J Sign Process Syst
DOI 10.1007/s11265-015-0989-1

A DVFS Cycle Accurate Simulation Framework with Asynchronous NoC Design for Power-Performance Optimizations

Davide Zoni · Federico Terraneo · William Fornaciari

Received: 9 February 2014 / Revised: 7 February 2015 / Accepted: 25 February 2015

Abstract Network-on-Chip (NoC) is a flexible and scalable solution to interconnect multi-cores, with a strong influence on the performance of the whole chip. On-chip network affects also the overall power consumption, thus requiring accurate early-stage estimation and optimization methodologies. In this scenario, the Dynamic Voltage Frequency Scaling (DVFS) technique have been proposed both for CPUs and NoCs. The promise is to be a flexible and scalable way to jointly optimize power-performance, addressing both static and dynamic power sources. Being simulation a de-facto prime solution to explore novel multi-core architectures, a reliable full system analysis requires to integrate in the toolchain accurate timing and power models for the DVFS block and for the resynchronization logic between different Voltage and Frequency Islands (VFIs). In such a way, a more accurate validation of novel optimization methodologies which exploit such actuator is possible, since both architectural and actuator overheads are considered at the same time. This work proposes a complete cycle accurate framework for multi-core design supporting Global Asynchronous Local Synchronous (GALS) NoC design and DVFS actuators for the NoC. Furthermore, static and dynamic frequency assignment is possible with or without the use of the voltage regulator. The proposed framework sits on accurate analytical timing model and SPICE-based power measures, providing accurate estimates of both timing and power overheads of the power control mechanisms.

Keywords Network-on-Chip · Simulation · DVFS · GALS · Simulation · Multi-cores

D. Zoni (✉) · F. Terraneo · W. Fornaciari
Politecnico di Milano - DEIB, Milano 20133, Italy
e-mail: davide.zoni@polimi.it

F. Terraneo
e-mail: federico.terraneo@polimi.it

W. Fornaciari
e-mail: william.fornaciari@polimi.it

1 Introduction

In this decade multi-core processors emerged as the solution to deliver increasing processing power to demanding applications, ranging from the embedded to the supercomputing market. This scenario highlights the Networks-on-Chip (NoCs) as a viable communication infrastructure to face scalability, flexibility and reliability issues. However suitable power-performance optimization methodologies are required, since the NoC power consumption is not negligible and its behavior directly impacts the whole multi-core performance.

In the perspective to tradeoff power-performance, Dynamic Voltage and Frequency Scaling (DVFS) has been widely used in processor designs to change both voltage and frequency at run-time, depending on the actual processor's load. Multi-core evolution drives the research in decomposing the chip in different Voltage and Frequency Island (VFI), to aggressively exploit the DVFS mechanism. However, synchronization between different VFIs must be carefully evaluated to prevent negative side-effects on the system performance. Today, these techniques are primarily referred as Globally Asynchronous Locally Synchronous (GALS) [1, 2], to capture the idea of a set of synchronous components communicating to each others by means of resynchronization circuits, since the clocks of the components can be totally uncorrelated.

Traditionally, DVFS schemes coupled with GALS design paradigm were mainly used in processors, while several recent proposals focus on DVFS and GALS design to optimize the NoC [3].

The evaluation of such complex multi-cores composed by processors, cache hierarchy and interconnect is usually carried out exploiting cycle accurate simulation frameworks. Though they allow a relatively fast exploration of the design space with respect to hardware prototyping, the obtained results greatly depend on the accuracy of the modeled components. In particular, the impact of accurate models for the DVFS actuators and resynchronization circuits widely influences the simulation results, and this is the rationale of this paper demonstrating how it is crucial using such models. However, to the best of our knowledge, the state of the art lacks of simulation frameworks which enable accurate DVFS and GALS implementation integrated (in a seamless way) within the architectural simulator. Conversely, the works in literature usually focus on specific methodologies where the DVFS and resynchronizers are intended as a functional part of the system, without accounting for their power and performance overheads.

This paper presents a multi-core simulation framework that sits on publicly available research tools. It enhances the NoC model with the support for accurate DVFS and GALS timing and power models to provide both system reachability and controllability. Thus, both power and performance metrics can be monitored at run-time and reaction strategies exploiting the DVFS actuator can be designed and assessed within the framework.

1.1 Network-on-Chip Background

This section briefly overviews the considered NoC architecture. In particular, NoC is an interconnection subsystem composed of links and routers delivering CPU and memory requests and responses, from source to destination. This work considers a wormhole NoC architecture with Virtual Channels (VCs) [13] where each request or response from the CPUs and memory blocks is decomposed in packets to be transmitted through the NoC. Moreover, the simulation flow exploits a 4-stage router pipeline [13]. A packet is considered split in multiple atomic transmission units called flits. The first flit of each packet is the header flit. A body flit represents an intermediate flit of the original packet while the tail flit is unique for each packet and represents the final flit of the packet itself. When a flit enters in the baseline router from one of the input ports, if it is an header flit it has to pass through four pipeline stages plus traversing a network link. First, it is stored in the virtual channel (VC) buffer that has been reserved by the upstream router through a buffer write stage (BW). Route Computation (RC) stage is performed in the same clock cycle only if the flit is an header one which determines the output port for this new packet. The virtual channel allocation (VA) represents the next pipeline stage that reserves to the new packet an available virtual channel in the downstream router from the selected output port. If the VC allocation is successful, the flit competes for a crossbar switch path to its output port during the switch allocation (SA) stage. Finally, if the flit wins the switch allocation, the following steps are switch traversal (ST) and link traversal (LT), that account for the delay to traverse the upstream router crossbar and upstream-downstream router links, respectively. Tail and body flits require to traverse fewer pipeline stages since they exploit some resources and information already reserved to the packet by the header flit (i.e., VC and RC). The NoC supports bidirectional communication between each pair of routers via two links for each communication directions: a network link, from source to destination, allowing packet transmission and a destination to source control link, to send back control flow information.

1.2 Novel Contribution

Starting from publicly available research tools the paper presents a multi-core simulation framework which supports accurate DVFS and GALS mechanisms for the NoC. The framework is available in [14]. The first steps of this work have been published in [15]. The novel contributions of this paper, described in Section 3.2, Section 3.4 and Section 4.2, are briefly summarized thereafter:

- *Complete, Flexible and extensible framework* - the proposed framework supports the exploration of different multi-core design metrics, providing accurate models for the actuators. Different simulation frameworks appeared in literature and Table 1 highlights their limitations as well as the new features added in this work.
- *DVFS models and GALS support* - the DVFS model sits on an accurate SPICE level PLL (Phase Locked Loop). Moreover, a delay model for voltage regulator is built starting from SPICE simulations of a commercial component [16]. The final DVFS accurately models timing aspects as well as worst power consumption. The GALS support implements two resynchronization schemes, handshake [17] and FIFO [18], allowing trading area and power consumption over performance penalties. Both the handshake and the FIFO resynchronizers allows to partition the design into multiple VFIs, even if the handshake one severely impacts the system performance, as discussed in Section 3.5 and Section 4.4. However, it is valuable to support multiple VFIs where the design imposes restrictive constraints on both area and power overheads. Even if a single VFI

Table 1 State-of-the-art on multi-core simulators: features, advantages and drawbacks with focus on GALS and DFS support for NoC.

Framework	Cycle-accurate CPU+NoC	NoC support	Power support	GALS support	DVFS/DFS projection	PLL/divider exploration	Objectives
Renau et al. (SESC) [4]	✓						multi-core simulation, parallel applications
Soteriou et al. (Polaris) [5]		✓	✓				Network-on-Chip design-space exploration
Hsieh et al. (SST) [6]		✓	✓				microarchitecture, power and thermal
Lis et al. (HORNET) [7]		✓	✓	✓			many-core processors, mainly NoC interconnect
Bartolini et al. [8]		✓	✓		✓		run-time control policies for multi-cores
Zoni et al. (HANDS [9, 10])	✓	✓	✓				power/perf, reliability/perf for multi-cores
T.E. Carlson et al. (Sniper [11])	✓	✓	✓		✓		distributed parallel simulator for multi-cores
Subodh Prabhu (Ocin tsim [12])		✓			✓		test different DVFS schemes for NoC
Our Proposal	✓	✓	✓	✓	✓	✓	CPU+NoC, PLL/divider, DFS, GALS design

running at a lower frequency without any handshake resynchronizer can provide the same overall system performance, it does not allow to tune the power metric at fine granularity.

