J. Parallel Distrib. Comput. 71 (2011) 714–728

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

CAFES: A framework for intrachip application modeling and communication architecture design

César Marcon [a], Ney Calazans [a,*], Edson Moreno [a], Fernando Moraes [a], Fabiano Hessel [a], Altamiro Susin [b]

[a] Faculty of Informatics, Pontifical Catholic University of Rio Grande do Sul, Porto Alegre, Brazil
[b] Electrical Engineering Department, Federal University of Rio Grande do Sul, Porto Alegre, Brazil

ARTICLE INFO

Article history:
Received 3 March 2010
Received in revised form
27 September 2010
Accepted 5 October 2010
Available online 16 October 2010

Keywords:
NoC
Application mapping
Design framework

ABSTRACT

This paper describes CAFES, an extensible, open-source framework supporting several tasks related to high-level modeling and design of applications employing complex intrachip communication infrastructures. CAFES comprises several built-in models, including application, communication architecture, energy consumption and timing models. It also includes a set of generic and specific algorithms and additional supporting tools, which jointly with the cited models allow the designer to describe and evaluate applications requirements and constraints on specified communication architectures. Several examples of the use of CAFES underline the usefulness of the framework. Some of these are approached in this paper: (i) a realistic application captured at high-level that has its computation time estimated after mapping at the clock cycle level; (ii) a multi-application system that is automatically mapped to a large intrachip network with related tasks occupying contiguous areas in the chip layout; (iii) a set of mapping algorithms explored to define trade-offs between run time and energy savings for small to large intrachip communication architectures.

1. Introduction

Recent silicon technologies allow the implementation of complex Systems-on-Chip (SoC) with an excess of one billion transistors integrated in a single chip [3]. The amount of modules communicating inside such a chip can also be quite large, reaching several dozen modules [33]. Additionally, current and future data-intensive applications like multimedia point to growingly communication-centric systems [16]. This increases the need for special intrachip communication resources to cope with the tight requirements of SoC design. There are plenty of architectures for on-chip communication suitable to deal with such requirements, including split busses [10] and several flavors of networks on chip (NoCs) [4].

NoCs have been proposed to provide high scalability, reusability and reliability. They usually display features to fulfill specific requirements such as energy consumption and latency. However, their design also brings up new challenges. To illustrate these challenges Fig. 1 depicts the process of solving two relevant problems in NoC-based SoC design: the partitioning and mapping synthesis steps. The *partitioning synthesis step* defines an optimal grouping of tasks on application modules. The *mapping synthesis step* consists in finding an association of each of n tasks to a given location in the communication architecture that minimizes some (often complex) cost function. The cost function may include combinations of latency and/or power dissipation figures. Assuming there are n equal regions here called *tiles*, to where it is possible to assign any of the n modules, there are clearly n! possible distinct mappings. It is possible to devise trivial algorithms that inspect each mapping and find the best solution, but this is obviously unfeasible for SoCs with hundreds of modules. Thus, the search of a SoC optimal implementation requires efficient algorithms and sound models. Referring to Fig. 1, given an application defined initially by a set of tasks (t_i), the partitioning process groups tasks into nodes (n_i), while the mapping process binds these nodes to specific tiles (τ_i) of the communication infrastructure.

This paper proposes a framework called *Communication Analysis For Embedded Systems* (CAFES) to automate and/or ease the execution of design tasks at the architectural level. The initial development of CAFES targeted the evaluation of energy consumption in communication architectures. CAFES is extensible in several directions, and already displays capabilities to support intrachip communication architecture design tasks like: (i) Model applications in their communication and computation aspects; (ii) Model several features of different intrachip communication architectures; (iii) Develop and integrate algorithms for design tasks such as mapping and partitioning over an intrachip communication architecture; (iv) Estimate static and dynamic power dissipation, as well as message latency, for a set of intrachip

* Corresponding author.
E-mail address: ney.calazans@pucrs.br (N. Calazans).

0743-7315/$ – see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2010.10.002

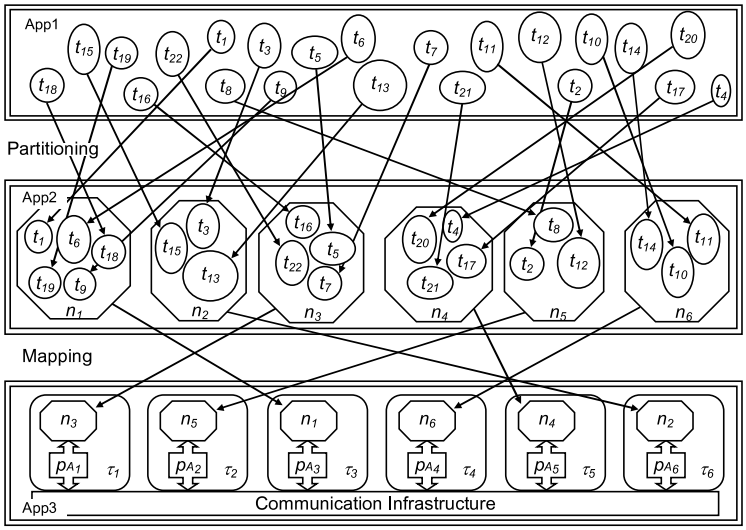


Fig. 1. Partitioning and mapping an application description (set of tasks) to a NoC-based SoC architecture.

communication architectures; (v) Evaluate application execution time at various abstraction levels; (vi) Estimate energy consumption and latency of a communication architecture for gate level descriptions; (vii) Generate synthetic applications for a set of application models; (viii) Automated model conversion; (ix) Generate application-specific traffic; (x) Support new algorithms, applications models and target architecture models through framework built-in extensibility features.

