# Modeling DVFS and Power-Gating Actuators for Cycle-Accurate NoC-Based Simulators

DAVIDE ZONI and WILLIAM FORNACIARI, Politecnico di Milano

*Networks-on-chip* (NoCs) are a widely recognized viable interconnection paradigm to support the multicore revolution. One of the major design issues of multicore architectures is still the power, which can no longer be considered mainly due to the cores, since the NoC contribution to the overall energy budget is relevant. To face both static and dynamic power while balancing NoC performance, different actuators have been exploited in literature, mainly *dynamic voltage frequency scaling* (DVFS) and power gating. Typically, simulation-based tools are employed to explore the huge design space by adopting simplified models of the components. As a consequence, the majority of state-of-the-art on NoC power-performance optimization do not accurately consider timing and power overheads of actuators, or (even worse) do not consider them at all, with the risk of overestimating the benefits of the proposed methodologies.

This article presents a simulation framework for power-performance analysis of multicore architectures with specific focus on the NoC. It integrates accurate power gating and DVFS models encompassing also their timing and power overheads. The value added of our proposal is manyfold: (i) DVFS and power gating actuators are modeled starting from SPICE-level simulations; (ii) such models have been integrated in the simulation environment; (iii) policy analysis support is plugged into the framework to enable assessment of different policies; (iv) a flexible GALS (*globally asynchronous locally synchronous*) support is provided, covering both handshake and FIFO re-synchronization schemas. To demonstrate both the flexibility and extensibility of our proposal, two simple policies exploiting the modeled actuators are discussed in the article.

Categories and Subject Descriptors: B.1.2 [**Hardware**]: Control Structure Performance Analysis and Design Aids—*Simulation*

General Terms: Performance, Design

Additional Key Words and Phrases: Network-on-chip, performance, power, design aids, simulation

**ACM Reference Format:**
Davide Zoni and William Fornaciari. 2015. Modeling DVFS and power-gating actuators for cycle-accurate NoC-based simulators. ACM J. Emerg. Technol. Comput. Syst. 12, 3, Article 27 (September 2015), 24 pages.
DOI: http://dx.doi.org/10.1145/2751561

## 1. INTRODUCTION

Multicore integration is becoming a popular solution to deliver increasing processing power to demanding applications, both for embedded and high-end computing, while proving a good control over the power consumption. The trend to integrate multiple cores onto a single chip and the limited performance of traditional bus-based solutions make the *network-on-chip* (NoC) a viable multicore interconnect paradigm [Banerjee et al. 2007]. Although transistor density still follows the Moore's law, current VLSI designs present several challenges hard to overcome. In particular, balancing performance and power still represents a first-class standing design issue. This aspect is quite severe and motivates ad-hoc optimizations encompassing also the NoC, since its power



This work is partially supported by EU-FP7-612069-HARPA.
Authors' addresses: D. Zoni (corresponding author), DEIB – Politecnico di Milano, Via Ponzio 34/5, 20133 Milano, Italy; email: davide.zoni@polimi.it.

DOI: http://dx.doi.org/10.1145/2751561

consumption ranges from 10% up to 28% of overall chip power [Hoskote et al. 2007] depending on the complexity of the cores, the memory hierarchy, and the architecture of the routers.

Srinivasan [2011] shows that leakage power represents a real threat for future designs, since it is predicted that it will increase by 5× for each new technology node while dynamic power will remain roughly constant. Since NoCs significantly influence overall multicore performance, proper power-performance NoC design methodologies, addressing dynamic and static power as well as performance, must be investigated to achieve an optimal balance of such conflicting metrics.

In this scenario the exploration of different knobs, that is, actuators to manage both power and performance, is strongly required. *Dynamic voltage and frequency scaling* (DVFS) is a popular technique widely exploited both in *systems-on-chip* (SoCs) and *chip multiprocessors* (CMPs) to dynamically adjust voltage and frequency of the *processing elements* (PEs). The objective is trading power and performance while meeting application demands, pushing from the bottom of the hardware. Several proposals argued the use of DVFS in the NoC to fulfill the communication requirements while saving power. For example, Chen et al. [2013] propose different theoretical control techniques to exploit the DVFS in multicores. Mishra et al. [2011] present a DVFS-based scheme for NoC which adapts the router frequencies depending on the actual network load. Our work is different from such proposals, since we mainly focus on the simulation framework instead of the policy development. Moreover, our solution allows to easily validate different power-performance policies. The possibility to dynamically change the frequency to different communicating blocks requires a suitable design to support signal resynchronization. This allows to avoid metastability issues and to guarantee signal integrity. The *globally asynchronous locally synchronous* (GALS) design scheme allows to partition the design into different so-called *voltage and frequency islands* (VFIs) considering signal resynchronization on VFI boundaries. Moreover, a GALS design scheme represents a suitable way to manage clock distribution to limit power consumption in SoC and CMPs [Ogras et al. 2007]. Traditional *very large-scale integrated* (VLSI) designs require a totally balanced clock distribution network to ensure minimal clock skew between communication logic blocks. However, such networks can consume up to 30% of the overall chip power [Alhussien et al. 2010]. NoCs naturally fit the GALS design paradigm by organizing groups of routers in different VFIs, where the size of each VFI is the knob to balance power and performance. *Power gating* (PG) [Chowdhury et al. 2008] is another exploited actuator to face the excessive leakage power consumption since, to save power, it allows to switch a part of the logic off when in an idle state.

The need to consider different orthogonal design metrics (e.g., power and performance), complex multicore architectures, and different actuators (DVFS and power gating) highlights cycle-accurate simulation as the suitable vehicle to ensure fast architectural design, accurate modeling, and flexibility in the design-space exploration. While different simulators integrating both power and performance analysis have been proposed in literature, to the best of our knowledge no proposal exists focusing on a complete simulation framework integrating the models of the actuators coupled with the power and performance simulation of the architecture.

This article addresses such a scenario by proposing a complete simulation framework tailored to design power-performance optimizations for NoCs. To enhance the value of the analysis, it integrates and exposes to the designers a wide range of knobs, including accurate DVFS, power gating actuator models, as well as a GALS design schema with two resynchronization strategies. Our work aims to achieve relevant impact and visibility in the research community working on accurate power-performance analysis and is available in Zoni and Fornaciari [2015].

Table I. State-of-the-Art on multicore Simulator

| Framework | Cycle-accurate CPU+NoC | NoC support | Power support | GALS support | DVFS/DFS projection | PLL/divider exploration | Objectives |
|---|---|---|---|---|---|---|---|
| Renau et al. (SESC) [Renau et al. 2005] | ✓ | | | | | | multicore simulation, parallel applications |
| Soteriou et al. (Polaris) [Soteriou et al. 2006] | | ✓ | ✓ | | | | Network-on-Chip design-space exploration |
| Hsieh et al. (SST) [Hsieh et al. 2011] | | ✓ | ✓ | | | | microarchitecture, power and thermal |
| Lis et al. (HORNET) [Lis et al. 2011] | | ✓ | ✓ | ✓ (simple) | | | many-core processors, mainly NoC interconnect |
| Bartolini et al. [Bartolini et al. 2010] | | ✓ | ✓ | | ✓ | | run-time control policies for multicores |
| Zoni et al. (HANDS) [Zoni et al. 2012; Corbetta et al. 2012] | ✓ | ✓ | ✓ | | | | power/perf, reliability/perf for multicores |
| Carlson et al. (Sniper) [Carlson et al. 2011] | ✓ | ✓ | ✓ | | ✓ | | distributed parallel simulator for multicores |
| Prabhu (Ocin tsim) [Prabhu et al. 2009] | | ✓ | | | ✓ | | test different DVFS schemes for NoC |
| *Our Proposal* | ✓ | ✓ | ✓ | ✓ (accurate) | ✓ | ✓ | CPU+NoC, PLL/divider, DFS, GALS design |

Features, advantages, and drawbacks with focus on GALS and DFS support for NoC. The GALS support, when present, does not include accurate resynchronization models.

### 1.1. Novel Contribution

Starting from publicly available research tools, we built up a multicore simulation framework providing accurate models for both DVFS and power gating actuators. Moreover, the GALS design paradigm is supported through two different resynchronization schemes: FIFO and handshake. In such a way it is enabled a flexibly partitioning of the chip into multiple VFIs to aggressively optimize power and performance. In particular, the main contributions of the work are detailed here.

—*Complete, Flexible and Extensible Framework.* The proposed framework supports exploration of different multicore design metrics, providing accurate models for the actuators. Different simulation frameworks have appeared in literature and Table I highlights their limitations as well as the new features added by this work.
—*Models of the Actuators and GALS Support.* The DVFS model sits on an accurate SPICE-level PLL analysis. Furthermore, a delay model for the voltage regulator is built starting from SPICE simulations of a commercial component [Linear Technologies 2013]. The final DVFS accurately models timing aspects as well as worst-case

power consumption. The GALS support implements two resynchronization schemes, that is handshake [Alhussien et al. 2010] and FIFO [Miro Panades and Greiner 2007], allowing to trade area and power consumption over performance penalties. The power gating support is introduced starting from SPICE-level simulations to equip the simulation flow with accurate timing and power sleep transistor models. Accuracy of the models for the actuators is crucial to enable further investigations related to reliability. For example, Calimera et al. [2009a] and Zoni and Fornaciari [2013] use power gating to counteract aging due to *negative bias temperature instability* (NBTI) while Calimera et al. [2009b] model the NBTI on sleep transistors of the power networks to assess their performance degradation over time. However, the proposed models are flexible enough to balance accuracy and simulation performance, as explained in Sections 3.2 and 4.3.

—*Runtime Policy Assessment.* A lightweight policy module interface allows to write novel management strategies exploiting DVFS and power gating in the NoC routers. The granularity, that is, the size of VFIs, can be chosen by the designer. By allowing access to runtime power and timing estimates for different router logic blocks, the policy can actuate on frequency and voltage of the controlled IP blocks at runtime. Moreover, power gating support is offered to buffers and crossbar only inside each router. To this extent, we focus on the most leaky components from the static power viewpoint, without affecting the routing function that must otherwise be revised if it were possible to completely switch a router off.