- *Run-time policy assessment* - a lightweight policy module interface allows to write novel management policies, which exploit DVFS in the NoC routers at different granularities, i.e. VFI of different sizes. By accessing to run-time power and timing estimates for different router logic blocks, the policy can actuate on frequency and voltage of the controlled IP blocks at run-time.

1.3 Paper Structure

The rest of the paper is organized in four sections. Section 2 details the state-of-the-art on simulation frameworks supporting DVFS and asynchronous NoC. Section 3 presents the framework focusing on different design aspects, i.e. PLL design, resynchronization scheme as well as the policy evaluation module. Results are then reported and discussed in Section 4 highlighting the flexibility of the proposed simulation flow. Finally, some conclusions are drawn in Section 5.

2 Related Works

Considering current and future multi-cores, power consumption issues represent an hot research topic, that is frequently faced in the form of the optimal power-performance trade-off design. Even if several works address power-performance balancing, most of them rely on low level actuators, i.e. power gating and dynamic frequency scaling (DFS), that are already available on the architecture employed for tests. However, the introduction of such hardware actuators requires accurate pre-silicon analysis, thus there's a need for accurate simulation frameworks allowing to explore the design space for such actuators. In this perspective this section is divided in two parts. First, it is presented an overview of the state-of-the-art related to simulation frameworks supporting DFS and GALS network-on-chip. Then, a few proposals exploiting DFS and GALS design will be described to underline the crucial importance of such technologies and the need for accurate simulation frameworks.

Several proposals can be found in the literature to aid designers during early stages of platform definition. Only few of them are specifically focused to support power-performance trade-off analysis in multi-core scenarios considering GALS NoC or DFS support for NoC routers. *Wattch* [19] constitutes the first cycle-accurate single-core power-performance simulator. However, the advent of multi-core architectures required simulation tool-chains that allow to accurately mimic the behavior of multi-core systems also considering asynchronous components and DFS support. In this perspective, the *SESC* simulator [4] provides cycle-accurate simulation of bus-based multi-core processors based on the MIPS architecture. However, it does not support Network-on-Chip architectures and does not support for DFS nor asynchronous NoC design. The *Polaris* framework [5] 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 an heterogeneous NoC model to allow for dynamic frequency changes during simulation. [7] is meant to simulate large-scale architectures, and exploits parallel simulation on physical hardware with particular emphasis on the on-chip network. While the framework enables the possibility for power-performance trade-off analysis, it lacks of a complete asynchronous on-chip network model, thus it is not possible to explore different GALS configuration for the interconnect as well as the simulation considering dynamic frequency scaling based on high level policies. The work in [11] presents *Sniper*, a framework which can simulate multi-cores underpinned by an on-chip network interconnect supporting per core DVFS. However such support is not present for the NoC.

Table 1 reports a summary of the main characteristics of the aforementioned analysis frameworks compared to our proposal.

Coming to a brief literature review regarding asynchronous NoCs and DFS, [1] proposes a design methodology for partitioning a NoC architecture into multiple voltage and frequency islands (VFIs) and to assign supply and threshold voltage to each VFI. The employed resynchronization scheme is based on FIFO buffers.

The work in [20] presented a complete DVFS scheme for IP unit integration to be employed for NoC-based design. However, this work does not easily allow to model different policies or different topologies as in a cycle accurate simulation framework, since it has mainly a prototyping focus.

In contrast, our proposal is intended for early stage design space exploration within an accurate simulator capable to simulate also a Linux based OS.

3 Proposed Estimation Flow

The capability of the proposed framework to simulate different multi-cores is expressed by Fig. 1. The left part reports a tiled 2D-mesh multi-core where the NoC is partitioned in multiple (five) VFIs, each of them with a different shape to put in evidence the flexibility of our proposal. The interconnection between each VFI pair is regulated by a resynchronizer module that is detailed in Section 3.5, i.e. the blue box in Fig. 1. Moreover, each router has one L2 bank connected at least one core, even if multiple cores per router are possible. It is worth noticing that both L2 and the cores are not considered part of the VFI on any router, thus a resynchronizer is used to manage their connection to the NoC. Last, the green box on the top right side of Fig. 1 highlights the policy, that is the baseline component used to manage voltage and frequency depending on the actual power and performance levels.

From the implementation viewpoint, the proposed simulation flow sits on a set of publicly available tools that are used as basic building blocks. GEM5 [21] is the exploited event-driven simulator for multi-core architectures with NoC-based communication [22]. Orion2.0 is used as the power model for the NoC, while additional components has been added to support the accurate GALS and DVFS models. Figure 2 reports the information flow between the various simulation framework components to implement DVFS, split in two parts: run-time (top box) and design-time (bottom box).

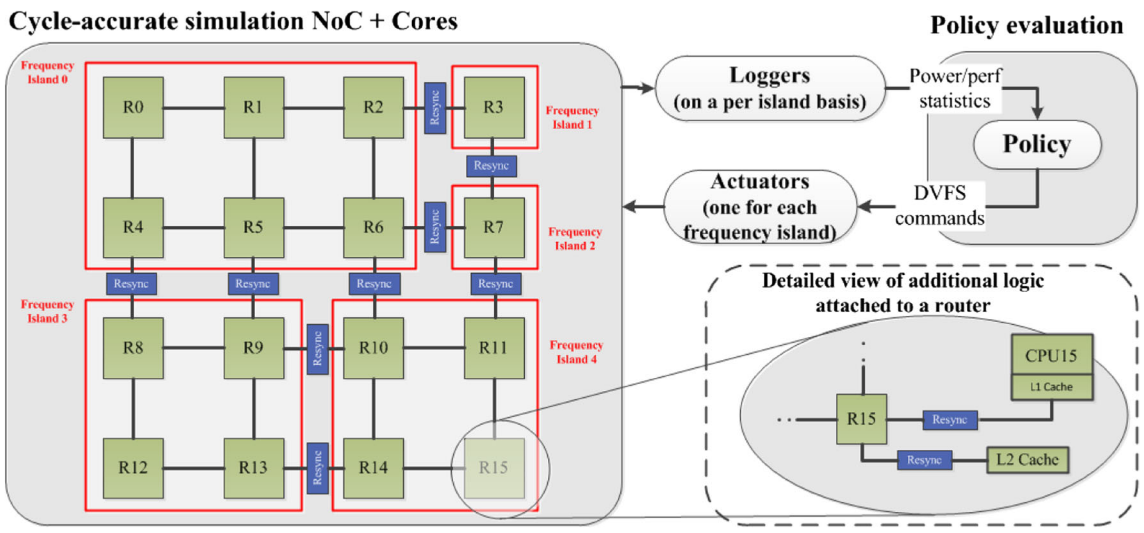


Figure 1 Logical view of the proposed simulation toolchain.