Under a user's point of view, CAFES is a framework for design space exploration of the communication infrastructure, enabling the evaluation of different mapping heuristics and communication models, and targeting latency and power/energy estimations. Other frameworks presented in the literature target the design space exploration of NoC structural parameters, such as number of virtual channels, topology, buffer depth, routing algorithm. Examples of such frameworks are ATLAS [24], the framework supporting the μSpider NoC [7], and commercial services like those offered by Arteris or INOCs Structured Interconnects.

This paper is organized as follows. Section 2 explores some related work. Section 3 introduces the main features and functionalities of the CAFES framework. Section 4 presents and compares models already supported by the framework to capture applications structure and/or functionality. The models consider communication and computation. Section 5 describes communication architecture models, and some models used for energy consumption and timing estimation. Together with models described in Section 4, these are used to estimate some relevant design requirements. Section 6 describes some of the additional CAFES built-in tools and facilities. Section 7 draws conclusions and depicts ongoing work.

2. Related work

NoC frameworks have to consider applications, SoC and/or MPSoC architectures and even overall system features. This section presents a brief account of some current works related to the topics addressed by the CAFES framework.

One encompassing framework for NoC generation and application production involves the *xpipes* compiler [14] and the SUNMAP [26] tools. The former can generate NoC components by specializing a library of soft macros (containing routers, network interfaces and link architectures), while the latter can help in the selection of a NoC topology and the customization of the topology to fulfill application requirements. All these tools base the development of NoC implementations from the Xpipes NoC architecture.

Krasteva et al. [17] present a framework for fast emulation and NoC prototyping. The framework provides a library composed by reusable hardware cores (i.e. pre-placed and pre-routed cores, and partial configuration files). Using partial FPGA reconfiguration can avoid whole system re-synthesis, enabling fast architecture analysis.

Palermo and Silvano [29] present PIRATE, a framework for high-level exploration of NoC power and performance trade-offs at varying traffic patterns. This work proposes a methodology to generate and to simulate a NoC whose interconnection elements and switches are configurable, which permits building different on-chip topologies. The framework includes facilities to: (i) generate several NoC topologies (e.g. Octagon, Cube and Mesh) in Verilog RTL; (ii) estimate power consumption by a power characterization flow; (iii) enable timing verification by cycle-based simulation.

Talwar et al. [32] present a study on NoC power, latency and throughput trade-offs. They vary micro architectural and circuit level parameters and use as support a NoC exploration framework capable of topology generation and comparison, using parameterized models of routers and links described in SystemC.

Dolif et al. [6] describe a framework for multi-task application mapping onto MPSoC platforms. A complete stochastic allocation and scheduling framework validate abstract models of system components and assess constraint satisfaction and objective function optimization. The framework provides an MPSoC virtual platform to accurately derive input parameters and allows describing an application by using models whose computation time, amount of communication and storage requirements are annotated. Then, the application can be simulated, and tasks are allocated and scheduled to an MPSoC execution platform.

Several other framework proposals can be found in the literature. One important distinguishing feature of CAFES is its capacity to support several application and abstract communication models of varying complexities and accuracies, as well as its open-source structure that supports extension to encompass new models.

3. CAFES framework architecture

This section describes the CAFES framework. Section 3.1 establishes the set of assumptions of the current version of the

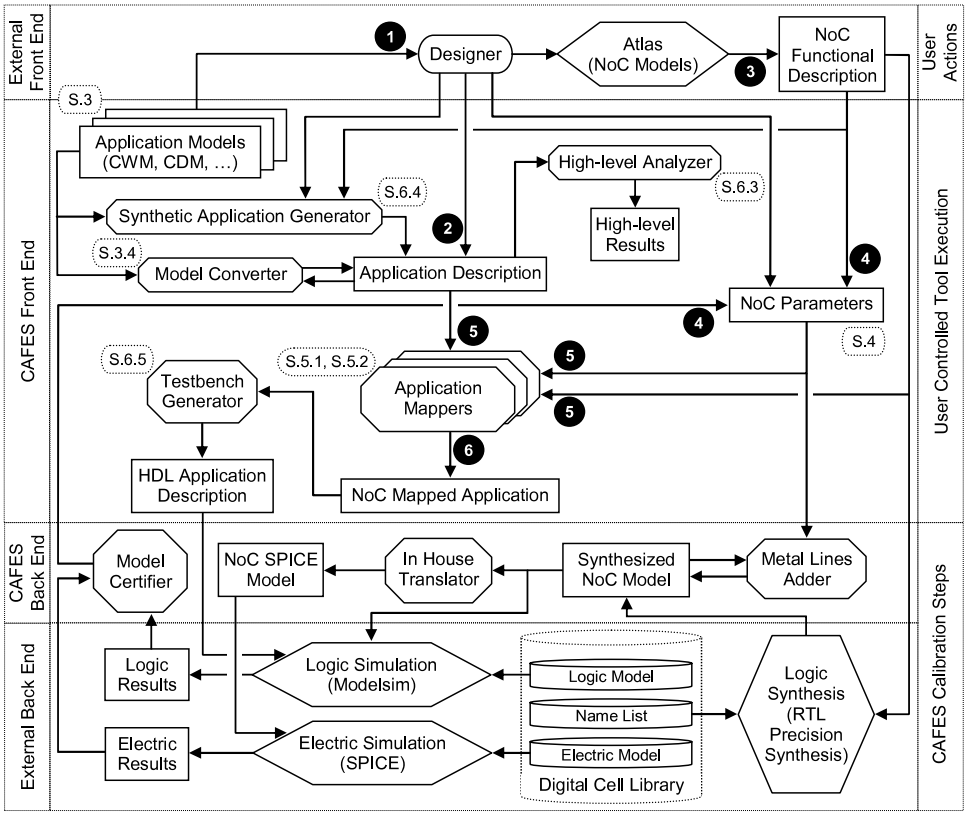


Fig. 2. CAFES framework and its interaction with external entities.

framework and gives some directions to where it may evolve. Sections 3.2–3.4 comprise the description of the framework architecture in itself. Finally, Section 3.5 gives a hint on how the CAFES user interface works.