### 1.2. Article Organization

The rest of the article is structured in six sections. Section 2 is an overview of the state-of-the-art on simulation frameworks, focusing on DFS and power gating design. Section 3 describes the proposed models for the actuators and the two resynchronization schemas. Results are then reported and discussed in Section 4, mainly to show the possibility of analysis and design-space exploration offered by our framework, while some final remarks are drawn in Section 5.

## 2. RELATED WORK

This section is organized in two parts. First of all, we discuss state-of-the-art related to simulation frameworks supporting DVFS and GALS, with particular emphasis on the NoC level. The second part discusses the literature on power gating network design and how it can be employed to cope with static power consumption.

### 2.1. Dynamic Voltage Frequency Scaling (DVFS)

Several proposals can be found in literature to aid designers during early stages of platform definition, but only few of them are specifically tailored to support power-performance trade-off analysis in multicore scenarios, considering GALS NoC or DVFS support. *Wattch* [Brooks et al. 2000] constitutes the first cycle-accurate single-core power-performance simulator. However, the advent of parallel architectures requires simulation toolchains allowing to accurately mimic the behavior of multicores, also considering GALS and DVFS features. In this perspective, *SESC* [Renau et al. 2005] provides cycle-accurate simulation of bus-based multicore processors, based on the MIPS architecture. However, it does not support networks-on-chip and neither DVFS nor GALS NoC design. The *Polaris* framework [Soteriou et al. 2006] allows power and area design-space exploration for network-on-chip architectures without considering a detailed power estimation for both processors and memory hierarchy. Moreover, it does not implement a heterogeneous NoC model to account for dynamic frequency changes during simulation. Lis et al. [2011] is meant to simulate large-scale architectures and exploit parallel simulation on physical hardware, with particular emphasis on the

on-chip network. While the framework enables power-performance trade-off analysis, it lacks a complete GALS on-chip network model, thus simulation of DVFS-based policies is not possible. The work in Carlson et al. [2011] presents *Sniper*, a framework that can simulate a multicore underpinned by an on-chip network interconnect. The simulator provides per-core DVFS, while such support is not present for the NoC. Moreover, it does not model accurate resynchronizers to enable GALS design. Such a lack, as we demonstrate in this article, can lead to poor accuracy. The HANDS [Zoni et al. 2012; Corbetta et al. 2012] framework sits on GEM5 and allows to simulate multicore architectures while collecting power-performance thermal and reliability estimates at the same time. Even if accurate, still it lacks a comprehensive GALS NoC model, thus it is not possible to test different DVFS schemes to trade off power versus performance.

Coming to a specific literature regarding GALS NoCs and DVFS, Ogras et al. [2007] propose a design methodology for partitioning a NoC architecture into multiple voltage and frequency islands and assigning supply and threshold voltage levels to each VFI. The employed resynchronization scheme is based on FIFO buffers.

In Beigne et al. [2008] is presented a complete DVFS scheme for IP unit integration to be employed for NoC-based design. However, their work focuses on demonstrating the achievable benefits of DVFS in NoCs, while our proposal aims at architectural exploration.

A SystemC/TLM-based DVFS model for NoCs has been proposed in Lebreton and Vivet [2008]. While it allows to mimic the interaction of a DVFS component inside a NoC-based multicore, it differs from our proposal in two main aspects. First, it is not intended for novel DVFS-based control policies validation. Moreover, it is not focused on application simulation considering the support for a Linux-based operating system, while our proposal sits on a GEM5 simulator that is capable of both full system as well as bare metal simulations.

Ducroux et al. [2013] proposed a power modeling approach of a complex manycore system considering the STHORM 16-core platform as a case of study. Although the proposal is interesting, our work addresses different aspects of the same research field. In particular, we focus on exploration and optimization considering the platform as completely customizable without being constrained by the real hardware. Moreover, Ducroux et al. [2013] do not explicitly focus on accurate modeling of the actuators using a block box model to obtain power estimates.

Thonnart et al. [2010] discussed a fully asynchronous low-power framework to design NoCs at RTL level, overcoming some CAD tool limitations due to the use of a *Quasi-Dealy insensitive* (QDI) asynchronous logic [Martin and Nystrom 2006]. While Thonnart et al. [2010] focus on the implementation details, our work aims at providing a complete cycle-accurate simulation solution supporting early design stages.

### 2.2. Power Gating

Dynamic power represents a hot research topic as detailed in Section 2.1. However, leakage power is receiving increasing attention since its contribution is becoming the first source of dissipation in current and future nanometer VLSI devices. This section provides an overview of state-of-the-art related to power gating methodologies to reduce static power consumption and a review of frameworks to support power gating network design. There exist several tools to aid designers during early-stage platform definition, specifically focused to support power-performance trade-off analysis in multicore scenarios.

CACTI-P [Li et al. 2011] is an architecture-level integrated power, area, and timing modeling framework for SRAM-based structures with advanced leakage power reduction techniques. In particular, CACTI-P is claimed to design optimal leakage reduction

power gating networks. Unfortunately, its power model is not directly bound to a cycle, accurate simulator, thus major modifications should be introduced accurately to support power gating, that is, stall simulation only on specific blocks.

Li et al. [2013] propose another architecture-level integrated power, area, and timing modeling framework for multicore that integrates CACTI-P [Li et al. 2011]. It produces area and power estimates for different multicore components (i.e., CPU, buses, and NoC) for different technology nodes, providing power gating support. However, such features cannot be easily exploited since CACTI-P is basically a power model, that is it is not included in a performance simulator, but rather, used as a post-processing analysis tool without easily offering the possibility to evaluate power gating policies.

Kahng et al. [2009] presented Orion2.0, an accurate and flexible power and area model to be used for NoC design. A stripped version of the proposed model has been revised and integrated in the GEM5 simulator, providing accurate power and performance estimates. However, it does not support power gating network design. Kahng et al. [2012] introduce Orion3.0, a totally novel power model for the NoCs with respect to Orion2.0. It sits on synthesizable RTL router models to increase result accuracy. While power and area estimates are closer to the modeled real hardware components, Orion3.0 lacks accurate power gating support.

Bolzani et al. [2009] present a layout-oriented synthesis flow which integrates both power and clock gating to aggressively reduce both leakage and switching power consumption, respectively. They enhance current synthesis tools to automatically support insertion of the additional logic to implement these two techniques in the final design, starting from the device netlist. The proposal focuses on an industrial design flow starting from the device netlist, whereas ours targets early-stage exploration in multicore architectures.

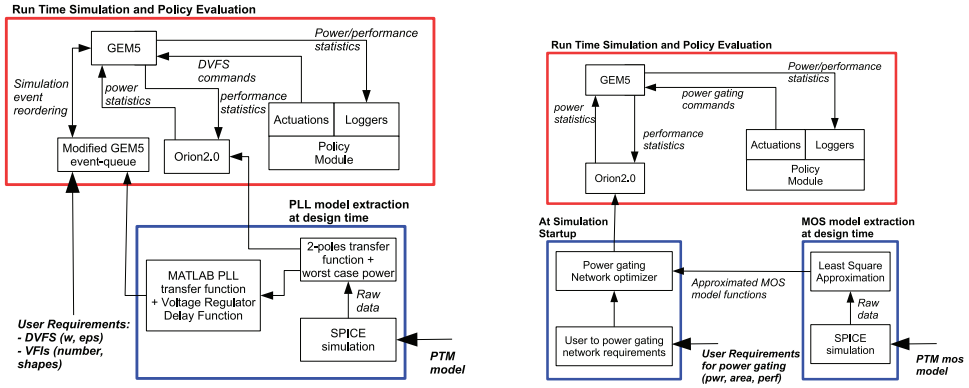
Aside from the proposed frameworks to support power gating, a lot of methodologies have been proposed to limit leakage power, testifying to the crucial importance of such a problem. A micro-architectural-based solution to reduce leakage power in CPU functional units has been presented in Hu et al. [2004]. In particular, their research tries to identify suitable idle patterns where it is possible to switch off the logic. The proposed framework simplifies such evaluation, since a proper logger module has been plugged in the cycle-accurate simulator, enabling an integrated evaluation of the power gating policies under analysis.

Agarwal et al. [2006] present a method to trade off leakage reduction considering device performance. In particular, it is possible to employ several voltage levels greater than zero to switch off the gated circuit, with different levels of leakage power. However, the gated circuit requires significantly less time to switch from OFF to the ON state.

Power gating actuators have been used also to face NBTI aging issues in NoC components [Zoni and Fornaciari 2012, 2013]. The works provide different policies to face NBTI issues employing power gating to switch off and on the logic blocks to be recovered. In particular, it has been demonstrated that a proper power gating component allows to significantly mitigate NBTI while preserving performance and reducing the risk of permanent faults.

## 3. PROPOSED ESTIMATION FLOW

The proposed simulation flow sits on a set of publicly available tools that are used as basic building blocks. GEM5 [Binkert et al. 2011] is an event-driven simulator for NoC-based multicores. Orion2.0 is used as the NoC power model. This section discusses the DVFS and power gating actuators which have been integrated into these tools, as well as the GALS support. Figure 1 reports the information flow between various components of the simulation framework to highlight the integration of both power gating and DVFS actuators. Although the two mechanisms can be used at the




(a) DVFS (b) power gating

Fig. 1. Information flow between all the tools to provide DVFS and power gating support to the NoC.

same time on the same router, the figure has been split into two pictures to improve readability. Both Figure 1(a) and Figure 1(b) put in evidence runtime (red boxes) and design-time (blue boxes) aspects.

A *policy module* is present in the runtime box of both figures to stress the possibility of reading power and performance data and of returning to the simulator engine the driving of the frequency/voltage/power gating actuators on a per-router basis.