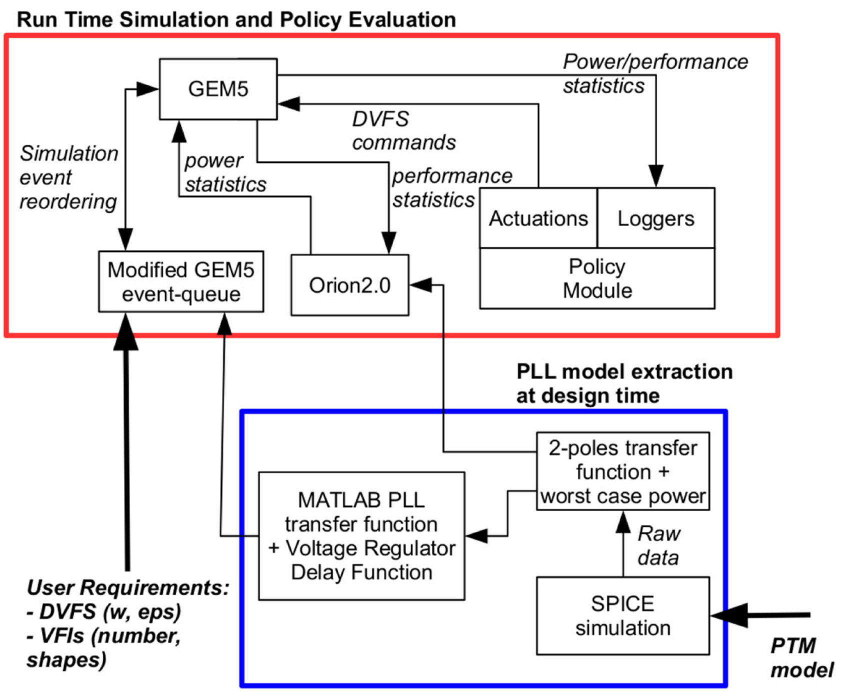

Run Time Simulation and Policy Evaluation
GEM5
Power/performance statistics
DVFS commands
Simulation event reordering
power statistics
performance statistics
Actuations
Loggers
Policy Module
Modified GEM5 event-queue
Orion2.0
PLL model extraction at design time
2-poles transfer function + worst case power
MATLAB PLL transfer function + Voltage Regulator Delay Function
Raw data
SPICE simulation
User Requirements:
- DVFS (w, eps)
- VFIs (number, shapes)
PTM model

Figure 2 Information flow between all the tools to provide the DVFS support to the NoC model.

Starting from the Predictive Technology Models (PTM) [23] (bottom-right corner in Fig. 2), that are SPICE level transistor models, a 45nm PLL SPICE has been developed and evaluated to extract worst case power figures and timing data. The power information are added to Orion2.0, while the timing data have been used to develop a MATLAB-based PLL model, which has been implemented in GEM5. Last, the simulation is driven by both architectural events as well as the events generated by the policy module that directly interact with GEM5.

Note that the event management system of the simulator has been extended to support the possibility to move already scheduled events between different simulation times. This aspect is mandatory to the DFS behavior. In particular, we needed a framework capable to group all the events scheduled for a single component, and to 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 time they will need to be processed considering the frequency change, and storing the new frequency value, so that subsequently generated events will be scheduled at the appropriate time. It is worth noticing that both the VFI partitions as well as the resynchronizers are not directly observable in the information flow, since they are integrated in GEM5 without exchanging information with the rest of the simulation framework.

The rest of this section is organized in four parts detailing the most important introduced components. In particular, Section 3.1 reports the baseline PLL model and its implementation, while Section 3.2 deals with advanced PLL modelling issues. Furthermore, Section 3.3 addresses the SPICE-level PLL implementation aspects. A delay model for the voltage regulator is discussed in Section 3.4, which is mandatory to take into account the need to rise up the voltage before increasing the frequency. Finally, Section 3.5 presents the two resynchronizer models implemented in the proposed simulation flow, i.e. FIFO and handshake.

3.1 Baseline DFS Module

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, a frequency change is simulated as an abrupt change from the previous to the new frequency. Frequency change requests not aligned to a clock edge boundary are properly delayed till the next clock edge to avoid the insertion of clock glitches, as real world clock switch implementations do.

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

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

A step change in the frequency set point is then simulated by performing multiple individual frequency changes to the frequency island controlled by the PLL to track the two-pole step response. This is performed by computing the step response of Eq. 1, which is Eq. 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.

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

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. To allow the user to trade off simulation accuracy for speed, a configuration option k has been introduced, to sample the step response (and therefore cause a frequency change) not every clock period, but every k clock periods, thereby reducing the number of frequency changes.

Figure 3 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.

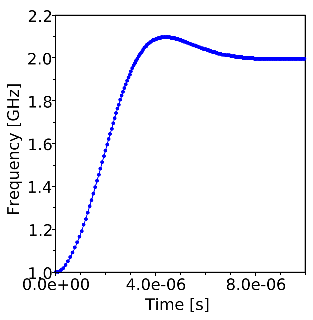
3.2 Multi-step PLL Model

While the PLL model presented in Section 3.1 provides a good approximation of real PLL dynamics, there are some cases for which the simulated model is not accurate enough. In particular, the closed form (2) is reasonable if the step input is applied when the system is in a steady state, i.e. every previous transient response has completed. However, considering a real multi-core, it is possible that the PLL controller operates faster with respect to the PLL module, thus providing a new frequency when the PLL has not reached the previous requested one. Figure 4 shows how the simulated PLL can differ from the real PLL dynamics simulated using MATLAB, when the required frequency changes are too fast with respect to the PLL dynamics.

Starting from this issues a new implementation of the PLL has been integrated in the proposed framework using the *Euler Direct Method* [24] to approximate a dynamic equation or a dynamic system of equations. It is worth notice that the *Euler Direct Method* uses the first order approximation term of the Taylor series of the function to compute the next approximate point, i.e. $y(t+h) = y(t) + \frac{df(t)}{dt} * h$, where h represents the integration step. In this perspective, the *state space* representation of the PLL model is required in spite of the transfer function (see Eq. 1) to exploit the *Euler Direct Method*. The final PLL implementation, which simulation results are shown in Fig. 5 provides much more accurate results with respect to the baseline implementation, without any additional overhead, since the frequency approximation is always required. Moreover, the implementation can still use $k = 16$ clock periods, thus keeping low the number of frequency changes.

3.3 SPICE PLL Model

The PLL model proposed in Section 3.2 is a flexible tool to evaluate the timing overhead and the feasibility of each methodology which exploits the PLL as actuator. However, its power consumption represents another critical metric to be considered. This section presents a SPICE-based, 45nm, 1V PLL implementation exploiting the PTM models [23]. The PLL implementation is based on the work in [25],



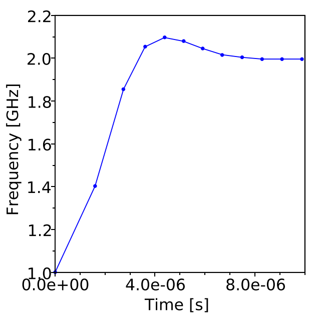


Figure 3 Simulation of a frequency change from 1 to 2GHz with two PLL simulation granularities.

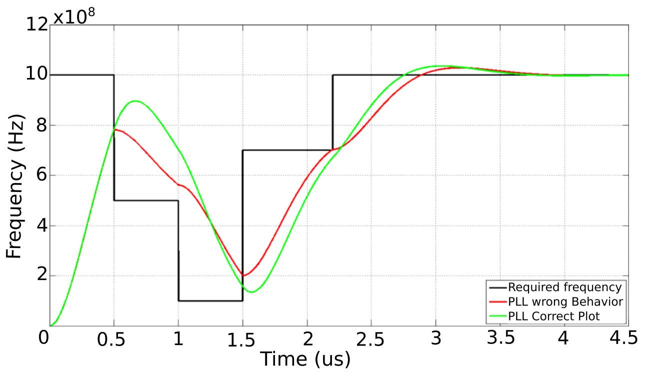


Figure 4 The single step PLL model against a fast frequency transition. The implemented PLL provides bad approximation results with respect to the real model simulated using MATLAB when the frequency set point changes before the previous transient response has exhausted.

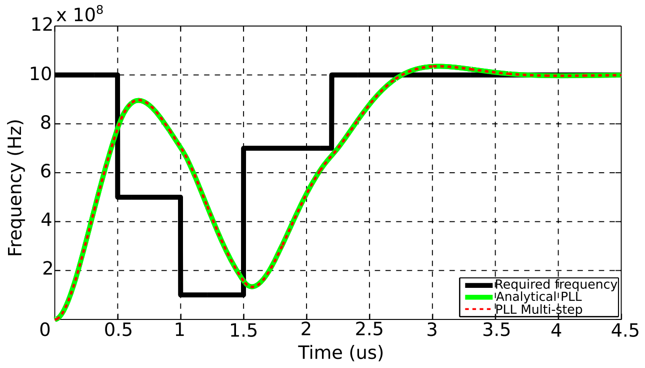


Figure 5 The multi step PLL model against a fast frequency transition. The behavior remains close to the MATLAB simulated system even when the frequency set point changes before the previous transient response has exhausted. Thus, the PLL response plots are overlapped.