3.1. CAFES framework assumptions

No framework can encompass the whole diversity of NoC-based systems existing today. CAFES was initially built with application scalability problems in mind. The software structure allows inserting new features and enforces the reuse of already available features, providing the environment with extensibility. To achieve this, some assumptions and constraints were necessary.

Among SoCs that employ NoCs as target communication architecture, many propose irregular and/or dedicated NoC topologies to account for power efficiency and maximizing performance. However, it is instructive to observe the related trends predicted by the ITRS. For example, the ITRS 2008 Update [13] estimates that the maximum number of processing modules (CPUs plus others) in a complex SoC for non-portable systems will grow from around 25 in 2010 to more than 280 in 2020. For portable systems SoCs values are even more impressive, going from around 80 modules in 2010 to more than 800 in 2020. Clearly, managing future systems design complexity and fulfilling their time to market demands may lead to increasing use of regular communication architectures. Accordingly, the CAFES framework supports a set of regular topologies, including 2D mesh and 2D torus. This enables simple and sound modeling with pre-characterized parameters, making it easier to evaluate several distinct communication architectures. Of course, expansions may be necessary to encompass other topologies, such as fat trees and butterflies as well as irregular topologies. CAFES also contains a library of algorithms dedicated to on-chip communication analysis, including generic optimization for mapping such as tabu search [9] and simulated annealing [34]. These can be adapted to solve other problems such as partitioning and/or incremented with alternative algorithms for these and other problems. The framework also contains a library to support the manipulation of generic data structures, such as graphs and binary search trees, which are normally used to build application and communication architecture descriptions.

At the low end of the implementation issues, all uses of the framework dealt so far only with synchronous systems. However, it is well accepted today that complex systems cannot be fully synchronous. Accordingly, CAFES models, especially lower abstraction models, need to consider other communication architectures paradigms, such as globally asynchronous, locally synchronous (GALS) and fully asynchronous.

3.2. CAFES framework general structure

Fig. 2 shows the structure of the CAFES framework and its interaction with external entities. This figure is separated into four horizontal regions: two related to the CAFES internal structure (central regions) and two other, related to external entities (top and bottom regions). Integer numbers inside a dark circle refer to the basic flow for solving the mapping problem within CAFES, as described in Section 3.3, and section numbers inside dotted rounded rectangles point to the paper section that details the topic.

The top region in Fig. 2, called External Front End comprises the (human) application designer and the ATLAS framework [24] that can generate synthesizable NoC models used by CAFES. The next region, CAFES Front End, is the focus of the work and will be explored in detail later in this section. Next, the CAFES Back End region enables two kinds of actions: (i) Certification of high-level models from the CAFES Front End, through the use of both simulation and tuning of high-level models from lower level information; (ii) Synthesis of lower level descriptions from high-level descriptions. Finally, the bottom region (External Back

End) shows some of the employed commercial design tools and libraries. The Back End regions serve the purpose of *calibration* of the environment. They define a flow that executes just once for a given technology-NoC type combination.

3.3. CAFES front end

Referring to the circled numbers in Fig. 2, obtaining a solution to the mapping problem starts when the designer chooses an application model to use (1). With this model he employs related tools from the environment, such as graphic editors, to produce the application description (2). Next, the use of ATLAS [24,28] allows producing a NoC functional description (3) [25]. During this phase, ATLAS and CAFES generate a set of relevant NoC parameters (4). The application description, the NoC functional description and the NoC parameters are inputs (5) to a mapper tool that produces a NoC-mapped application description (6). Besides this basic flow, the CAFES Front End comprises several accessory tools that complete its functionality. Some degree of compatibility exists among application models. Accordingly, the environment offers a Model Converter tool with functionality discussed in Section 4.7. Additionally, CAFES supports a straightforward analysis of application characteristics from the initial description, using the High-level Analyzer tool, approached in Section 6.3. To enable fast production of benchmark examples, CAFES includes a Synthetic Application Generator, described in Section 6.4. After mapping an application to a synthesizable NoC description it is possible to simulate the whole system at the RTL level, as long as the Testbench Generator tool is used, as described in Section 6.5.

3.4. CAFES back end and external back end

To implement the model certification and synthesis actions, the CAFES Back End includes in-house and commercial tools. The main back end data repository is called the Digital Cell Library in Fig. 2 (e.g. TSMC or IBM) [15]. With the Digital Cell Library set up, an instance of a NoC functional description (RTL) undergoes logic synthesis. This generates a low level NoC description, which serves to refine a model that includes long metal lines delays, computed with the CAFES Metal Lines Adder tool. The refined description is the input of logic and electrical simulations, by selecting specific modules of the NoC. The obtained logic and electrical descriptions are then simulated and the resulting simulation data is input to the Model Certifier that compares logic and electrical simulation results with previous electric NoC Parameters. This enables the successive certification and tuning of the original NoC parameters. After concluding this calibration step, the framework is able to provide accurate high-level estimations for the design space exploration process. No recourse to these calibration steps is necessary during regular use of CAFES for design exploration.

3.5. CAFES user interface

From the opening screen of CAFES, partially displayed in Fig. 3, the designer may select an application model and target communication architecture. He may also select a specific synthesis/analysis tool (on the Tools Menu) to use during the communication architecture analysis. Section 4 details the application models depicted in the figure.