Figure 1(a) details the (DVFS information flow) inside the simulation framework. Starting from the *predictive technology models* (PTM) [Zhao and Cao 2006], a 45nm PLL SPICE circuit has been developed to extract the worst-case power and timing. The power information is added to Orion2.0, while the timing data is used to develop a MATLAB-based PLL model which has been implemented in GEM5. Such a model is fully characterized by exploiting the user information provided for each simulated scenario, reporting the percentage of overshooting and the transient response time of the PLL; see Section 3.2 for a complete description of the configurable parameters. Moreover, the event management system of the simulator has been extended to support the possibility of moving already scheduled events between different simulation times, a feature required to implement DFS behavior. In particular, we needed a framework capable of grouping all those events scheduled for a single component and move them forward or backward with respect to the actual scheduled time. Changing the frequency of a component entails moving already scheduled events to the new time they will need to be processed, considering the frequency change, and storing the new frequency value in such a way that all subsequently generated events will be scheduled at the appropriate time. Both the VFI partitions as well as resynchronizers are not directly observable in the information flow, since they are integrated into GEM5 without exchanging information with the rest of the simulation framework. Finally, we considered a delay model for the voltage regulator to be used to mimic the need to raise the voltage before increasing the frequency. The worst-case power values for the voltage regulator have been taken from the commercial LTC3589 [Linear Technologies 2013]. In such a way we can accurately mimic the DVFS actuator. It is worth noticing that neglecting to accurately consider the dynamics of the voltage regulator does not invalidate the DVFS model, since PLL dynamic behavior always dominates the DVFS response. In particular, when a frequency increase is required, the voltage regulator intervenes in advance to properly set the voltage in order to support the new frequency that the PLL is imposing. On the contrary, when the frequency is moved down the PLL immediately starts decreasing the frequency, while the voltage will follow without

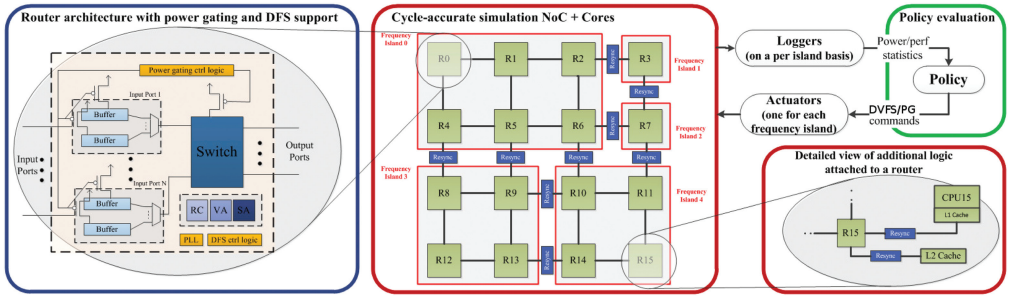


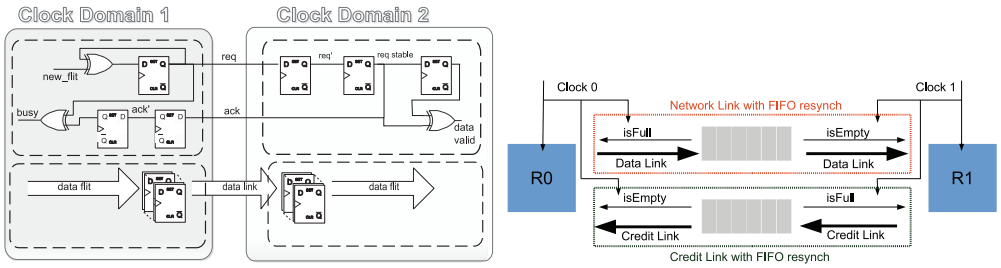

Fig. 2. Logic view of the proposed simulation toolchain.

imposing any additional delay. For the sake of conciseness, avoiding a discussion on such model in a specific section, we assumed a voltage regulator delay of 5us and a worst-case power consumption of 2.5mA at 1V when considering a single router, as observed during SPICE-level simulation of the LTC3589 component [Linear Technologies 2013].

Power Gating information flow is depicted in Figure 1(b). Similarly to the DVFS, the different modules are properly executed at runtime, at compile time, or during runtime initialization in order to minimize the simulation overhead. Again, the policy module can send power gating signals on a per-router basis using the provided power and performance information received by the simulation engine. The SPICE-based sleep transistor MOS characterization module allows to derive approximate while accurate functions capturing MOS parameters, starting from the SPICE PTM for different technology nodes. Such functions allow to skip all SPICE executions during the runtime simulation stage. The power gating network design and optimization module is tailored to design the power network around specific micro-architectural blocks, allowing to specify additional constraints on the optimization procedure. In such a way, it is possible to derive a power network for different components inside the multicore architecture that have different requirements and goals. The cycle-accurate power-performance simulator module simulates multicore architectures augmented with all the designed power gating networks for each selected multicore logic block. This choice makes possible to evaluate power consumption, and eventually the saved leakage power due to gated components. Timing information and overheads can also be obtained.

The overall framework capability to simulate different multicores is depicted in Figure 2. The middle of the figure displays a 2D mesh multicore partitioned into multiple VFIs (five), each with a different shape, to put in evidence the flexibility of our proposal. The interconnection between each VFI pair is regulated by a resynchronization module, that is, the blue box in Figure 2. Moreover, each router has one L2 bank connected and at least one core, even if multiple cores per router are possible. It is worth noticing that both the L2 and cores are not considered part of the VFI on any router, thus a resynchronizer is used to manage their connection to the NoC. The green box on the top-right side of Figure 2 highlights the policy which is the baseline component used to manage voltage, frequency, and power gating actions based on the logged values. Last, the leftmost part of Figure 2 shows the modified router microarchitecture, where DVFS has been added on a per-router basis while power gating support is added on buffers and crossbar only due to their high static power consumption.

The rest of this section discusses all modifications to the flow, focusing on each main component. Resynchronizer and DFS aspects are addressed in Sections 3.1 and 3.2, respectively, with some details on the multistep PLL model provided in Section 3.3. Section 3.4 presents parameter estimation for the baseline header sleep transistor, and




(a) handshake (b) FIFO

Fig. 3. Logical view of the two proposed resynchronization schemas.

the approach to design a power network supporting power gating is the objective of Section 3.5.

### 3.1. Resynchronization Method

When different parts of a complex digital system are working at different clocks or at the same frequency but in different phases, an interface is required to ensure reliable data transfer as well as to prevent metastability. The proposed framework allows to select different granularities for the size of frequency islands in terms of number of routers. With the goal of improving flexibility in the analysis of design alternatives, the simulation flow includes models for the two resynchronization strategies: *FIFO* and *handshake*. The objective is to allow the designer to evaluate the impact of adopting high-throughput solutions (FIFO) with respect to slower but less expensive approaches from an area/power standpoint (handshake). Without loss of generality, the discussion refers to a pair of routers considered as a couple of asynchronous systems. Actually, the proposed simulation flow places a resynchronizer at the boundary of each frequency island. Since we are focusing on a flexible GALS NoC design, a resynchronizer is in between each router and its computing-related components, namely cores, memory controllers, and cache controllers. Although this article focuses only on NoC, its extension to cover the support of DVFS and power gating for CPUs and memory is straightforward; in fact, the models are the same and the resynchronizer between CPUs/caches and routers is already in use.

*3.1.1. Handshake Resynchronizer.* The handshake resynchronizer represents a low-cost and -performance solution with respect to the FIFO schema, mainly due to the absence of storage elements. Starting from the work in Alhussien et al. [2010], the framework implements a 2-way handshake protocol which adds only two single bit-lines—request and acknowledge—to a network link. Note that the same circuit has to be implemented for the credit link. Figure 3(a) shows the main logic blocks of the model considering two clock domains, for example, two routers. In particular, the left side of the Figure 3(a) reports the output port interface of the upstream router, while the right side depicts the input port interface of the downstream router. When a new flit is ready to be sent out, the upstream router asserts the *new_flit* signal for one clock cycle. This forces to toggle the *req* signal one cycle later. Moreover, the back path in the upstream router switches the *busy* signal to high. After the propagation delay, the *req* signal enters into a 2 flip-flop chain in the downstream router that is used to prevent metastability. The third flip-flip in the chain is used jointly with the *req_stable* signal as a front detector, since our resynchronizer operates on fronts to increase throughput [Alhussien et al. 2010]. The front detector asserts the *data_valid* line, signaling that there is a valid flit on the link. The *req_stable* signal is also sent back as an acknowledge to the upstream router to report the data transfer completion. Similarly, the upstream router manages

the *ack* signal using a 2-flip-flip chain to avoid metastability. The *busy* signal is used to prevent transmission of new flits until reception of the acknowledge signal.

The handshake resynchronizer requires at least four cycles for each transmission, with a power overhead lower than 1mW at full load, considering 6-millimeter link length between two routers and an accurate floorplan generated from Alpha21364 estimates [Corbetta et al. 2012]. The power is evaluated using Orion2.0, since the power consumption due to the resynchronization links fairly dominates that of the resynchronizer logic.

*3.1.2. FIFO Resynchronizer.* Considering a producer and a consumer running at different speeds, that is, frequencies, the introduction of a FIFO queue between them allows to decouple the activities of the two actors. This effect is magnified by increasing the FIFO length. In a nutshell, the producer is allowed to insert data into the FIFO at its own speed until the queue is full, while the consumer can read with no restrictions provided that the queue is nonempty. As expected, FIFO-based resynchronizers are faster than *handshake*-based ones, but such speed comes at a cost in terms of area and power overheads due to the FIFO storage elements.

While different resynchronization schemas have been presented in literature [Chakraborty and Greenstreet 2003; Sarmenta et al. 1995; Beigne and Vivet 2006], in our simulation flow we implemented (to the best of our knowledge) the fastest and most flexible FIFO resynchronizer. Figure 3(b) details the basic blocks of the implemented bi-synchronous FIFO [Miro Panades and Greiner 2007] which can be used to resynchronize two clocks even with totally unrelated frequencies and phases (as the proposed handshake model presented in Section 3.1.2). The implemented FIFO resynchronizer provides two clocks, one to write and one to read, managed by the output port of the upstream router and the input port of the downstream one, respectively. From a timing viewpoint, the FIFO resynchronizer takes two cycles to deliver the message in the best case, since one cycle is required to write in the FIFO and the second to read the data from the queue. Note that the two cycles may have different frequencies (producer verus consumer). In the worstcase, it takes three cycles to deliver the message to the destination. Moreover, FIFO throughput is maximized when the FIFO queue has six slots, since three cycles are required in the worstcase to deliver the message while three additional cycles are required to return the credit back to the upstream router. In Miro Panades and Greiner [2007] the authors provided a detailed description of the implementation, including the power and area estimates we plugged into our flow.