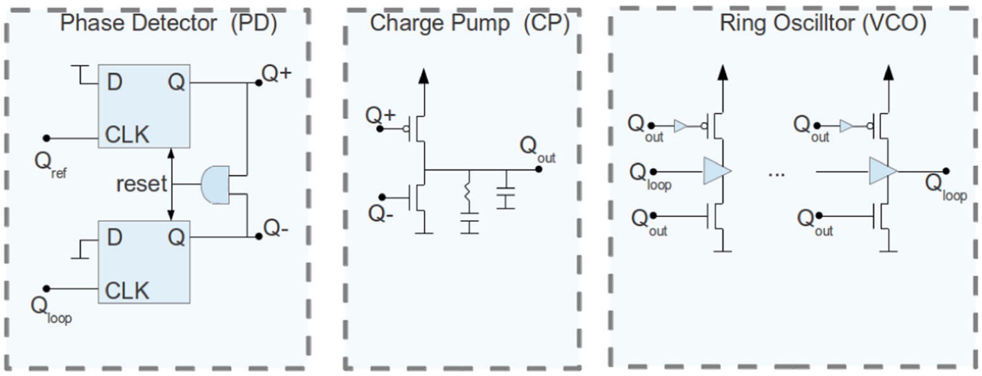
which however provides results for a 350nm PLL implementation thus not considering actual sub-micron technology aspects. While a complete analytical power model for the PLL has not been developed yet, the SPICE implementation provides a coarse grained way to evaluate the PLL power consumption which emerges to be less than 2mW for a PLL which works in a 100MHz-1.5GHz frequency range. Figure 6 details the generic PLL logic scheme composed of three blocks. The *phase detector* (PD) compares the reference signal (Q_{ref}) with the PLL output signal (Q_{loop}). The Q_{loop} is also the output signal produced by the PLL used as clock signal for the blocks of logic. The PD produces a two signal output, which are used to increase or decrease the voltage applied to the VCO (Voltage Controlled Oscillator) block. Note that the frequency depends on the applied voltage, thus to increase or decrease the PLL output frequency a proper increase or decrease of the applied voltage is required. The *charge pump* block is in charge to increase or decrease the voltage level applied as modifier to the *ring-oscillator-based* VCO block.

3.4 Dynamic Voltage Scaling

For what concerns the DVS scheme, the simulation framework implements it as a slave of the DFS component, where the voltage is set to the lowest possible value according to the desired frequency, considering the necessary safety margin. In particular four assumptions are hold. First, all the voltage-frequency pairs are assumed to be in the linear voltage-frequency region, where higher frequencies requires higher voltages. Moreover, we considered a finite number of possible voltage values. This means that, at each step, the lower possible voltage value is selected to support the actual required frequency. Third, the frequency increase takes place when the required voltage is stable. For example, when a lower frequency is required such a change happens immediately, since the current voltage value can support the new required frequency. On the other hand, the request of an higher frequency could result in a delayed actuation. For example, if the new required frequency requires an higher voltage value, the voltage regulator is triggered to increase the voltage and only when the new voltage is stable the frequency change starts. It is worth to note that in case of a frequency decrease the voltage value can be decreased too, but this action happens in parallel with the frequency reduction. However for each frequency modification the voltage value is checked to guarantee that no increase happens using an incorrect voltage value. Although the simulation infrastructure is flexible enough to consider different voltage regulator timing overheads, in Section 4 we consider a specific voltage regulator instance with real timing overheads.

From the power viewpoint, we studied a commercial Linear Technology Power Management solution, namely the LTC3589 [16], since the reported specifications matches our requirements. The LTC3589 provides multiple switching voltage regulators and linear regulator addressing the power management of a complete System-on-Chip (SoC) solution. In particular, the switching regulators can provide up to 1.6A with a power overhead up to 2.5mA. Results in

Figure 6 The logic scheme for a PLL model from [25].



Section 4 report a single router power consumption always lower than 150mA at 1V, thus the considered power management solution allows to safely group up to 8 routers in a single frequency island. As already said, the considered module represents an example of use of real data for a specific instance of voltage regulator. However, the structure of our simulator is flexible enough to plug different power consumption values for the voltage regulator depending on the considered real component.

3.5 Resynchronization Scheme

Once different parts of the same digital system are working at different clocks or at the same frequency but at different phases, an interface is required to ensure reliable data transfer as well as avoid metastability. This work focuses on asynchronous NoCs only, thus a resynchronization scheme between each router pair or a router and its connected computational components, i.e. cores, memory controllers and cache controllers, is required. Note that extending the DVFS capabilities to CPUs is straightforward, since the DVFS actuator model is the same and the resynchronizers are already in the between of each CPU-router pair. It is worth noticing that the proposed resynchronization scheme interface is conceived to be a customization point of the simulator, and it is extensible to easily plug different resynchronization schemes to verify their performance. The rest of this section details the schemes that are currently implemented.

Handshake Resynchronizer Starting from the work in [17], the implemented 2-way handshaking protocol adds two single bit lines only, i.e. request and acknowledge, to a network link between two routers. The resynchronization logic scheme is depicted in Fig. 7 with focus on two clock domains, e.g. two routers. In particular the left part of the figure reports the output port interface on the upstream router, while the right side provides the input port interface of the downstream router. When a new flit is ready to be sent out, the upstream router triggers the *new_flit* signal for 1 clock cycle. This forces to toggle the *req* signal 1 cycle later. Moreover, the back path in the upstream router switches the *busy* signal high. After the propagation delay the *req* signal enters to a two flip-flop chain in the downstream router, that is used to avoid metastability issues. The third flip-flip in the chain is used in couple with the *req_stable* signal as an edge detector, since our resynchronizer works on edges to increase throughput [17]. The edge detector triggers 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 signal the data transfer completion. Also the upstream router manages the *ack* signal using a two flip-flip chain to avoid metastability issues. The *busy* signal is used to prevent the transmission of new flits until the reception of the acknowledge signal.

FIFO Resynchronizer Starting from the work in [18], the proposed framework implements the FIFO model to resynchronize two routers as well as its connected computational and memory components. The FIFO resynchronizer allows to decouple the transmitter and receiver, since the former can send data up to fully fill the FIFO at its own frequency while the receiver can read data up to the frequency of the sender. In particular, the FIFO has two clocks, i.e. write and read, while the first is the same clock signal of the sender and the second is the clock of the receiver. In addition two signals, i.e. full/empty, move from the FIFO to the sender and the receiver, respectively. The deeper the FIFO the stronger the decoupling capabilities. All in all, the presented FIFO introduces only two delay cycles in the best scenarios, i.e. one to send from the upstream router to the FIFO and one to read from the FIFO, while [18] reported a 6 cycle penalty in the worst case, i.e. some corner cases when the clocks of

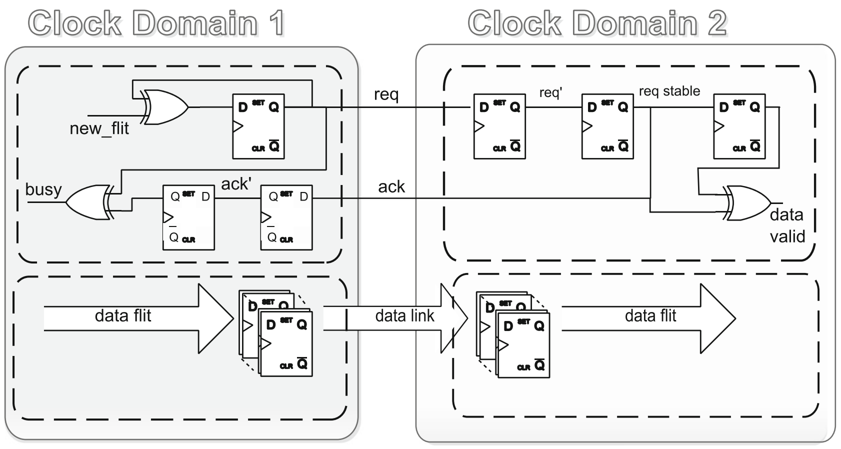


Figure 7 The implemented edge sensitive resynchronization scheme.