CAFES employs some basic communication architecture parameters for energy consumption and execution time estimations. At the moment, these parameters have default values characterized according to the target architecture and pre-defined CMOS technologies. Nevertheless, the designer may change any of these values to fit other technologies. This can take place after choosing a

Fig. 3. CAFES initial screen, displaying the choice of application models and target communication architecture.

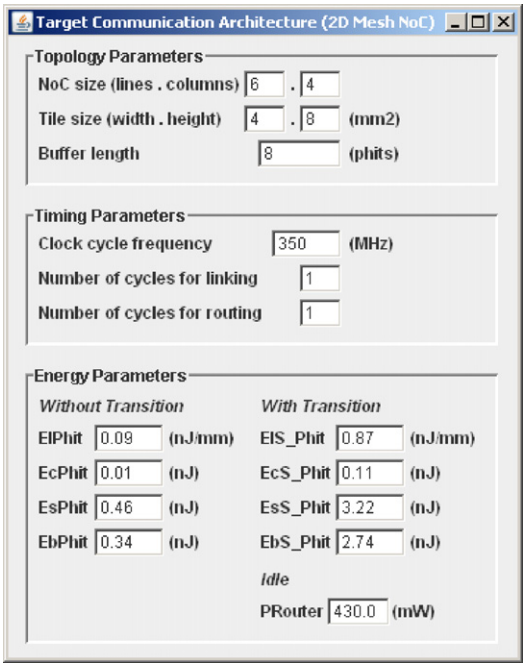
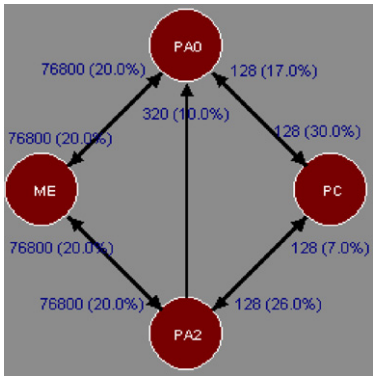


Fig. 4. Interface for setting target architecture parameters for 2D mesh topology and applications modeled with ECWM. The displayed energy parameters correspond to the CMOS TSMC 0.35 μm technology.

given application model. For instance, when the Extended Communication Weight Model (ECWM) is selected together with a 2D mesh topology NoC, XY routing and wormhole switching, CAFES shows the parameters as illustrated in Fig. 4. Section 5.2 provides a discussion on how to define these parameters. Most parameters are independent of application models. Exceptions are the *NoC Energy Parameters*, since only ECWM considers the bit transition effect, as described later, in Section 4.2.

4. Application models

Taking into account only the communication architecture design, the most relevant aspects of an application model are: (i) the exact instant when a communication occurs and for how long this communication occupies communication resources; (ii) the amount of data communicated; and (iii) the communication data pattern. Even when the number of aspects is reduced, there are several possible application models with different features that



(a) ECWM graphic description.

```
#numberOfLines numberOfColumns
 2      2
#_ECWG_GraphicVertices
#module xPosition yPosition
 ME     50      150
 PA0    250     50
 PA2    250     250
 PC     450     150
#_ECWG_Edges
#source-target phits transition(%)
 ME   - PA0     76800   20
 ME   - PA2     76800   20
 PA0  - PC      128     30
 PA0  - ME      76800   20
 PA2  - PA0     320     10
 PA2  - PC      128     7
 PA2  - ME      76800   20
 PC   - PA0     128     17
 PC   - PA2     128     26
```

(b) Corresponding textual description.

Fig. 5. Example of a synthetic application captured with ECWM.

can be employed. According to the chosen application model, it is possible to evaluate some application requirements and constraints with different algorithms, creating distinct trade-offs of accuracy, computation time and memory consumption. Depending on the selected application model, the designer may explore distinct computation and/or communication aspects. Most models proposed here are independent of the adopted target communication architecture. In addition, the framework supports extensions of the built-in models to other application models. Currently CAFES includes six application models and the Model Converter tool that are described in the rest of this section.

4.1. Communication weight model (CWM)

CWM models an application by its communication volume, which is defined by the number of bits transmitted by all communications during application execution. CWM is the simplest of CAFES built-in model. It is normally dedicated to estimating dynamic energy consumption of the target communication architecture. CWM facilitates the application modeling, since it is possible to use e.g. RTL simulation to count the number of bits transmitted by any source to any target. Similar models exist, including the *Core Graph* of Murali and De Micheli [27] and the *Application Characterization Graph* of Hu and Marculescu [11].

CAFES implements CWM using a *communication weight graph* (CWG), which is defined as a directed graph CWG $= \langle M, C\rangle$, where $M = \{m_1, m_2, \ldots, m_n\}$ is the set of application modules, corresponding to the CWG vertices, and $C = \{(m_i, m_j, w_{ij}) | (m_i, m_j) \in M$ and $w_{ij} \in \mathbb{N}^*\}$ denotes the communications channels between application modules, corresponding to the CWG edges. The edge weight w_{ij} in (m_i, m_j, w_{ij}) represents the amount of bits transmitted from m_i to m_j.

4.2. Extended communication weight model (ECWM)

According to [23], to estimate the energy consumption of the communication architecture accurately, it is important to know not just the amount of bits transmitted but also how many transitions occur during the transmission. The absence of this information can lead to errors of up to 45% in the energy consumption estimation. To minimize this error, the authors of [23] proposed to improve CWM by including the number of transitions occurred between successive bit transmissions. This new model is called ECWM.