### 3.2. DFS Module and PLL Model

The simulation framework supports two different DFS implementations: one that employs a single PLL for the whole chip and derives the clock for each frequency island through frequency dividers, and another using a dedicated PLL for each island.

In the first case, frequency modification is simulated as an abrupt change from the previous to new frequency. Frequency change requests not aligned to a clock edge boundary are properly delayed untill the next clock edge to avoid insertion of clock glitches. This is common practice in realistic clock switch implementations.

Conversely, in a clocking scheme employing a PLL for each frequency island, frequency changes are implemented by modifying the PLL set point. When simulating this implementation, the PLL step response is modeled using the two-pole transfer function of Eq. (1) whose parameters are configurable to approximate the response of a given PLL.

$$G(s) = \frac{1}{1 + 2\frac{\xi}{\omega}s + \frac{s^2}{\omega^2}} \tag{1}$$

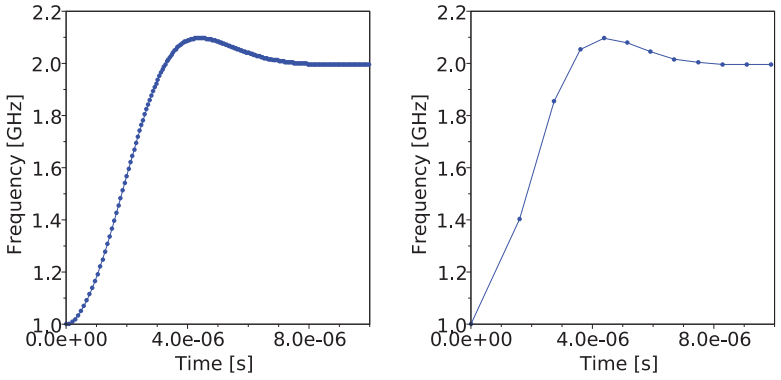



Fig. 4. Simulation of a frequency change from 1 to 2GHz with two PLL simulation granularities.

To capture the two-pole step response, a step change in the frequency set point is simulated by performing multiple individual frequency changes to the frequency island controlled by the PLL. This is performed by computing the step response of (1), which is (2). This equation gives the step response of a frequency change, given $f_o$ (the old frequency) and $f_n$ (the new desired one), and is sampled at each clock edge to compute the next clock period.

$$f(t) = f_o + (f_n - f_o)\left(1 + \frac{1}{\sqrt{1-\xi^2}} e^{-\xi\omega t} sin(\omega t\sqrt{1-\xi^2} + acos(\xi))\right). \quad (2)$$

This results in continuously changing the clock period on a cycle-by-cycle basis until the step response reaches its steady state, allowing an accurate simulation of the frequency change during this transition phase.

As this process entails a large number of individual frequency changes, it introduces an overhead in the simulation, although Section 4.3 shows it is negligible with respect to the time required to complete the whole simulation. However, the proposed implementation introduces a $k$ parameter to trade off simulation accuracy versus speed. Thus the step response is not evaluated, that is, not changed every clock period, but rather every $k$ clock periods, thereby reducing the number of frequency changes as well as the accuracy of the modeled behavior.

Figure 4 shows the simulation of the PLL model when changing its frequency set point from 1 to 2GHz, where individual frequency changes are marked with a dot. The left plot shows the results with $k = 1$. The frequency transition smoothly follows the two-pole step response, but to achieve this result 184 individual frequency changes are required. The right plot shows the results with $k = 16$. In this case, the frequency change is approximated with only 12 frequency changes.

### 3.3. Multistep PLL Model

While the PLL model presented in Section 3.2 mimics the reality, there are some cases where the simulated model cannot be accurate. In particular, the closed form Eq. (2) is reasonable if the step input is applied when the system is in a steady state, that is, at the end of any previous transient. However, considering a real multicore, it is possible that the PLL controller has a faster dynamic with respect to the PLL module, thus requiring a new frequency when the PLL has not reached the previous requested one. For example, Figure 5 shows how the PLL-simulated dynamic, that is, the blue line can greatly diverge from the real dynamic simulated one using MATLAB (green line) when the required steps are too fast with respect to the PLL dynamic. In particular,

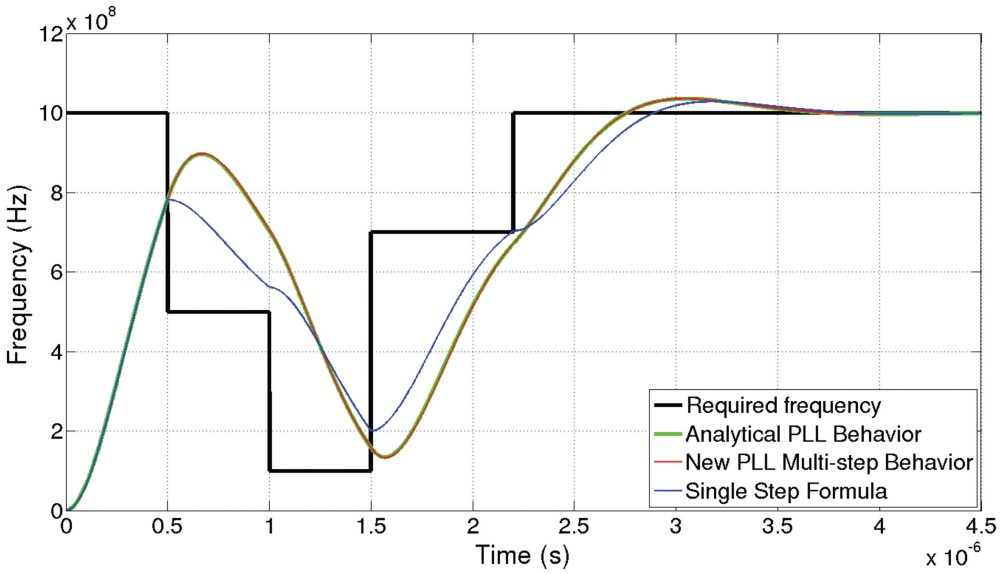



Fig. 5. The single-step PLL model against a too-fast frequency transition model. The implemented PLL provides a bad approximation with respect to the real model simulated using MATLAB when the new frequency is required before the previous transient has ended.

the PLL has a dynamic of 1.9us to reach the steady state while the required step frequency changes, that is, black line, can be faster than its dynamic, thus providing an inaccurate step response behavior.

Starting from this observation, a new implementation of the PLL has been integrated into our flow using the *Euler direct method* [Butcher 2003] to approximate a dynamic equation or a system dynamic of equations. Such method uses the first-order approximation term of the Taylor series of the function to compute the next approximate point, that is, $y(t+h) = y(t) + \frac{df(t)}{dt} * h$, where $h$ represents the integration step. In this perspective, the state-space representation of the PLL model is required despite the transfer function (see Eq. (1)) to exploit the Euler direct method. The final PLL implementation provides much more accurate results with respect to the baseline implementation, without any additional overhead. Moreover, the flow can still use $k = 16$ clock periods to produce accurate results, thus keeping low the number of frequency changes. Figure 5 shows how the multistep PLL proposed in this section, that is, the red line in Figure 5, can mimic the real dynamic simulated one using MATLAB (green line), even in presence of multiple frequency changes (black line), within a single transient period.

### 3.4. Spice-Based MOS Characterization

This module allows to decouple the accurate SPICE-based sleep transistor characterization from the runtime simulation. It provides a great simulation speedup, since it is executed once for each used MOS model. Starting from the SPICE-based *predictive technology models* (PTMs) [Zhao and Cao 2006], the module produces a set of approximate functions for the main parameters of the MOS to be used in optimization of the power gating network. Such functions are employed during power network design instead of invoking time-consuming SPICE runs. In particular, we consider only header sleep transistors to provide power gating support: a PMOS is inserted between the power rail and the circuit as detailed in Figure 6(a). Hence we model a $V_{dd}$ cut-off to switch off the logic block. There are two motivations to use header sleep transistor power

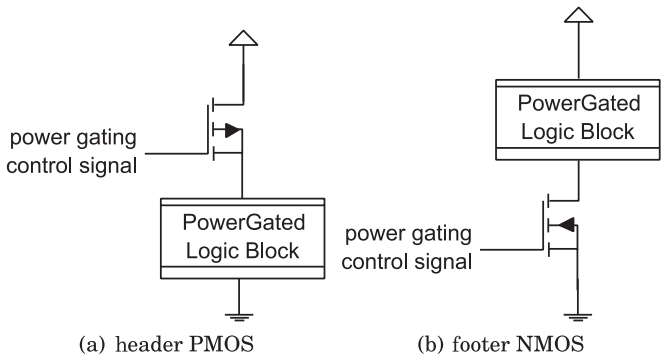



Fig. 6. Header and footer sleep transistor insertion for power gating support. Our flow exploits the header-based design.

gating. First, in sub-90nm designs, either a header or footer switch is used due to the constraint of sub-1V power supply voltage. Second, the PMOS-based power network is less leaky than the NMOS one [Shi and Howard 2006].

Referring to the available PTM SPICE models [Zhao and Cao 2006], four functions are automatically extracted to characterize the baseline sleep transistor: the ON resistance $R_{on}$, the switch resistance $R_{switch}$, the ON current $I_{on}$, and OFF leakage current $I_{off}$. This has been carried out for different MOS models and considering different technologies (45nm and 32nm), referring to the PMOS models of Shi and Howard [2006]. Data is extracted for a fixed technology node considering temperatures in the range of 300K–400K with steps of 5 degrees, also varying the channel widths: $1 \leq \frac{W}{L} \leq 10$, where L = technology node. From the raw data, a set of approximate functions, one for each required information, has been characterized as a function of both temperature and channel width and constitutes the module output.