Table 2 Architectural parameters for the considered voltage regulator and PLL. Note that the voltage decrease is assumed shadowed by the PLL timing as detailed in Section 3.4.

$0 \leq f \leq 250MHz$	0.7V
$250 \leq f \leq 500MHz$	0.8V
$500 \leq f \leq 750MHz$	0.9V
$750 \leq f \leq 1GHz$	1V
V- $\rightarrow$ V+	5us (voltage increase delay)
PLL delay	2us (ω=4 $\cdot$ 10^6, ξ=0.6) (See Eq. 1)

the two blocks are totally uncorrelated. Thus, the FIFO resynchronization scheme provides better performance than the handshake one, while a non negligible power and area overhead is introduced mainly due to the FIFO queues. It is worth noticing that our flow implements the timing model of the FIFO resynchronizer, while it relies on [18] for synthesis results, i.e. area and power.

4 Results

This section addresses the flexibility and scalability of the proposed framework supporting exploration and optimization of power and performance metrics for a NoC considering DVFS, DFS and GALS mechanisms. The section is organized in five parts. Section 4.1 presents a simple yet effective test to assess the correctness of the implementation. Section 4.2 considers the design space exploration aspects related to power and performance metrics accounting both DFS and DVFS also analyzing different VFI sizes. Section 4.3 presents a threshold-based DFS policy, thus highlighting the possibility to dynamically change frequencies by exploiting a simple frequency divider to enhance the flexibility of the framework. Section 4.4 discusses the performance overhead due to different resynchronization schemes. Last, Section 4.5 reports timing overheads introduced by the simulator to support DVFS and GALS for the NoC(Table 2).

All the presented results are obtained using a 16 core architecture using a 4x4 2D-mesh NoC. The details of the microarchitectural configuration are reported in Table 3. Moreover, Table 2 details the voltage regulator and the PLL parameters used for the simulated scenarios of this section.

4.1 Implementation Correctness

The proposed framework implements an asynchronous NoC model inside a cycle accurate simulator, also allowing to implement DVFS. The implementation impacts the kernel of the simulator, thus a proof of the correctness of this enhancement should be provided, since the final semantics cannot be guaranteed *a priori*.

To this extent we provide a proof of the validity of our solution addressing the correctness of the expected behavior of the simulation in two ways. First of all a black-box test was performed by simulating the full architecture. The code chosen to run on the cores is a subset of 9 tests from the MiBench [27] suite. The tests were performed with a simple policy that changes the frequency of the NoC every 100ns, with the sole aim to stress the added DFS and resynchronization functionality. No discrepancies were found between the expected and obtained output. This shows that the modifications to the GEM5 simulator did not introduce errors that affect code execution. Second, we checked the timing accuracy of the introduced components. To this purpose we considered multiple simulations starting from the same multi-core and using the same benchmark set, but changing the frequency of the NoC for each simulation. These tests were performed using

Table 3 Experimental setup: processor and router micro-architectures and technology parameters.

Processor core	2GHz, 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 placed in the corners of the 2D-mesh
Coherence Prot.	MESI token (for real traffic) [22]
Router	4-stage wormhole switched with 64b link width, 4vcs per vnet Frequency variable from 500 MHz to 2GHz
Routing	Dimension Order Routing, XY.
Topology	4×4 2D-mesh, based on Tilera iMesh network [26]
Technology	45nm at 1.0V

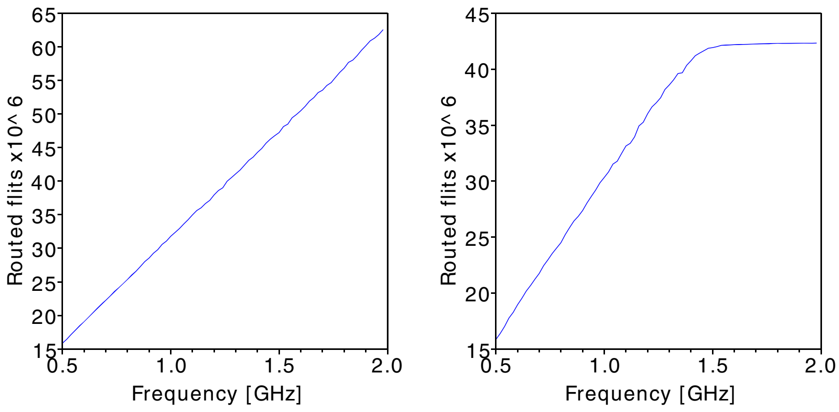


Figure 8 Number of routed flits as a function of **router frequency** with two network load scenarios, to demonstrate the ability of the proposed framework to operate with different router frequencies.

a synthetic traffic generator to be able to control the rate of packets in the NoC. Figure 8 reports the number of routed flits as a function of the router frequency, in two different network load scenarios. The router frequencies range from 500 MHz to 2GHz with a step of 20 MHz, i.e. we have $\frac{2GHz-500MHz}{20MHz}$ different simulations with fixed frequency. The left part of Fig. 8 reports the simulations using 0.50 flit/port/cycle while the right part of the same figure reports the same simulations using an injection rate of 0.07 flit/port/cycle. The first case shows how, with a sufficiently high network load, the increase in frequency produces a linear increase in the processed flits. This means that the resynchronizers, as expected, do not affect the linearity of the frequency/processed flits relation. Moreover, the second case shows saturation at 1.5GHz. This is not a microarchitectural saturation, but rather shows that the NoC does not benefit from high frequencies when the injection rate is low. The third test addresses the timing correctness of the DFS implementation. We used the same 16-core architecture and traffic load as in the second test, but the change in frequency between simulations was emulated using a PWM-like (Pulse Width Modulation) scheme alternating between only the two boundary frequencies: 500MHz and 2GHz. For each simulation we operate a fixed number of frequency changes between the two frequencies considering all the routers as a single frequency island. Each simulation is different from the others in terms of duty-cycle, i.e. the percentage of the simulation time spent in each one of the two frequencies. During each of these simulations there were two frequency changes (high-to-low and low-to-high) per 800ns, for a total of 40000 frequency changes per simulation. In particular, Fig. 9 reports the received packets as a function of the duty-cycle. The range is from 2.5%, where most of the time of the NoC is spent at 500MHz, up to 97.5%. The left and right graphs of Fig. 9 are two set of simulations that differ in the flit injection ratio exactly as in the previous test.

4.2 Design Space Exploration

The joint power-performance exploration represents a critical step in the evaluation of the NoC. This is mainly due to the need to evaluate multiple interacting subsystems and the impact of different actuators. Thus, this section explores a simple policy for NoC power-performance optimization that exploits both DFS and DVFS actuators using the FIFO resynchronizer model with 6 buffer slots. Moreover, two scenarios are analyzed, i.e. one router and 4 routers (organized in a 2x2 topology) per VFI. The purpose of the policy is to steer the description of the framework capabilities. Each VFI is equipped with a DFS/DVFS module and both

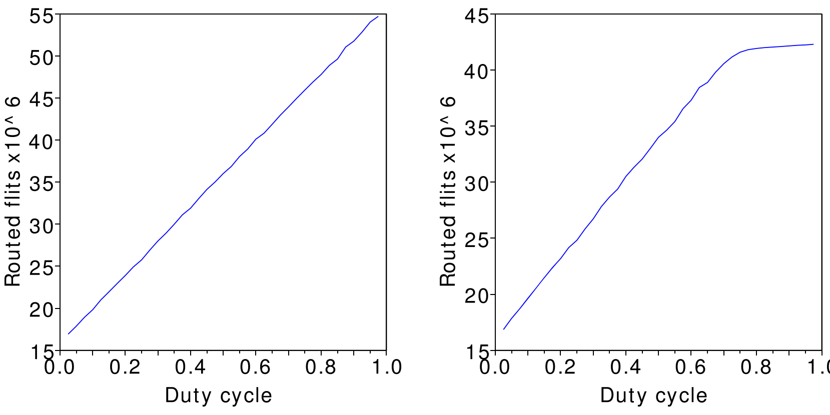


Figure 9 Number of routed flits as a function of **duty cycle** with two network load scenarios, to demonstrate the ability of the proposed framework to operate frequency changes correctly.