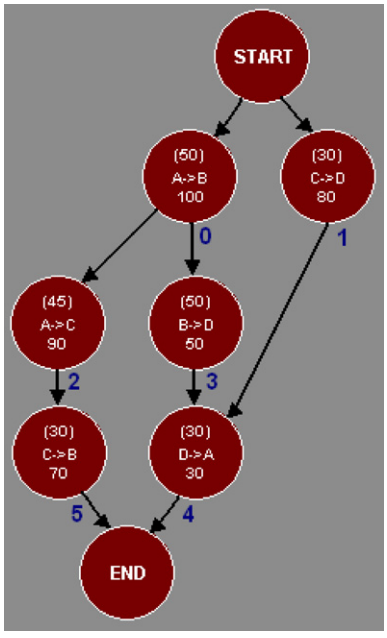
CAFES implements ECWM using an *extended communication weight graph* (ECWG), which is defined as the directed graph ECWG $= \langle M, T\rangle$. As defined before, M is the set of application modules and w_{ij} is the amount of communications. Assuming σ_{ij} is the number of bit transitions occurred on all packets sent from m_i to m_j, then $T = \{(m_i, m_j, w_{ij}, \sigma_{ij}) | m_i, m_j \in M, w_{ij} \in \mathbb{N}^*, \sigma_{ij} \in \mathbb{N}\}$ is the ECWM edge set. In practice, σ_{ij} may be expressed as a percentage of w_{ij}. T represents all communications between modules, containing both the amount of bits and the amount of bit transitions.

The designer may describe ECWG manually or extract ECWG automatically, during the application RTL simulation, by capturing the volume and switching activity of the bit traffic in specified NoC channels. Fig. 5(a) and (b) display graphic and textual descriptions of a simple synthetic application with 4 modules and 9 communications. For example, module PA2 sends 320 *phits* to module PA0, having bit transition activity of 10%. *Phit* is the physical width in bits of the link between routers or between a router and a module. Assuming a *phit* composed by 8 bits, this would correspond to 256 bit transitions.

4.3. Communication dependence model (CDM)

CDM, introduced in [19], models an application according not only the communication volume, but also to the communication dependence. The dependence information, which improves CWM, allows capturing concurrent requests to a same resource of the target architecture, thus enabling the design phase to evaluate, avoid or minimize message contention events. A basic assumption of the CDM is that any pair of *non-dependent* communications that *may compete for some resource will cause contention* in the model. In this sense, CDM is a pessimistic model that enables computing and reducing contention. In order to reduce or avoid contention, the CDM algorithm searches for better mapping scenarios, i.e. the ones where non-dependent communications may not compete for some resource.

CAFES implements CDM with the *communication dependence graph* (CDG), which is defined as an acyclic directed graph CDG $= \langle P, D\rangle$. P is the set of all application messages, corresponding to the CDG vertices. Additionally, P also contains two special vertices named START and END, which represent the beginning and the end of the application flow defined in a CDG. Given $P = \{p_n = (n, m_i, m_j, w_{ijn}) \mid n \in \mathbb{N}, m_i, m_j \in M$ and $w_{ijn} \in \mathbb{N}^*\} \cup \{\text{START, END}\}$, p_n is the nth message sent from m_i to m_j containing w_{ijn} bits and n is an identifier that permits distinguishing different messages exchanged by a same pair of modules. D is the set of CDG edges that represents the communication dependence. Considering that p_r and p_n are application messages such that $n \neq r$, then $D = \{(p_r, p_n) | p_r, p_n \in P\}$, where $(p_r, p_n) \in D$ if and only if p_r depends on p_n. A message p_r without dependence on another message implies an edge connecting the START vertex to p_r. A message p_r which no other message depends on implies an edge connecting p_r to the END vertex.



(a) CDCM graphic description. Numbers outside each circle are drawn just to improve identification correlating graphic and textual descriptions. They do not actually exist in the graphic view.

```
#nLines nColumns
 2   2
#_CDCG_Graphics
#id xPosition yPosition
 START  470  45
 END    330  350
 0      390  115
 1      560  115
 2      260  205
 3      390  205
 4      390  295
 5      260  295
#_CDCG_Vertices
#id source-target phits:time
 0  A - B  100 : 50
 1  C - D   80 : 30
 2  A - C   90 : 45
 3  B - D   50 : 50
 4  D - A   30 : 30
 5  C - B   70 : 30
#_CDCG_Edges
#id : Dependences
 START : 0  1
 END   :
 0     : 3  2
 1     : 4
 2     : 5
 3     : 4
 4     : END
 5     : END
```

(b) Corresponding ECWM textual description.

Fig. 6. Example of a synthetic application captured with CDCM.

4.4. Communication dependence and computation model (CDCM)

CDCM, introduced in [20], add features to CDM. Besides capturing all information that is conveyed by CDM, CDCM includes the *computation time* on the source module that precedes each communication initiated by this module. This model allows estimating precisely figures like application execution time, static energy consumption and power.

CAFES implements CDCM with the *communication dependency and computation graph* (CDCG), which is defined as an acyclic directed graph CDCG $= \langle Q, D \rangle$. Likewise to CDM, D is the set of edges, representing the communication dependences. However, the CDCM message model enhances the CDM message model by adding a t_i parameter. This parameter represents the computation time of the source module (m_i) after having all vertex dependences of vertex i satisfied, until the transmission of message n. The parameter t_i is a natural number that represents the number of cycles of the m_i clock. The set of vertices is thus $Q = \{q_n = (n, m_i, m_j, w_{ijn}, t_i) | n \in \mathbb{N}, m_i, m_j \in M, w_{ijn} \in \mathbb{N}^*,$ and $t_i \in \mathbb{N}\} \cup \{\text{START}, \text{END}\}$.

Fig. 6(a) illustrates a graphic description of a CDCM synthetic example and Fig. 6(b) shows the corresponding textual description.