### 3.5. Power Gating Network Design and Optimization

This module designs the power gating network for each selected component to be equipped with a sleep network. It takes three sets of inputs: the approximate MOS functions for the single sleep transistor from the SPICE-based MOS characterization module, the user constraints on wakeup time, minimum retention voltage, and maximum allowed performance overhead, and the worst-case temperature. The output is passed to the cycle-accurate power-performance simulator module by providing area, power, and performance overheads, due to the sleep network, for each selected component. Note that the methodology we present to design the power network resembles that described in Li et al. [2011], but is more accurate. In fact, to size up the sleep transistor W parameter and to meet the timing wakeup constraints, we use approximate functions based on real data instead of generic formulas.

The power network optimization procedure depicted in Algorithm 1 designs a distributed sleep transistor network [Long and He 2004] in three stages. The module analyzes the user requirements as well as worst-case power consumption for the considered logic block. Then, the user requirements are transformed into a set of new constraints that are expressed in terms of the required (conservative) power network design (see lines 1–3 in Algorithm 1).

In the second step, the best sleep transistor is selected starting from the provided worst operating temperature for the circuit and the $opt\_for$ value. The $opt\_for$ value allows to specify whether the baseline transistor must have the minimum $I_{off}$, maximum $I_{on}$, or maximum $\frac{I_{on}}{I_{off}}$, that is, the efficiency (see line 4). To this extent, the power

**ALGORITHM 1:** Power network optimization procedure.

**Input**: $u_{area,max}$, $u_{wakeup,max}$, $u_{perfdeg,max}$, $f_{PMOS}$, opt_for, $T_{worst}$

**Output**: $area_{overhead}$, $powerDyn_{overhead}$, $powerStatic_{overhead}$, $wakeup\_time$

1 $C_{block}$, $R_{block}$ = EvaluateBlockPower();

2 $R^{req}_{perf}$,$R^{req}_{on}$,$R^{req}_{switch}$ = ToPwrNetReq($u_*$,$C_{block}$, $R_{block}$);

3 $R^{req}_{min}$, $[swtich|on]$ = min($R^{req}_{perf}$,$R^{req}_{on}$,$R^{req}_{switch}$);

4 $R_st_on$ = getBestPMOS($[swtich|on]$, $f_{PMOS}$, opt_for,$T_{worst}$);

5 #PMOS = $R_st/R^{req}_{min}$;

6 $PN_{area}, PN_{wk}, PN_{I_{off}}, PN_{R_{on}}$ = PwrNetParam(#PMOS, $f_{PMOS}$, opt_for,$T_{worst}$);

7 $\tau = PN_{R_{on}} * C_{block}$;

network is evaluated as the number of sleep transistors required to meet the most conservative requirements.

Finally, power network statistics are collected and stored in the cycle-accurate simulator to be used at runtime. This procedure is executed at the beginning of simulation as a runtime initialization step. Hence, even if time consuming, it does not impact the runtime evaluation, which is the most time-consuming stage.

## 4. RESULTS

This section spans over five subparts whose goal is to show in practice the flexibility and some of the exploration capabilities offered by the proposed framework. Section 4.1 presents a simple DVFS policy to exploit the DFS model coupled with the supported dynamic voltage scaling. The latter sits on a delay model extracted from SPICEs level simulations of a commercial voltage regulator [Linear Technologies 2013]. A complete evaluation of the performance penalties associated with the two available resynchronization schemes is discussed in Section 4.2. In particular, the performance metric for the DVFS policy presented in Section 4.1 is evaluated for both resynchronization schemas. In addition, the impact of the resynchronizers is evaluated by imposing a fixed frequency to a GALS NoC equipped with handshake and FIFO resynchronization circuits between each router pair. In Section 4.3, we address the overhead in terms of time required for the simulation of GALS NoCs.

Section 4.4 outlines the identification process of those main MOS parameters useful to design the power gating network. A simple, while representative, power gating policy is finally discussed in Section 4.5 to exhibit the flexibility of the presented solution. Particular emphasis is given to the variety of available tuning parameters to customize both the policy and the actuators, as well as the fine-grain level of detail that is achievable. All the presented results are obtained using the microarchitectural configuration reported in Table II considering a 16-core architecture.

### 4.1. Model Exploitation through a Simple DVFS Policy

This section details a policy for power-performance exploration and optimization exploiting the accurate DVFS-developed model. We consider the 16-core architecture with a 45nm technology node and Vdd = 1V, as detailed in Table II and Section 4, whose DFS tunes the frequency in the range of 100MHz-1GHz that is the range of the identified PLL model. Moreover, for the DVFS actuator, we model a piecewise

Table II. Experimental Setup: Processor and Router Micro-Architecture and Technology Parameters

| | |
|---|---|
| Processor core | 1GHz, out-of-order Alpha core |
| Int-ALU | 4 integer ALU functional units |
| Int-Mult/Div | 4 integer multiply/divide functional units |
| FP-Mult/Div | 4 floating-point multiply/divide functional units |
| L1 cache | 64kB 2-way set assoc. split I/D, 2 cycles latency |
| L2 cache | 512KB per bank, 8-way associative |
| Memory Controllers | 4 at the topology corners |
| Coherence Prot. | MESI [Agarwal et al. 2009] |
| Router | 4-stage wormhole switched with 64b link width, 4vcs per vnet |
| | Frequency variable from 100 MHz to 1GHz |
| Topology | 4x4 2D-mesh, based on Tilera iMesh network [Wentzlaff et al. 2007], XY routing |
| PLL | 100MHz - 1GHz. Transient time 1.9us |
| Voltage Regulator | 0.7V - 1.0V. Transient time 5us |

continuous Vdd function defined ($f$ is the frequency in MHz) as follows:

$$f >= 800MHz\ Vdd = 1V,\ f >= 500MHz\ Vdd = 0.9V,$$
$$f >= 250MHz\ Vdd = 0.8V \text{ otherwise } Vdd = 0.7V,$$

The scheme accounts for the time overhead of the voltage regulator by suitably delaying frequency increases whenever a voltage increase is required. In particular, the timing overhead for frequency changes is accounted for in the simulator through the accurate PLL model described in Terraneo et al. [2013], while the voltage regulator has a $5\mu s$ actuation delay to allow the voltage to settle to the new value. This means that, when a frequency increase requires to increase the voltage, the frequency cannot be incremented before a $5\mu s$ delay. On the contrary, in the case of a frequency decrease the frequency is lowered immediately, following the PLL dynamics, while the voltage can be reduced later in time. As a consequence, a safe operational region on the voltage/frequency plane is ensured. Last, we implemented a 45nm SPICE PLL model using the PTM model [Zhao and Cao 2006] which provides 2mW worst-case power consumption. On the other hand, the worst-case voltage regulator power consumption is 2.5mW. Note that the PLL power consumption is aligned with state-of-the-art results [Elshazly et al. 2012], while accurate power models for both PLL and voltage regulator are left as future work.

Starting from the depicted actuators, we consider a policy for the routers where the frequency ($f_t$) varies according to a function depending on the congestion ($C_t$) of the router at time t, using the following law.

$$f_t = kC_t \tag{3}$$

Given a router, the congestion $C_t$ represents the number of flits stored in all the input ports of such router. The $k$ parameter is selected at design time to tune the sensitivity of the frequency actuator to the level of congestion. In particular, we selected k = 0.04 for the simulations since, with experimental validation, it was a good compromise between power saving and performance. The policy is executed at 10MHz and samples the $C_t$ for the considered router to steer the frequency. It is worth noticing, one more time, that we are not presenting a novel policy; the goal here is to highlight the capabilities of the proposed simulation flow.

Figure 7 compares different metrics, that is, frequency, dynamic power consumption, and congestion, on router 5 (a router in the center of the 4X4 2D-mesh NoC) running

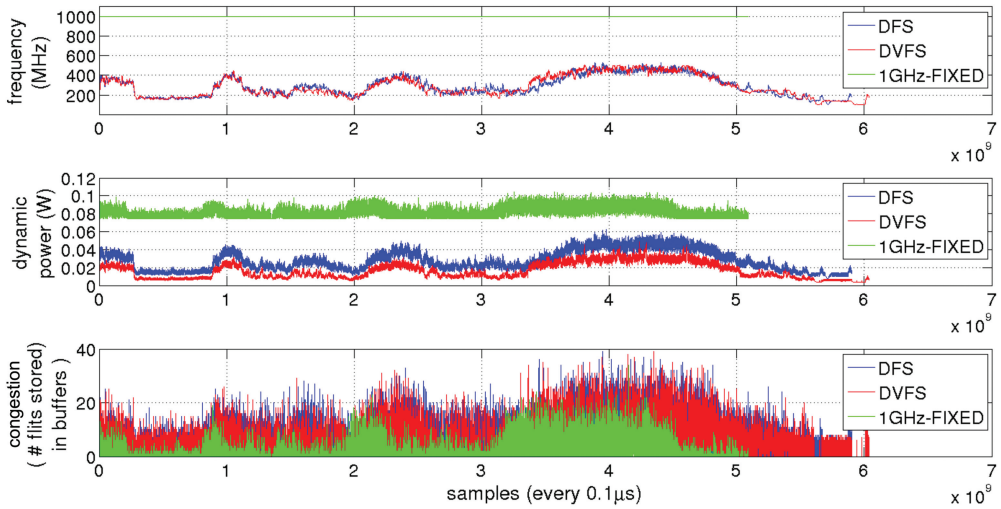



Fig. 7. Congestion and frequency for router 5 in a 16-core NoC multicore, where the NoCs run three policies: first, a 1GHz fixed frequency (green line), second a policy defined as $f_t = k * C_t$ with $k = 0.04$ and DFS with PLL blue line, and last, the same policy, that is, $f_t = k * C_t$ with $k = 0.04$, considering a DVFS actuator (red line).

the *qsort* benchmark in the cycle-accurate simulator with three different schemes for the NoC routers:

—static frequency scheme at 1GHz with GALS support (green line);
—policy in Eq. (3) with a DFS actuator, where the PLL settling time is 2us (blue line);
—policy in Eq. (3) with a DVFS actuator, where the PLL settling time is 2us and the voltage regulator settling time is 5us (red line).