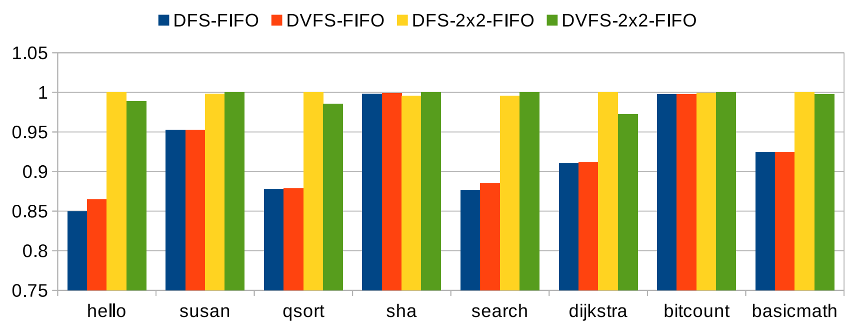


Figure 10 Execution time considering 8 MiBench applications on a 16 cores architecture with a 4x4 2D-mesh NoC. Results are normalized per benchmark.

the voltage regulator and the PLL parameters are defined in Table 2: four voltage values are considered, 5us is the voltage regulator transient time and the considered PLL for both DVFS and DFS actuators has a 2us transient time. Moreover, each VFI updates its frequency and eventually the voltage according to the following policy:

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

where f_t is the VFI frequency and C_t represents the number of flits that are stored in the input buffer of the router at time t. K is a constant fixed at 0.04 which represents the actuation strength depending on the actual router load. While different values are possible, the selected one represents a conservative choice obtained through experimental exploration. A sample rate of C_t at 10MHz is considered, i.e. low actuation frequency.

Figure 10 and Fig. 11 report the execution time and the power consumption of the NoC considering eight benchmarks belonging to the MiBench [27] suite. Each benchmark is simulated in four different scenarios combining DFS or DVFS with VFIs of different sizes, as reported in Table 4.

Figure 10 highlights a comparable time for the execution with DFS and DVFS. This is reasonable due to the small transient time of the voltage regulator. Considering the 2x2 VFI scenarios, the DVFS solution can slightly overcome the DFS due to the side effect of averaging the buffer utilization of each router in the VFI. In particular, this effect is not present in the single router VFI configurations where the DFS is faster than DVFS to react, even if negligible improvement are reported. On the other hand, increasing the VFI size imposes a frequency update based on the average buffer occupancy of the buffers in all the router inside the considered VFI. This degrades the reaction of the policy, as showed in the results where for each benchmark the use of single router VFIs always overcome the architectures with VFIs of 4 routers. Such performance reduction is independent of the used actuator, i.e. DFS and DVFS, and it is 10.45% in average.

Both *sha* and *bitcount* show comparable performance regardless the VFI size and the use of DVFS or DFS. However, from the analysis of the results an almost flat interconnect utilization is observed. Hence the policy has no reasons to frequently change the frequency, so that both power and performance overheads are limited.

Power consumption is also greatly affected by the exploited mechanism, i.e. DFS or DVFS, and the organization of the VFIs. The DVFS greatly impacts power reduction, since both frequency and voltage are controlled. Considering single router VFI scenarios, DVFS overcomes the power reduction obtained by DFS of up to 22.45%. Besides the same behaviour between DVFS and DFS is observed even when 2x2 VFIs are used.

Surprisingly, the power consumption of the DVFS and DFS, where single router VFIs are used for the first and 2x2 VFIs are used for the latter, are similar. In particular, we assumed 2.5mW worst case power consumption for the voltage regulator as extracted from SPICE simulations of a commercial device [16]. However, the FIFO buffer distribution in the NoC also impacts the total power. To this

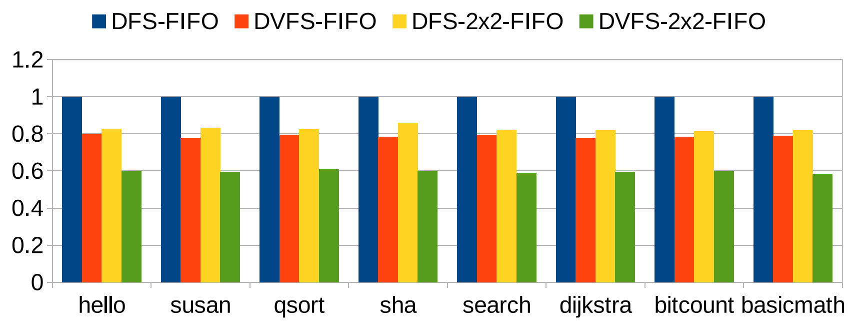


Figure 11 Total power, i.e. static, dynamic and clock power, considering 8 MiBench applications on a 16 cores architecture with a 4x4 2D-mesh NoC. Results are normalized per benchmark.

Table 4 Four explored configurations combining DFS or DVFS, with the size of the VFI. VFI contains 1 or 4 router. The latter organizes routers in 2x2 topology.

	DFS	DVFS
single router VFI	DFS-FIFO	DVFS-FIFO
2×2 VFI	DFS-2×2-FIFO	DVFS-2×2-FIFO

extent, we approximatively evaluated the power required by a single FIFO buffer as the static power consumption of the buffer of the same size, i.e. 6 slots, plus the dynamic power obtained from its write and read statistics. Moreover, we used both the voltage and frequency of the downstream router to collect data considering the FIFO attached to the downstream VFI [28].

The use of DVFS provides many advantages in term of power reduction while negligible performance overheads are introduced. On the other hand the use of bigger VFIs contributes to lower the power yet again with a performance degradation around 10%.

4.3 Run-time Optimization Policies

One of the most prominent uses of the presented framework is to evaluate the quality of different DFS/DVFS policies operating on NoC routers. In this perspective, this section discusses a per router DFS-based and threshold-based policy, used as an example to highlight the flexibility of the proposed work. We present a simple policy that can switch between three frequencies, i.e. *HIGH*=800MHz, *NORMAL*=500MHz and *LOW*=250MHz depending on the router load. The switch is managed using threshold values on the congestion metric (C_t) that is the per router used performance metric, i.e. the number of flits stored in the input ports of the considered router. In particular, the policy distinguishes between an high congestion threshold and a low congestion threshold, *HIGH-TH* and *LOW-TH* respectively. Last, we impose *FREQ-CHANGE-LIMIT* as the minimum time between two frequency changes. At the beginning of the simulation the frequency is set at 500MHz. Then, the congestion values are sampled every 0.1us. However, the policy can change the router frequency on a multiple of the sampling period, i.e. $\frac{FREQ\text{-}CHANGE\text{-}LIMIT}{0.1us}$. Figure 12 reports a timing diagram showing frequencies, dynamic power and congestion levels for $R5$ on a 16-cores running the FFT MiBench. The *HIGH-TH* and *LOW-TH* are set at 10 and 20 flits respectively, as highlighted by the red lines in the congestion graph. The *FREQ-CHANGE-LIMIT* is set at 10 times the sampling period, i.e. 1us. This means that the policy has to maintain the frequency constant at least for 1us regardless of the actual congestion level.

Data reported in Fig. 12 highlight two aspects. First, the threshold policy allows to reduce the router frequency at low router load, thus reducing dynamic power consumption. This is clear from Fig. 12 where the congestion level is almost always below the *HIGH-TH*, thus the *HIGH* frequency is used only when required. Moreover, the threshold policy allows to reduce even further the power consumption lowering the frequency at *LOW* in case of very low traffic.