The set of vertices Q is $\{q_0, q_1, q_2, q_3, q_4, q_5, \text{START}, \text{END}\}$, such that $q_0 = (0, A, B, 100, 50)$, $q_1 = (1, C, D, 80, 30)$, $q_2 = (2, A, C, 90, 45)$, $q_3 = (3, B, D, 50, 50)$, $q_4 = (4, D, A, 30, 30)$ and $q_5 = (5, C, B, 70, 30)$, and D is $\{(\text{START}, q_0), (\text{START}, q_1), (q_0, q_2), (q_0, q_3), (q_1, q_4), (q_2, q_5), (q_3, q_4), (q_4, \text{END}), (q_5, \text{END})\}$. It means, for instance, that q_0 and q_1 are possibly concurrent messages, while q_0 and q_4 can never compete for an identical communication resource, since q_4 is dependent on q_0.

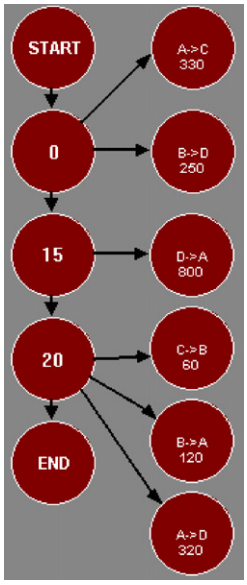
4.5. Application communication pattern model (ACPM)

ACPM, introduced by Kreutz et al. in [18], models applications through a totally ordered set of events, where each event is a message associated to a tag. This tag does not carry any kind of timing information; it is only responsible for event ordering. When implemented inside the framework, ACPM was improved by changing the tag meaning. Now, the tag represents the instant of time that an associated message is sent from the source to the target module.

CAFES implements ACPM with the *application communication pattern* (ACP). This is simply the totally ordered set of events. Let $G = \{g_0, g_1, \ldots, g_k\}$ be the set of messages exchanged during an application execution and M be the set of application modules. Then, each message g_k is modeled as $g_k = (m_i, m_j, w_{ijn})$ with $m_i, m_j \in M$ and the weight $w_{ijn} \in \mathbb{N}^*$, is defined as in previous models. In addition, let Γ_θ be a subset of G, such that each Γ_θ has an associated time tag θ that gives the initial sending instant for all messages of Γ_θ. Then, ACP $= \{(\theta, \Gamma_\theta) | \Gamma_\theta \subseteq G, \Gamma_\theta \neq \emptyset, \theta \in \mathbb{N}\} \cup \{\text{START} = (\emptyset, \emptyset), \text{END} = (\emptyset, \emptyset)\}$. START and END are special tuples of ACP determining respectively the start and the end of the arrangement.

Fig. 7(a) illustrates a graphic description of a synthetic example described with ACP and Fig. 7(b) shows the corresponding textual description. Here, six messages compose the communication of a synthetic example $g_0 = (A, C, 330)$, $g_1 = (B, D, 250)$, $g_2 = (D, A, 800)$, $g_3 = (C, B, 60)$, $g_4 = (B, A, 120)$ and $g_5 = (A, D, 320)$, performing the set of messages $G = \{g_0, g_1, g_2, g_3, g_4, g_5\}$. Messages are grouped into the following three sets: $\Gamma_0 = \{g_0, g_1\}$, $\Gamma_{15} = \{g_2\}$ and $\Gamma_{20} = \{g_3, g_4, g_5\}$, according to their start sending instant. Each set of messages is associated to the corresponding tag to accomplish the ACP $= \{\text{START}, (0, \Gamma_0), (15, \Gamma_{15}), (20, \Gamma_{20}), \text{END}\}$ structure.

It is easy to obtain an application description with ACPM, because besides source and target modules it is only necessary to annotate the volume and the instant of each communication during a high level simulation. On the one hand, module mapping may change the communication instants, compromising the timing estimation precision of this model. On the other hand, communication energy estimations are more accurate here than using CWM because it allows including static energy estimations.



(a) ACP graphic description.

```
#_NoC_Size
#numberOfLines numberOfColumns
 2 2

#_ACPG_TagsGraphic (list of Tags)
# tag xPosition yPosition
 START     50    50
 0         50    125
 15        50    200
 20        50    275
 END       50    350

#_ACPG_VerticesGraphic (vertices)
# tag source - target phits : x y
 0    A - C    330   :  300   50
 0    B - D    250   :  300   125
 15   D - A    800   :  150   200
 20   C - B    60    :  300   200
 20   B - A    120   :  300   275
 20   A - D    320   :  300   350
```

(b) Corresponding textual description.

Fig. 7. Example of a synthetic application captured by ACPM.

Table 1
A comparison of the CAFES application models. CV = *Communication volume*.

Model	Application view	Modeled features	
		Communication	Computation
CWM	Module	CV	–
ECWM	Module	CV + switching activity	–
CDM	Module	CV + message partial order (untimed)	–
CDCM	Module	CV + message partial order (untimed)	Module execution time
ACPM	Module	CV + message total order (timed)	–
CTM	Task	CV	Task scheduling

4.6. Communication task model (CTM)

Hu and Marculescu introduced the communication task model (CTM) in [12]. This model captures the computation and communication of an application through task scheduling and communication volume. Unlike the communication dependence assumed in CDM and CDCM, CTM takes into account the task scheduling dependence.

CAFES implements CTM with the *communication task graph* (CTG), which is defined as the acyclic directed graph CTG = $\langle V, B\rangle$, where V is the set of vertices and B is the set of edges. A vertex $v_i = (d_i, \varphi_i, Ei)$ represents characteristics of the application task $i(t_i)$. The term d_i is the deadline for t_i conclusion, while φ_i and E_i are vectors whose elements represent the task execution time and the task energy consumption on the processing elements of the target architecture, respectively. Each edge $b_{ij} \in B$ characterizes the dependence of control and communication between the pair of vertices (v_i, v_j). To each b_{ij} there is an associated amount of bits transmitted from v_i to $v_j(w_{ij})$. Additionally, CTG contains two special vertices named START and END, which represent the beginning and the end of the application.