All the considered scenarios have GALS support through the handshake resynchronizer, including the one where the frequency is fixed. Note that Section 4.2 discusses the impact of the two implemented resynchronization schemes against a non-GALS NoC, considering both static and dynamically adjusted frequencies through DVFS.

Figure 7 highlights three aspects. First, the plot reporting the frequency shows how even a simple policy can effectively manage the router frequency based on traffic in the router. In fact, a lower power consumption with respect to adopting a single fixed frequency is achieved by raising the frequency when the congestion increases and lowering it during the periods with limited congestion. Compared to the static frequency scheme, both DFS and DVFS have a lower power consumption while the execution time increases. Last, the DVFS provides an even lower power consumption due to the possibility of scaling the voltage down, paid in terms of higher execution time compared to the DFS solution. This is due to the timing overhead introduced by the voltage regulator which has its power overhead fixed at 2.5mW. Table III reports both timing and power for the three considered schemes, highlighting a 60% power reduction for the DFS scheme which increases to 74% if the DVFS scheme is used. Moreover, there is a 16% and 18% execution-time increase for DFS and DVFS, respectively, compared to the fixed frequency scheme.

### 4.2. Comparing FIFO and Handshake Resynchronizers

This section compares in two ways the performance of the two resynchronization schemes implemented in the simulation flow. First, the simple DVFS policy presented in Section 4.1 is exploited considering the two implemented resynchronization schema

Table III. Timing and Power Considering the *qsort* Benchmark Executed on Three Different NoC Schemes, (fixed frequency, DFS, and DVFS)

| Frequency scheme | Execution time (ms) | Power (mW) |
|---|---|---|
| Fixed | 5.1 | 81.5 |
| DFS | 5.9 | 26.3 (+2 PLL, +4 Resynch) |
| DVFS | 6.0 | 15.0 (+2 PLL, +4 Resynch, 2.5 VREG) |

We accounted for the 2mW PLL power overhead in the DFS and DVFS rows as well as 4mW due to 5 resynchronizers, that is, N, S, O, W, and PE ports of the router.

Table IV. Performance Analysis Considering FIFO and Handshake Resynchronization Schemes for Some Benchmarks from the MiBench Suite

| Resynch | susan | qsort | sha | search | dijkstra | bitcnts | basicmath |
|---|---|---|---|---|---|---|---|
| Handshake | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| FIFO-1slot | 1.01 | 0.98 | 1 | 1 | 0.96 | 1.00 | 0.99 |
| FIFO-2slots | 0.76 | 0.63 | 0.98 | 0.70 | 0.82 | 0.99 | 0.76 |
| FIFO-4slots | 0.71 | 0.56 | 0.97 | 0.63 | 0.75 | 0.99 | 0.73 |
| FIFO-6slots | 0.72 | 0.57 | 0.98 | 0.63 | 0.68 | 0.99 | 0.73 |

Different queue sizes are considered for the FIFO resynchronizer. For each combination of benchmark and resynchronization scheme the total execution time is reported, normalized with respect to the handshake resynchronizer.

with different MiBench [Guthaus et al. 2001] applications. Moreover, we explore the performance impact of the FIFO queue size. Second, the performance overhead due to resynchronization is evaluated comparing a baseline NoC without GALS support with two NoCs that support GALS, exploiting FIFO and handshake resynchronizer models. In all cases the frequency is kept fixed. In such a way it is possible to put consistently into evidence the impact of the resynchronizers. Moreover, we analyzed the scenario where a resynchronizer is put between each router pair.

Table IV reports the total execution time for different benchmarks using both FIFO and handshake resynchronization circuits, where the FIFO model is parametric in the number of queue slots. Moreover, the number are normalized to the handshake resynchronizer (the slower model): the lower-number the better performance. Moreover, all architectural parameters but the resynchronizer are the same for all the reported results, thus the impact of the resynchronization scheme is examined when coupled with a DVFS policy.

Results in Table IV highlight similar performance for the handshake model and the FIFO with a single queue slot (*FIFO-1slot*). This similarity is expected since the *FIFO-1slot* requires waiting for the credit after each flit transmission, that is, at least 2 plus 2 cycles.

In addition, increasing the number of FIFO slots greatly improves the performance for almost all benchmarks. In particular, a net improvement is shown by using 2 slots instead of 1, and performance saturation is observed between 4 and 6 slots depending on the relation between the two interfaced clock domains, as discussed in Miro Panades and Greiner [2007]. However, computationally intensive benchmarks with limited communication requirements are not greatly influenced by the exploited resynchronization scheme. For example, *bitcnts* shows low performance improvement using a bigger FIFO resynchronizer.

Other important considerations can be done by observing the data of Table V comparing three scenarios. The objective is to highlight the performance overhead between a GALS-based NoC and the baseline NoC without GALS support. In particular, we

Table V. Performance Analysis Comparing FIFO-6 and Handshake Resynchronization Schemes Keeping the Frequency Fixed against the Baseline NoC Without GALS Support

| Resynch | susan | qsort | sha | search | dijkstra | bitcnts | basicmath |
|---|---|---|---|---|---|---|---|
| Baseline NoC (No GALS) | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Handshake | 1.27 | 2.15 | 1.03 | 1.79 | 1.33 | 1.01 | 1.31 |
| FIFO-6 | 1.04 | 1.16 | 1.00 | 1.11 | 1.06 | 1.00 | 1.05 |

Timing results are normalized with respect the baseline NoC without GALS support to depict the performance overhead due to a specific resynchronization scheme.

Table VI. Timing Overhead (in microseconds) to Perform a Frequency Change Varying the Frequency Island Size and the Number of Used Resynchronizers

| Island size | 16 | 8 | 4 | 2 | 1 |
|---|---|---|---|---|---|
| Single change | 749 | 395 | 201 | 89 | 43 |
| Fast PLL | 5969 | 3666 | 1918 | 997 | 470 |
| Detailed PLL | 71529 | 42916 | 24626 | 12186 | 5249 |

focus again on a single router per frequency island, that is, the most flexible design, which maximizes the impact of the employed resynchronization scheme. Note that for all three architectures the NoC frequency is fixed at 1GHz.

Results in Table V show great performance degradation for the handshake resynchronizer: more than 2× slower with respect to the baseline non-GALS NoC. Moreover, the FIFO scheme with 6 slots shows a limited performance degradation, that is, less than 6% on average.

### 4.3. Simulation Overhead for DFS Modules

This section discusses the simulation time overhead that the proposed DVFS and GALS models introduce with respect to the baseline GEM5 implementation. In particular, we conducted several experiments considering different frequency island sizes with 1, 2, 4, 8, and 16 routers. The frequency of a single island has also been changed collecting the associated delays. Results are reported in Table VI.

It is interesting to note that the absolute time required to perform one single frequency change is below one millisecond, namely it is negligible compared to the typical time required to perform an entire simulation. The PLL models, as expected, require more time since this implies multiple individual frequency changes but, as shown in Section 3.2, it is possible to trade off accuracy versus speed. In addition, the time required to move events decreases with the size of frequency island. This was quite predictable, since the number of events is bound to the number of components in the frequency island. Moreover, the reported data points, out the quasi-linearity of the time required to move the events of a frequency island in response to a frequency change.

Contrary to the frequency change overhead which only stretches the simulated time (and simulation time as well, as shown previously) without altering the number of clock cycles to execute a given benchmark, the overhead of resynchronizers results in the introduction of additional clock cycles in the simulation due to the request and acknowledge scheme. The exact magnitude, however, depends on the frequency and phase of the two clock domains connected by the resynchronizer, which change throughout the simulation when DFS policies are active.

### 4.4. Approximate MOS Characterization for Power Gating

This section analyzes some results indicating the accuracy of approximate functions for the main MOS parameters, that is, $R_{switch}$, $R_{on}$, $I_{on}^{max}$, $I_{off}$. The use of simple functions to

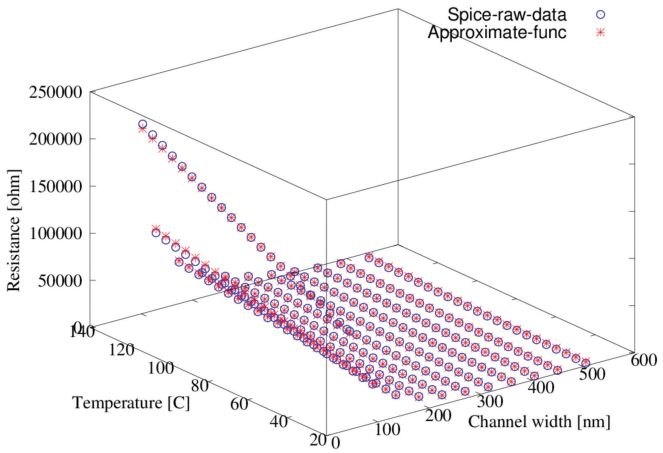



Fig. 8. Comparison of $R_{on}$ values considering a 45nm PMOS: SPICE simulation data vs. our approximate function. The maximum relative error is 2.5%.

gather the main parameters of a specific MOS model produces two main advantages: first, such abstract MOS representation provides enough information on a specific MOS model to replace a complete SPICE-based simulation; second, the mathematical formulation of the MOS model allows to employ standard optimization techniques, that is, linear and nonlinear programming, to design an optimal power network.

Figure 8 plots $R_{on}$ values for a specific PMOS transistor at 45nm a using a PTM model [Zhao and Cao 2006]. It depicts the $R_{on}$ considering both SPICE raw data and that computed using our approximate function. A very good accordance is evident: with a relative error always lower than 3%, the $R_{on}$ function can be safely (and effectively) used instead of SPICE simulations. Moreover, the low approximation error holds over a wide range of channel width and temperature combinations, in particular between 300K and 400K considering a channel width between $1 \leq \frac{W}{L} \leq 10$ (we considered $L = 45nm$ for simplicity instead of using the $L_{gate}$). Similar results have been obtained for all other significant PMOS parameters, considering also different technology nodes.