Second, the policy is not optimal, since in some cases the HIGH frequency requirement due to high congestion is not

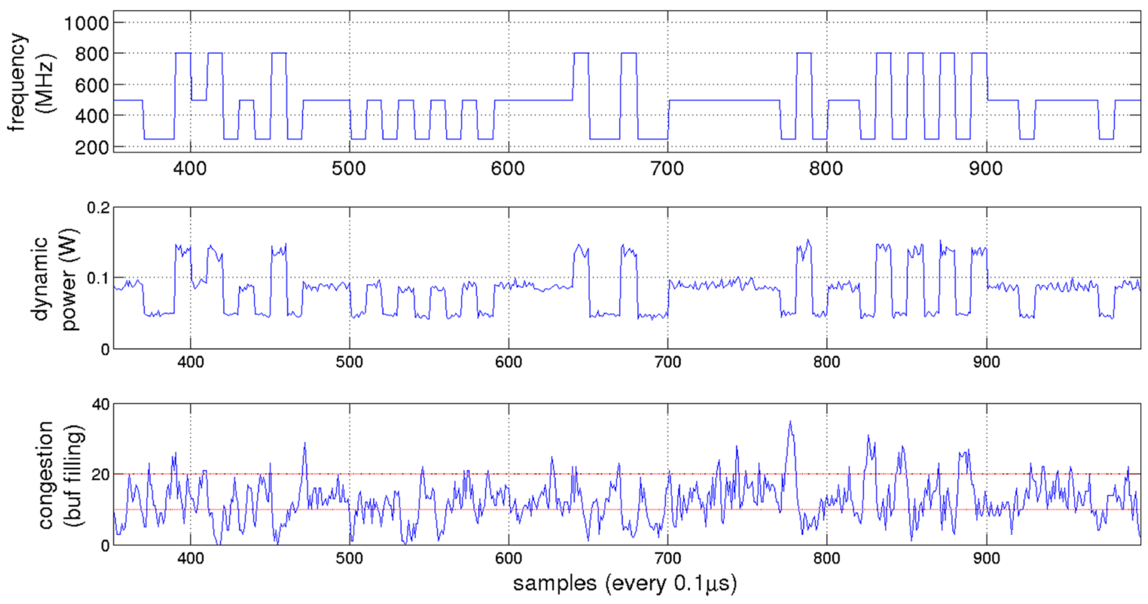


Figure 12 Frequency, power and congestion traces sampled while running the described DFS policy.

satisfied, thus the router performance decreases. For example, around sample 780 the congestion has a peak reaching a value close to 40 flits, while the policy cannot increase the frequency that has been changed to *LOW*. In this situation the congestion peak decreases due to the application behaviour and remain under the *LOW-TH* value until sample 800. However, the policy samples the congestion value during the slope as an *HIGH-TH* thus the frequency increases to HIGH without motivation, since congestion is decreasing due to application behaviour. To this extent the policy does not take the right choice in all cases, sometimes wasting power without a performance improvement. However, even if the proposed policy is not optimal, it allows to show how the presented framework can help to assess runtime power-performance optimization procedures focusing on the limits of the procedures themselves, while the framework provides information on power, performance and timing at the same time.

4.4 Resynchronization Schemes

This section explores the impact on performance of different resynchronization schemes with respect to a non-GALS NoC. In particular, three different scenarios are analyzed, i.e. *Base NoC*, *Handshake* and a six slot deep FIFO resynchronizer called *FIFO-6*. Each configuration has been tested using multiple benchmarks from the MiBench suite. Every scenario has the frequency of the NoC fixed at 1GHz, while both *Handshake* and *FIFO-6* assume each router as a VFI. Handshake and FIFO resynchronizers are thus present. Furthermore, the frequency for all the simulations is fixed to allow a consistent comparison between the three architectures. Table 5 reports the performance results, as total execution time normalized to the *Base NoC* time. Lower values means faster simulations while higher ones highlight a performance penalty with respect to the *Base NoC*. Results point out the better performance achieved by the *FIFO-6* resynchronizer with respect to the *Handshake* one in all the simulation benchmarks. Moreover, *FIFO-6* can perform up to 13.38 times better than the *Handshake*, as reported for the *sha* application. All in all, the *FIFO-6* introduces a limited performance overhead with respect to the *Base NoC* without GALS support, providing an average 6% performance penalty. In summary, the *FIFO-6* can be exploited to support the DVFS mechanisms if a low performance penalty is required. The *Handshake* resynchronizer is instead preferable when minimizing area and reducing leakage are the main design drivers, as it does not require buffers and only consists of a few logic gates and flip-flops.

Table 5 Performance analysis comparing *FIFO-6* and *Handshake* resynchronization schemes keeping the frequency fixed against the baseline NoC without GALS support. Timing results are normalized with respect the *Base NoC* without GALS support to isolate the performance overhead due to a specific resynchronization scheme.

Resynch	Susan	Qsort	Sha	Search	Dijkstra	Bitcnts	Basicmath
Base NoC (No GALS)	1	1	1	1	1	1	1
Handshake	1.27	2.15	13.38	1.79	1.33	1.01	1.31
FIFO-6	1.04	1.16	1.00	1.11	1.06	1.00	1.05

4.5 Simulation Overhead

This part details the simulation time overhead that the presented models introduce with respect to the baseline GEM5 implementation. A 6-core Xeon-v2 with core frequency up to 2.93GHz and 16 GB of RAM was used for all the simulations in the rest of this section. Several scenarios are examined considering different frequency island sizes with 1, 2, 4, 8, 16 routers. Then, we changed the frequency of a single island and collected the time required to change the island frequency. Results are reported in Table 6. It is interesting to note that the absolute time required to perform one single frequency change is below one millisecond, thus it is negligible compared to the usual time required to perform an entire simulation. Considering the 4x4 2D-mesh architecture whose architectural parameters are provided in Table 3, the $hello_world$ benchmark takes around 30 seconds to be completed without using the DVFS, while small timing overheads for the DVFS are reported in Table 5 for a single frequency change.

The PLL model, as expected, requires more time as it implies multiple individual frequency changes, but as shown in Section 3.1, it is possible to trade off accuracy for speed. The time required to move events decreases with the size of the frequency island. This was expected, since the number of events is bound to the number of components in the frequency island. Moreover, the reported data highlights the quasi-linearity of the time required to move the events of a frequency island in response of a frequency change.

Contrary to the frequency change overhead, which only stretches the simulated time (and simulation time as well,

Table 6 Timing overhead (in microseconds) for performing a frequency change depending on frequency island sizes and change model.

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

as shown above) without altering the number of clock cycles to execute a given benchmark, the overhead of the resynchronizers results in the introduction of additional clock cycles in the simulation. The exact number, however, depends on the frequency and phase of the two clock domains among which the resynchronizer is connected, which changes throughout the simulation when DFS policies are active.

5 Conclusions

This paper presented a novel simulation framework available in [14] to support the exploration and optimization of the power and performance metrics in the NoC. Furthermore, it accounts for accurate DVFS, DFS and GALS mechanisms encompassing their power and performance overheads. Such overheads are integrated and added to the timing and power consumption of the architectural simulated components, thus providing a measure of the actuator impact as well as a real benefit for each methodology that exploits DVFS mechanisms and GALS paradigm.

Results discussed in Section 4 highlight the great impact different hardware models can have on the overall simulation results. For example, the use of a FIFO in spite of an handshake resynchronization circuit can degrade the multicore performance up to 13 times. To this extent, it is of paramount importance to use a simulation flow like the one proposed in this work to prevent a possible overestimation of the benefit of the proposed methodologies.

The simulation flow allows to easily validate DVFS based policies ensuring accurate results. In this scenario, our proposal represents the first, to the best of our knowledge, comprehensive full system simulation flow that allows to validate novel microarchitectural solutions also exploiting actuators, focusing on the NoC.

Last, the extendibility represents an additional key feature of the presented work, since DVFS models can be adapted to the CPUs, while resynchronization schemes can be optimized starting from the provided models for each of the most representative families, i.e. FIFO and handshake.