4.7. A taxonomy for application models and model conversion

Sections 4.1–4.6 described the six models currently supported inside CAFES. These models provide a spectrum of choices for capturing application characteristics, but fully understanding them is challenging. This section attempts to help in this, comparing models as to what features can or cannot be modeled with each, and providing relations between the abstractions defined by each.

Table 1 compares all CAFES models. Functional and structural are two orthogonal but related *views* of an application. Functionally, a set of *tasks* forms an application, while structurally a set of *modules* implements it. Most models in CAFES use the structural view, except for CTM, which uses the functional view. The communication among tasks or modules is the fundamental information to model, and every model allows capturing it. A model like CWM that captures only communication volume enables to estimate dynamic energy consumption in the communication architecture. This is limited, because congestion and its effects cannot be precisely modeled with this kind of information. To enhance precision, models like CDM, CDCM and ACPM can capture the ordering of communication, using either partial or total orders for these events. Estimating the total execution time is important for system designers. Accordingly, some models in CAFES enable the estimation of total execution time, either by computing the execution time of modules (in CDCM) or the moment at which tasks are scheduled (in CTM). Note that in ACPM the execution time of modules is implicitly considered, since it implies a total order for messages.

To illustrate and explain the relationship among models, Fig. 8 shows in its top middle portion an example application composed by five tasks $(t_1, t_2, t_3, t_4, t_5)$ that are partitioned in two modules (m_1, m_2). A partitioning choice is illustrated by the plain arrows from tasks to modules. The dotted arrows show how the models available in CAFES allow capturing the communication and/or computation aspects of application modules or application tasks. Some models can be automatically extracted from others that contain greater amount of details. These automatic conversions define an abstraction relation between models. Possible automatic

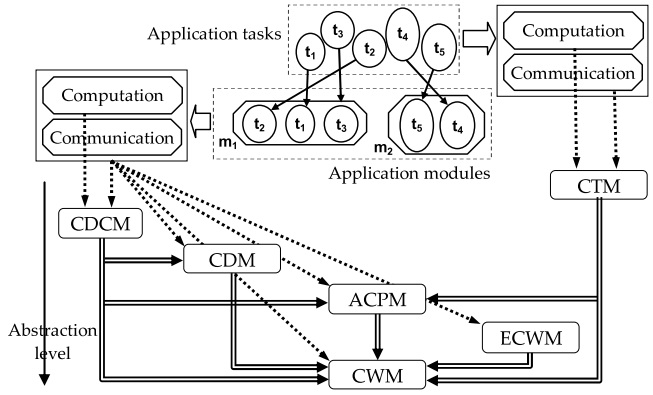


Fig. 8. Application models composition regarding computation and communication. Double solid lines indicate possible inter-model conversions. Dotted lines indicate which model captures which aspect (computation or communication) of an application view (functional or structural).

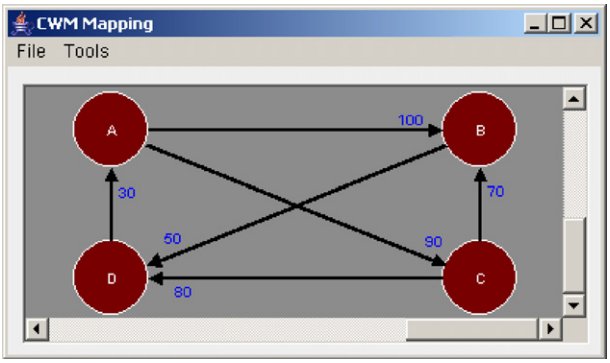


Fig. 9. An example of CWG that results after using the *CDCM to CWM conversion tool* on the CDCG of Fig. 6(a).

model extractions are shown in Fig. 8 using double line arrows. For instance, ACPM may be extracted from CDCM and CTM. On the other hand, no automatic conversion can obtain CDCM, CTM or ECWM, due to their amount of details, not encompassed by any other of the discussed models. Any model can be converted to CWM, which is thus the most abstract model of the CAFES model set.

To exemplify the use of model conversion tools, Fig. 9 shows the CWG that results when applying the *CDCM to CWM conversion tool* over the CDCG data structure of Fig. 6. Clearly, this last graph is generated when abstracting the computation time information from the CDCG, taking into account only the communication volume information.

This application inter-module conversion tool was used in some works to evaluate how complete an application model is to estimate a given design requirement. For instance, [20] demonstrates that CWM is a simpler model, which is easy to extract from the application specification, but an energy estimation with CWM may imply up to 20% of error when compared to an estimation achieved with CDCM.

5. Communication architecture modeling

This section details some of the fundamental structures and models used inside CAFES to model intrachip communication architectures. Rather than being complete, the goal of the section is to illustrate the process of employing models inside the framework. First, Section 5.1 defines the graph used to describe the overall structure of communication architectures. Section 5.2 then discusses an example energy model built inside CAFES. Timing models are important in most design aspects of communication architectures and they are deeply dependent on structural design choices like routing algorithms, buffering strategies and flow control options. Accordingly, Section 5.3 presents an example computation of a timing model used inside CAFES. Finally, Section 5.4 approaches the influence of the application model choice over energy and latency computations.