The baseline MOS sleep transistors have been designed at SPICE level and approximated with analytical functions to be directly used in the optimization stage inside the simulator. It is of paramount importance the availability of the exact equivalent capacitance and resistance of the whole power-gated block to optimally design the sleep network. In this perspective, we focused on the two main sources of static power in a NoC router, that is, crossbar and buffers. We developed accurate SPICE-level models for both components considering different flit widths and extracting their equivalent capacitance. On the other side, we identified the maximum current for each of these blocks (worstcase) and then the equivalent resistance has been computed employing Orion2.0.

### 4.5. Flexible Runtime Optimization Policy Exploiting Power Gating

This section explores a simple power gating policy targeting the crossbar switch, acting on the power state of such a block. It emerges how the implemented power gating module is flexible enough to easily cast different policies, thanks to seven tunable parameters. The ultimate goal of such a simple policy is basically a demonstration of flexibility of the proposed toolset. The following notations are adopted for this example: $th_L$ and $th_H$ are the two thresholds triggering the power gating; $time_{ON \rightarrow OFF}$ and $time_{OFF \rightarrow ON}$ are used to model the delay introduced by the power network implementation;

Table VII. Power Gating Policy Parameters for Different Simulations

| $time_{ON \rightarrow OFF}$ (ps) | $time_{OFF \rightarrow ON}$ (ns) | $f_{policy}$ (ns) | $f_{sample}$ MHz | $th_L$ (flits) | $th_H$ (flits) | Results |
|---|---|---|---|---|---|---|
| 1 | 1 | 200 | 100 | 2 | 10 | Figure 9 |
| 1 | 1 | 200 | 100 | 0 | 5 | Figure 10 |

$f_{policy}$ represents the frequency of the controller running the policy; and, finally, $f_{sample}$ is the sampling frequency. The last two parameters model actuation and data collection using three independent frequencies to mimic real designs. In fact, typically, the frequency of the chip is higher with respect the frequency of the optimization policy and of the data collection in order to make possible the processing of information by the policy itself. For example, the DVFS schema runs on the order of hundred ms, the CPU has a frequency around the GHz, while data collection using performance counters in GNU Linux takes some ms. The last element considered is the $timeout_{OFF \rightarrow ON}$ which represents the timeout before the power-gated block is switched on without considering any other parameter. This is mandatory to prevent starvation, since all the threshold-based policies suffer the impossibility to react unless the controlled variable crosses a threshold. For example, if two flits are in the buffers while the $time_{OFF \rightarrow ON}$ is set to three flits, the power-gated block will never be switched on.

Table VII reports different choices for parameters of the same threshold-based policy to show the flexibility of the proposed power gating module. The policy is run on a 16-core architecture where the main architectural parameters are reported in Table II. For each simulation, we display a figure reporting a timing diagram of both the power state and the level of congestion collected on router 5.

Table VII presents two policies where only the $th_L$, $th_H$ are different. There are three main aspects worth pointing out. First, we set the actuation logic slower than the power gating network, that is, 5ns and 1ns, respectively. Our SPICE-based evaluation indicates the possibility to switch on a 4-port 32-bit crossbar in 1ns, considering a performance overhead due to the sleep network lower than 10%. Furthermore, the work in Das et al. [2013] even reports an optimistic 5.1ns to wake up an entire router. The proposed policy is a valuable vehicle to demonstrate the flexibility of the proposed framework, although the specific focus here is towards the possibility to consider different overheads on both the actuator and decision policy. In particular, this example highlights the actuator as the bottleneck limiting the performance of the policy. It is the mimic of a real controller that can actuate at a lower frequency with respect to the circuit. For example, Figure 10 depicts this aspect slightly before sample 1200, where congestion increases while the crossbar switch remains switched OFF for some time before reacting.

The second observation is related to the sampling frequency that is lower than the actuation policy frequency. This models a scenario where the controller has to take decisions based on possibly old data. Both Figure 9 and Figure 10 expose this situation. For example, between samples 1600 and 1700 in Figure 9, the crossbar switch is active while the congestion remains at a level of four before reaching zero.

The last consideration pertains to the threshold levels. In particular, results in Figure 9 and Figure 10 differ because of different threshold levels. Simulation results show no improvement when the $th_L$ is too low, that is, 0, since a reasonably low value for this parameter is enough to schedule all the flits before the crossbar switches off. For example, Figure 9 reports data considering $th_L = 2$. In certain cases the crossbar switch is switched off by the policy when some flits are still in the input buffers, that is, between 1200 and 1300 the congestion never reaches zero, although the crossbar is switched off and on multiple times. On the other side, there are several cases where the

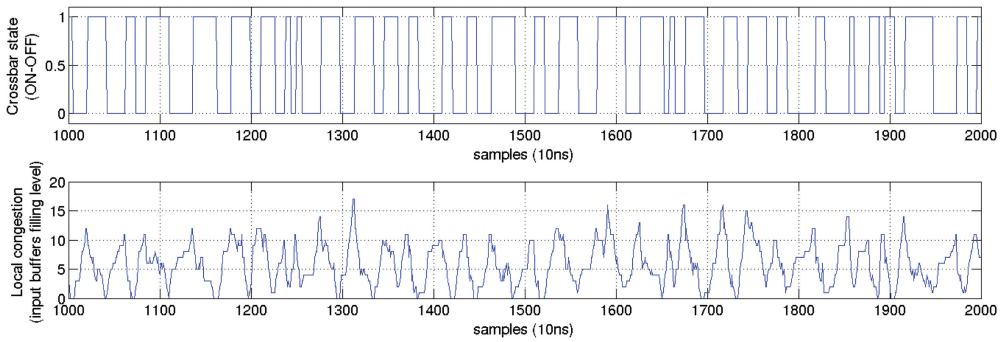



Fig. 9. Crossbar power state and local congestion running the described power gating policy. We set the $ON \rightarrow OFF$ time and $OFF \rightarrow ON$ to 1ps and 1ns, respectively, while 5ns for the control logic to answer the command. The high threshold is 10 flits while the lower is set to 2 flits.

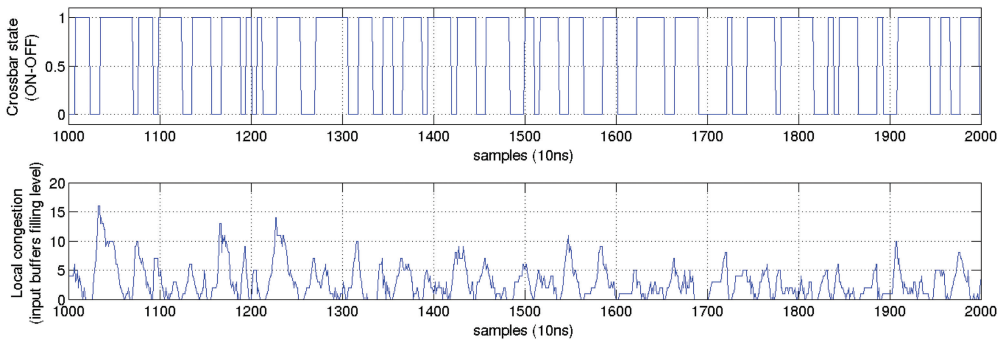



Fig. 10. Crossbar power state and local congestion running the described power gating policy. We set the $ON \rightarrow OFF$ time and $OFF \rightarrow ON$ to 1ps and 1ns, respectively, while 5ns for the control logic to answer the command. Moreover, the high threshold is 5 flits while the lower is set to 0 flits.

congestion reaches zero before the crossbar is switched off even if $th_L = 2$ (see all the times the congestion reaches zero in Figure 9). This occurs because different packets can have independent input and output port paths, thus the nonblocking property of the crossbar allows to faster reduce the congestion.

## 5. CONCLUSIONS

This article proposes a novel cycle-accurate simulation framework available in Zoni and Fornaciari [2015] to support exploration and optimization of the power and performance metrics during NoC design. Furthermore, it accounts for accurate power gating, DVFS, DFS, and GALS mechanisms, encompassing their power and performance overheads. Such overheads are integrated and added into the timing and power consumption figures of the architecturally simulated components. The value added is a measure of the impact of the actuators as well as of the real benefits for each methodology exploiting DVFS and power gating mechanisms, and possibly the GALS paradigm.

Results discussed in Section 4 highlight the relevant impacts on results produced by different hardware models. This justifies the need to use accurate actuator models in the simulation framework to avoid collecting unreliable and misleading results. For example, the use of a FIFO despite a handshake resynchronization circuit can degrade by more than 2× the multicore performance. To this extent, it is of paramount importance to use a simulation flow like the one proposed in this article to ensure

accurate results without overestimating the benefits of the proposed methodologies. Moreover, the accuracy of the modeled actuators enables further investigation strictly focused on reliability aspects, where the DVFS and power gating knobs can be seen as the solution to reliability issues, or the target of the reliability threat itself [Calimera et al. 2009a, 2009b].

In addition, the simulation flow allows to easily validate DVFS- and power-gating-based policies while still ensuring accurate results. Note that the introduced timing overhead to simulate a multicore equipped with our models is negligible. While different works addressed the use of power gating and DVFS considering also fully asynchronous design for the NoC interconnect, our proposal represents, to the best of our knowledge, the first comprehensive full-system simulation flow for early-stage microarchitectural explorations and optimizations, with power gating and DVFS support the NoC.

Finally, the extendability represents an additional key feature of the presented work, since both DVFS and power gating models can be adapted to work with the CPUs as well. Furthermore, our proposal implements the handshake and FIFO schemas that can be exploited to flexibly develop novel resynchronization circuits.

## REFERENCES


K. Agarwal, H. Deogun, D. Sylvester, and K. Nowka. 2006. Power gating with multiple sleep modes. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED'06)*. 633–637.

N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha. 2009. GARNET: A detailed on-chip network model inside a full-system simulator. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'09)*. 33–42.

