEC), Leuven, Belgium, in 2001 and 2005, respectively. He is also Post-Doctoral Research Associate at the Integrated Systems Laboratory in EPFL, Switzerland. His research interests focus on design methodologies for integrated systems, including novel architectures for logic and memories in forthcoming nano-scale electronics, thermal management techniques for multiprocessors system-on-chip, networks-on-chip (NoC) design, and dynamic memory management for embedded systems. In these fields, he has published more than 90 papers in prestigious journals and international conferences: ACM Transactions on Design Automation of Electronic Systems (TODAES), IEEE Micro, IEEE Transactions on VLSI Systems, Elsevier-Integration: The VLSI Journal, Journal of Embedded Systems, Design Automation Conference (DAC), International Conference on Computer-Aided Design (ICCAD), Design, Automation and Test in Europe (DATE) Conference, Asia and South Pacific Design Automation Conference (ASP-DAC), etc. Also, he is Associate Editor of IEEE Transactions on Computer-Aided Design of Circuits and Systems (TCAD), and part of the Technical Program Committee of the DATE, ICCAD and VLSI-SoC conferences. Dr. Atienza is an elected Executive Member of the IEEE Council of Electronic Design Automation (CEDA) since 2008.

Federico Angiolini received the M.Sc. degree (summa cum laude) in electronic engineering from the University of Bologna, Italy, in 2003, and is now a Ph.D. student in the Department of Electronics and Computer Science at the same university and a visiting student at the Ecole Polytechnique Federale de Lausanne (EPFL), Switzerland. His research interests are in the domain of multiprocessor embedded systems, for which he has been investigating memory hierarchy optimizations, cycle-accurate simulation environments and especially interconnection architectures, with a focus on networks-on-chips, spanning from design tools to implementation and physical design.

Srinivasan Murali is currently a post-doctoral researcher at the LSI Lab, EPFL, Switzerland. He received his Ph.D. and M.Sc. in Electrical Engineering at Stanford University in 2007. He received his Bachelor's degree in Computer Science and Engineering from the University of Madras, India, in 2002. His research interests include reliable and efficient design methods for networks-on-chips and systems-on-chips. He has authored more than 30 technical articles in these fields. Dr. Murali was a recipient of a Best Paper Award in the DATE Conference in 2005.

Antonio Pullini is a research assistant with the EDA group at the Politecnico di Torino, Italy. He is currently also a visting student at EPFL, Lausanne. His research interests include all aspects of low-power digital design and networks-on-chip with particular emphasis on physical design and CAD methodologies. Pullini has an M.Sc. degree in electrical engineering from the University of Bologna, Italy.

Luca Benini received the B.S. degree (summa cum laude) in electrical engineering from the University of Bologna, Italy, in 1991, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University in 1994 and 1997, respectively. He is currently a Full Professor in the Department of Electronics and Computer Science in the University of Bologna. He also holds a visiting professor position at EPFL. Dr. Benini's research interests are in all aspects of computer-aided design of digital circuits, with special emphasis on low-power applications, and in the design of portable systems. On these topics, he has published more than 200 papers in international conferences and journals, and he is co-author of three books. Dr. Benini is a member of the technical program committee for several technical conferences, including the Design Automation Conference (DAC), the International Symposium on Low Power Design (ISLPED) and the International Symposium on Hardware–Software Codesign (CODES/ISSS). He has been the General Chair of the 2008 Design, Automation and Test in Europe (DATE) Conference and the Program Chair of the 2005 DATE Conference. He is Associate Editor of the IEEE Transactions on Computer-Aided Design of Circuits and Systems and of the ACM Journal on Emerging Technologies in Computing Systems. He is a Fellow of the IEEE.

Giovanni De Micheli is Professor and Director of the Integrated Systems Centre at EPFL, Switzerland, and President of the Scientific Committee of CSEM, Neuchatel, Switzerland. Previously, he was Professor of Electrical Engineering at Stanford University. He is author of Synthesis and Optimization of Digital Circuits, McGraw-Hill, 1994, co-author and/or co-editor of six other books and of over 300 technical articles. He is a Fellow of the ACM and IEEE. Prof. De Micheli is the recipient of the 2003 IEEE Emanuel Piore Award for contributions to computer-aided synthesis of digital systems. He received the 1987 D. Pederson Award for the best paper on the IEEE Transactions on CAD/ICAS, two Best Paper Awards at the Design Automation Conference, in 1983 and in 1993, and a Best Paper Award at the DATE Conference in 2005. His research interests include several aspects of design technologies for integrated circuits and systems, such as synthesis, hardware/software codesign and low-power design, as well as systems on heterogeneous platforms including electrical, optical, micromechanical and biological components.