References

1. Ogras, U., Marculescu, R., Choudhary, P., & Marculescu, D. (2007). Voltage-frequency island partitioning for gals-based networks-on-chip. In *Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE* (pp. 110–115).
2. Chapiro, D.M. (1984). Globally-asynchronous locally-synchronous systems, Ph.D. dissertation, Stanford University, Report No. STAN-CS-84-1026.
3. Mishra, A.K., Yanamandra, A., Das, R., Eachempati, S., Iyer, R., Vijaykrishnan, N., & Das, C.R. (2011). Raft: A router architecture with frequency tuning for on-chip networks. *J. Parallel Distrib. Comput.*, *71*(5), 625–640. doi:10.1016/j.jpdc.2010.09.005. [Online]. Available:.
4. Renau, J., Fraguela, B., Tuck, J., Liu, W., Prvulovic, M., Ceze, L., Sarangi, S., Sack, P., Strauss, K., & Montesinos, P. (2005). SESC simulator. In: http://sesc.sourceforge.net.
5. Soteriou, V., Eisley, N., Wang, H., Li, B., & Peh, L.-S. (2006). Polaris: A system-level roadmap for on-chip interconnection networks. In: *ICCD 2006.*, pp.134 –141.
6. Hsieh, M.-y., Rodrigues, A., Riesen, R., Thompson, K., & Song, W. (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*, 63–68.
7. Lis, M., Ren, P., Cho, M.H., Shim, K.S., Fletcher, C., Khan, O., & Devadas, S. (2011). Scalable, accurate multicore simulation in the 1000-core era. In *Performance Analysis of Systems and Software (ISPASS), IEEE International Symposium on* (pp. 175 –185).
8. Bartolini, A., Cacciari, M., Tilli, A., Benini, L., & Gries, M. (2010). A virtual platform environment for exploring power, thermal and reliability management control strategies in high-performance multicores. In: GLSVLSI'10. New York, NY, USA: ACM, pp. 311–316.
9. Zoni, D., Corbetta, S., & Fornaciari, W. (2012). Hands: Heterogeneous architectures and networks-on-chip design and simulation. In: IEEE ISLPED'12, aug.
10. Corbetta, S., Zoni, D., & Fornaciari, W. (2012). A temperature and reliability oriented simulation framework for multi-core architectures. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI2012), University of Massachusetts, Amherst*. USA.
11. Carlson, T., Heirman, W., & Eeckhout, L. (2011). Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation," in. In *High Performance Computing, Networking, Storage and Analysis (SC), 2011 International Conference for* (pp. 1–12).
12. Prabhu, S., Grot, B., Gratz, P.V., & Hu, J. (2009). Ocin tsim- dvfs aware simulator for nocs.
13. Peh, L.-S., & Dally, W.J. A delay model for router microarchitectures. *IEEE Micro*, *21*(1), 26–34. doi:10.1109/40.903059. Jan. 2001. [Online]. Available:.
14. Zoni, D., Terraneo, F., & Fornaciari, W. Source code." [Online]. Available: http://hipeaclab.deib.polimi.it.
15. Terraneo, F., Zoni, D., & Fornaciari, W. (2013). A cycle accurate simulation framework for asynchronous noc design. In: System on Chip (SoC), International Symposium on, Oct 2013.
16. ltc3589 datasheet. http://cds.linear.com/docs/en/datasheet/3589ff.pdf..
17. Alhussien, A., Wang, C., & Bagherzadeh, N. (2010). A scalable delay insensitive asynchronous noc with adaptive routing. In *Telecommunications (ICT), 2010 IEEE 17th International Conference on* (pp. 995–1002).
18. Panades, M.I., & Greiner, A. (2007). Bi-synchronous fifo for synchronous circuit communication well suited for network-on-chip in gals architectures. In *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on* (pp. 83–94).
19. Brooks, D., Tiwari, V., & Martonosi, M. (2000). Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of the 27th annual international symposium on Computer architecture, ser. ISCA '00. New York, NY, USA: ACM* (pp. 83–94).

20. Beigne, E., Clermidy, F., Miermont, S., & Vivet, P. (2008). Dynamic voltage and frequency scaling architecture for units integration within a gals noc. In *Networks-on-Chip, 2008. NoCS 2008. Second ACM/IEEE International Symposium on* (pp. 129–138).
21. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M.D., & Wood, D.A. The gem5 simulator. *SIGARCH Comput. Archit. News*, *39*(2), 1–7. Aug. 2011. [Online]. Available: doi:10.1145/2024716.2024718.
22. Agarwal, N., Krishna, T., Peh, L.-S., & Jha, N. (2009). Garnet: A detailed on-chip network model inside a full-system simulator. In: ISPASS.
23. Zhao, W., & Cao, Y. (2006). New generation of predictive technology model for sub-45nm design exploration. In: *Quality Electronic Design, ISQED '06. 7th International Symposium on*, pp. 6 pp. –590.
24. Butcher, J.C. Numerical methods for ordinary differential equations. In: J. Wiley, 2003. [Online]. Available: http://www.worldcat.org/isbn/9780471967583.
25. Duarte, D.E. (2002). *Clock network and phase-locked loop power estimation and experimentation*: Ph.D. dissertation, Pennsylvania State University.
26. Wentzlaff, D., Griffin, P., Hoffmann, H., Bao, L., Edwards, B., Ramey, C., Mattina, M., Miao, C.-C., Brown, J., & Agarwal, A. (2007). On-chip interconnection architecture of the tile processor. In: Micro.
27. Guthaus, M.R., Ringenberg, J.S., Ernst, D., Austin, T.M., Mudge, T., & Brown, R.B. (2001). Mibench: A free, commercially representative embedded benchmark suite. In *Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop. Washington, DC, USA: IEEE Computer Society* (pp. 3–14).
28. Keating, M., Flynn, D., Aitken, R., Gibbons, A., & Shi, K. (2007). *Low Power Methodology Manual: For System-on-Chip Design. Springer Publishing Company. In: Incorporated.*

Davide Zoni received the Master Degree in Computer Engineering from Politecnico di Milano in 2010 and the Ph.D. degree in Information Technology from Politecnico di Milano in 2014. He currently holds a Post-Doc position at the Dipartimento di Elettronica Informazione e Bioingegneria (DEIB), Politecnico di Milano, Italy. His main research interests include networks-on-chip, computer architecture as well as the application of control theory methodologies for power, performance and reliability optimizations in multi-cores. He received a best paper award in 2012 and two HiPEAC Collaboration Grants in 2013 and 2014. In FP7 he participates to the EU-funded HARPA project.

Federico Terraneo received his Ph.D. degree in Information Technology from Politecnico di Milano in 2015. Currently he holds a Post-Doc position at Politecnico di Milano – Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB). His research interests include embedded systems and the application of principles of control theory to the design of software systems. Since 2008 he is the main developer and maintainer of the Miosix embedded operating system.

William Fornaciari Ms, PhD, is Associate Professor at Politecnico di Milano. He published six books and around 200 papers, collecting 5 best paper awards, one certification of appreciation from IEEE and holds 3 international patents on low power design. Since 1993 he is member of program committees and chair of international conferences in the field of computer architectures, EDA and system-level design. Since 1997 he has been involved in 14 EU-funded international projects. In FP7 he has been WP leader for the IP COMPLEX projects and Project Technical Manager of 2PARMA (ranked as success story by the EU) and he also participates to the Artemis SMECY project on smart multicore embedded systems. Currently he is WP leader of the CONTREX project on mixed criticalities and Project Coordinator the HARPA project on embedded and HPC technologies to ensure dependable performance.

He cooperated for around 20 years with the Technology Transfer Center of POLIMI, gaining significant experience in cooperating with international companies for the development of leading edge products: industrial exploitation of research ideas is one of his main attitudes and teaching task, and in 2013 he created a startup company focusing on embedded systems. His main research interests cover multi-many core architectures, NoC, low power design, software power estimation, run time resource management, wireless sensor networks, thermal management, and EDA-based design methodologies. He is co-author of the first italian book on embedded systems and he acted as project reviewer for EC funded projects and invited speaker during EU consultation/information workshops. William Fornaciari is IEEE Senior Member.