A. Alhussien, C. Wang, and N. Bagherzadeh. 2010. A scalable delay insensitive asynchronous NoC with adaptive routing. In *Proceedings of the 17th IEEE International Conference on Telecommunications (ICT'10)*. 995–1002.

A. Banerjee, R. Mullins, and S. Moore. 2007. A power and energy exploration of network-on-chip architectures. In *Proceedings of the 1st International Conference on Network-on-Chip (NOCS'07)*. IEEE Computer Society, 163–172.

A. Bartolini, M. Cacciari, A. Tilli, L. Benini, and M. Gries. 2010. A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores. In *Proceedings of the 20th Great Lakes Symposium on VLSI (GLSVLSI'10)*. ACM Press, New York, 311–316.

E. Beigne, F. Clermidy, S. Miermont, and P. Vivet. 2008. Dynamic voltage and frequency scaling architecture for units integration within a GALS NoC. In *Proceedings of the 2nd ACM/IEEE International Symposium on Networks-on-Chip (NOCS'08)*. 129–138.

E. Beigne and P. Vivet. 2006. Design of on-chip and off-chip interfaces for a GALS NoC architecture. In *Proceedings of the 12th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC'06)*. 172–183.

N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. 2011. The gem5 simulator. *SIGARCH Comput. Archit. News* 39, 2, 1–7.

L. Bolzani, A. Calimera, A. Macii, E. Macii, and M. Poncino. 2009. Enabling concurrent clock and power gating in an industrial design flow. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'09)*. 334–339.

D. Brooks, V. Tiwari, and M. Martonosi. 2000. Wattch: A framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th Annual International Symposium on Computer Architecture (ISCA'00)*. ACM Press, New York, 83–94.

J. C. Butcher. 2003. *Numerical Methods for Ordinary Differential Equations*. Wiley.

A. Calimera, E. Macii, and M. Poncino. 2009a. NBTI-aware power gating for concurrent leakage and aging optimization. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED'09)*. ACM Press, New York, 127–132.

A. Calimera, E. Macii, and M. Poncino. 2009b. NBTI-aware sleep transistor design for reliable power-gating. In *Proceedings of the 19th ACM Great Lakes Symposium on VLSI (GLSVLSI '09)*. ACM Press, New York, 333–338.

T. E. Carlson, W. Heirman, and L. Eeckhout. 2011. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'11)*. 1–12.

A. Chakraborty and M. R. Greenstreet. 2003. Efficient self-timed interfaces for crossing clock domains. In *Proceedings of the 9$^{th}$ International Symposium on Asynchronous Circuits and Systems (ASYNC'03)*. 78–88.

X. Chen, Z. Xu, H. Kim, P. V. Gratz, J. Hu, M. Kishinevsky, U. Ogras, and A. Ayoub. 2013. Dynamic voltage and frequency scaling for shared resources in multicore processor designs. In *Proceedings of the 50$^{th}$ Annual Design Automation Conference (DAC'13)*.

M. H. Chowdhury, J. Gjanci, and P. Khaled. 2008. Innovative power gating for leakage reduction. In *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'08)*. 1568–1571.

S. Corbetta, D. Zoni, and W. Fornaciari. 2012. A temperature and reliability oriented simulation framework for multi-core architectures. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'12)*.

R. Das, S. Narayanasamy, S. K. Satpathy, and R. G. Dreslinski. 2013. Catnap: Energy proportional multiple network-on-chip. In *Proceedings of the 40$^{th}$ Annual International Symposium on Computer Architecture (ISCA'13)*. ACM Press, New York, 320–331.

T. Ducroux, G. Haugou, V. Risson, and P. Vivet. 2013. Fast and accurate power annotated simulation: Application to a many-core architecture. In *Proceedings of the 23$^{rd}$ International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS'13)*. 191–198.

A. Elshazly, R. Inti, M. Talegaonkar, and P. K. Hanumolu. 2012. A 1.5GHz 1.35mW 112dBc/Hz in-band noise digital phase-locked loop with 50fs/mV supply-noise sensitivity. In *Proceedings of the Symposium on VLSI Circuits (VLSIC'12)*. 188–189.

M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown. 2001. MiBench: A free, commercially representative embedded benchmark suite. In *Proceedings of the IEEE International Workshop on Workload Characterization (WWC'01)*. IEEE Computer Society, 3–14.

Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar. 2007. A 5-GHz mesh interconnect for a teraflops processor. *IEEE Micro* 27, 5, 51–61.

M. Hsieh, A. Rodrigues, R. Riesen, K. Thompson, and W. Song. 2011. A framework for architecture-level power, area, and thermal simulation and its application to network-on-chip design exploration. *SIGMETRICS Perform. Eval. Rev.* 38, 4, 63–68.

Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose. 2004. Microarchitectural techniques for power gating of execution units. In *Proceedings of the International Symposium on Low Power Electronics and Design (ISLPED'04)*. 32–37.

A. B. Kahng, B. Li, L.-S. Peh, and K. Samadi. 2009. ORION 2.0: A fast and accurate NoC power and area model for early-stage design space exploration. In *Proceedings of the Design, Automation, and Test in Europe Conference (DATE'09)*. 423–428.

A. B. Kahng, B. Lin, and S. Nath. 2012. Explicit modeling of control and data for improved NoC router estimation. In *Proceedings of the 49$^{th}$ ACM/EDAC/IEEE Design Automation Conference (DAC'12)*. 392–397.

H. Lebreton and P. Vivet. 2008. Power modeling in SystemC at transaction level, application to a DVFS architecture. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI (ISVLSI'08)*. 463–466.

S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. 2013. The McPAT framework for multicore and manycore architectures: Simultaneously modeling power, area, and timing. *ACM Trans. Archit. Code Optim.* 10, 1.

S. Li, K. Chen, J. H. Ahn, J. B. Brockman, and N. P. Jouppi. 2011. CACTI-P: Architecture-level modeling for SRAM-based structures with advanced leakage reduction techniques. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design (ICCAD'11)*. 694–701.

Linear Technology. 2013. Ltc3589 datasheet. http://cds.linear.com/docs/en/datasheet/3589ff.pdf.

M. Lis, P. Ren, M. H. Cho, K. S. Shim, C. W. Fletcher, O. Khan, and S. Devadas. 2011. Scalable, accurate multicore simulation in the 1000-core era. In *Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS'11)*. 175–185.

C. Long and L. He. 2004. Distributed sleep transistor network for power reduction. *IEEE Trans. VLSI* 12, 9, 937–946.

A. J. Martin and M. Nystrom. 2006. Asynchronous techniques for system-on-chip design. *Proc. IEEE* 94, 6, 1089–1120.

I. Miro Panades and A. Greiner. 2007. Bi-synchronous FIFO for synchronous circuit communication well suited for network-on-chip in GALS architectures. In *Proceedings of the 1st International Symposium on Networks-on-Chip (NOCS'07)*. 83–94.

A. Mishra, A. Yanamandra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C. Das. 2011. RAFT: A router architecture with frequency tuning for on-chip networks. *J. Parallel Distrib. Comput.* 71, 5, 625–640.

U. Y. Ogras, R. Marculescu, P. Choudhary, and D. Marculescu. 2007. Voltage-frequency island partitioning for GALS-based networks-on-chip. In *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC'07)*. 110–115.

S. Prabhu, B. Grot, P. Gratz, and J. Hu. 2009. Ocin_tsim-DVFS aware simulator for NoCs. http://homepages.inf.ed.ac.uk/bgrot/pubs/TSIM_ SAW09.pdf.

J. Renau, B. Fraguela, J. Tuck, W. Liu, M. Prvulovic, L. Ceze, S. Sarangi, P. Sack, K. Strauss, and P. Montesinos. 2005. SESC simulator. http://sesc.sourceforge.net.

L. F. G. Sarmenta, G. A. Pratt, and S. A. Ward. 1995. Rational clocking [digital systems design]. In *Proceedings of the IEEE International Conference on Computer Design: VLSI in Computers and Processors (ICCD'95)*. 271–278.

K. Shi and D. Howard. 2006. Sleep transistor design and implementation - Simple concepts yet challenges to be optimum. In *Proceedings of the International Symposium on VLSI Design, Automation, and Test (VDAT'06)*. 1–4.

V. Soteriou, N. Eisley, H. Wang, B. Li, and L.-S. Peh. 2006. Polaris: A system-level roadmap for on-chip interconnection networks. In *Proceedings of the International Conference on Computer Design (ICCD'06)*. 134–141.

J. Srinivasan. 2011. An overview of static power dissipation. http://wenku.baidu.com/view/6215a7145527072 2192ef711.html.

F. Terraneo, D. Zoni, and W. Fornaciari. 2013. A cycle accurate simulation framework for asynchronous NoC design. In *Proceedings of the International Symposium on System-on-Chip (SOC'13)*.

Y. Thonnart, P. Vivet, and F. Clermidy. 2010. A fully-asynchronous low-power framework for GALS NoC integration. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'10)*. 33–38.

D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. Brown, and A. Agarwal. 2007. On-chip interconnection architecture of the tile processor. *IEEE Micro* 27, 5, 15–31.

W. Zhao and Y. Cao. 2006. New generation of predictive technology model for sub-45nm design exploration. In *Proceedings of the 7th International Symposium on Quality Electronic Design (ISQED'06)*. 585–590.

D. Zoni, S. Corbetta, and W. Fornaciari. 2012. HANDS: Heterogeneous architectures and networks-on-chip design and simulation. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design (ISLPED'12)*. 261–266.

D. Zoni and W. Fornaciari. 2015. Sources of the simulation flow. http://hipeaclab.deib.polimi.it.

D. Zoni and W. Fornaciari. 2012. A sensor-less NBTI mitigation methodology for NoC architectures. In *Proceedings of the IEEE International SOC Conference (SOCC'12)*. 340–345.

D. Zoni and W. Fornaciari. 2013. Sensor-wise methodology to face NBTI stress of NoC buffers. In *Proceedings of the Design, Automation, and Test in Europe Conference and Exhibition (DATE'13)*. 1038–1043.

Received December 2013; revised February 2015; accepted March 2015