5.1. Communication resource graph (CRG)

CAFES supports some of the communication architectures considered as good candidates to become mainstream in future SoCs, including 2D mesh and 2D torus NoCs. To model the functionalities of these communication architectures, CAFES employs a structure called the *communication resource graph* (CRG), formally defined as a directed graph CRG $= \langle T, \Pi \rangle$, where the set of vertices $T = \{\tau_1, \tau_2, \ldots, \tau_p\}$ denotes the set of regions where modules are placed (the tiles). The set of edges $\Pi = \{(\tau_i, \tau_j) | \tau_i, \tau_j \in T\}$ designates the set of direct connections existing from tile i (τ_i) to tile j (τ_j). In regular geometries of NoC topologies, tiles are usually represented by their Cartesian coordinates, i.e. instead of using the notation τ_6 to stand for the sixth tile of a 2D NoC being positioned into the coordinates [3, 2], the notation used is $\tau_{[3,2]}$.

The CRG does not encompass the description of the employed routing algorithm. However it does impose constraints on the kind of routing algorithm used. The composition of the CRG and the routing algorithm allows defining possible paths to follow between communicating modules. Based on these computed paths and on estimations of tile physical dimensions, it is possible to define the links and routers used during communication. Therefore, it is possible to calculate information like energy consumption or latencies.

5.2. Energy model

Reducing energy consumption remains one of the main goals of actual electronic designs [1,31,2]. Each design must have several parts of it analyzed individually as is the case of very long wires and large memory and logic blocks. At design time, sound energy models may predict with reasonable precision the whole system consumption or even the energy consumption of some isolated parts, enabling to search for optimal solutions, as soon as possible. In this sense, this section shows how the basic energy consumption model of CAFES was developed.

An energy model uses structural and technological information to estimate dynamic and static energy consumptions of the target communication architecture. Nevertheless, each kind of communication architecture has its own set of topological and physical peculiarities, making it infeasible to use a single energy model for all purposes and all NoCs. To illustrate how energy models are built, this section shows the basis for energy consumption models, and a specific equation to estimate the dynamic energy consumption of a specific 2D mesh NoC.

CAFES models the dynamic energy consumption, using the concept of *bit energy* (EBit), similarly to models described elsewhere, like in [35,8]. Nevertheless, some of the models specified here consider also the effect of the number of bits transmitted and the number of bit transitions during message transfer [30]. For several communication architectures, EBit can be expressed as a function of four variable quantities, as depicted by Eq. (1). Here, EsBit is the dynamic energy consumption of a single bit on wires and on logic gates of each router. EbBit is the bit dynamic energy consumption on router buffers. EcBit is the dynamic energy consumption of a single bit on links between routers and the local module. ElBit is the bit dynamic energy consumption on the links between routers

$$\text{EBit} = \text{function}(\text{EsBit}, \text{EbBit}, \text{EcBit}, \text{ElBit}). \quad (1)$$

To exemplify the modeling of dynamic energy consumption, Eq. (2) illustrates how EBit is composed to model a 2D direct mesh NoC. It computes the dynamic energy consumed by a bit passing in such a NoC from tile i (τ_i) to tile j (τ_j), where η_{ij} corresponds to the number of routers that the bit traverses

$$\text{EBit}_{ij} = \eta_{ij} \times (\text{EsBit} + \text{EbBit}) + 2 \times \text{EcBit} + (\eta_{ij} - 1) \times \text{ElBit}. \quad (2)$$

According to the application model, CAFES decomposes each one of the four variable quantities of EBit in two new energy parameters: one that reflects the dynamic energy consumption when consecutive bits have opposite values, and another that does not consider the bit transition effect. This decomposition may be observed in the field *NoC energy Parameter* of Fig. 4. As noticeable in that figure, to inform the EsBit, EbBit, EcBit and ElBit the designer has to supply EsPhit, EbPhit, EcPhit and ElPhit parameters, which are respectively analogous to the first set of parameters, but express the energy consumed at phit level in the target communication architecture.

The NoC static energy consumption, on the other hand, depends on the number of transistors of the target communication architecture, which grows linearly with the number of tiles. To estimate the NoC static energy consumption inside CAFES, the designer starts supplying an estimation of the static power dissipated in a single router, which is the PRouter parameter in Fig. 4. This value is multiplied by the number of NoC routers ($|T|$) providing the total NoC power consumption, as stated by Eq. (3)

$$\text{PStNoC} = |T| \times \text{PRouter}. \quad (3)$$

Finally, the static energy consumed by the application is computed by Eq. (4) that multiplies the total dissipated power by the application execution time (texec). This last parameter may be obtained by simulating the application or with the help of timing models

$$\text{EStNoC} = \text{PStNoC} \times \text{texec}. \quad (4)$$

5.3. Timing model

CAFES uses timing models to compute message latencies, application execution time, and static energy consumption. However, in NoCs the computation of these values is strongly dependent on the specific communication architecture, on the routing scheme and on the employed switching mode. Due to this, it is not possible to have a single generic model that satisfies any communication architecture with reasonable accuracy.

This section exemplifies the composition of one timing model for the 2D mesh NoC used in Section 5.2, assuming wormhole switching and deterministic XY routing, which is one of the timing models already available in CAFES framework. This model considers (i) the *routing delay*, defined as the time necessary for a packet header to reach the target tile, and (ii) the *payload delay*, which depends only on the remaining phits of the packet. It should be kept in mind that the following discussion assumes the use of a fully synchronous communication architecture.

Let λ be the clock period of the communication architecture, nR be the number of cycles necessary to define the routing of a packet at each router, nL be the number of cycles necessary to transmit a single phit between routers, and nl be the number of cycles necessary to transmit a single phit between a router and its local module. These are the *NoC Timing Parameters* of Fig. 4. Then, Eq. (5) represents the minimum routing delay ($\text{d}R_{ij}$) of a packet going from tile i (τ_i) to tile j (τ_j) passing by η_{ij} routers without contention

$$\text{d}R_{ij} = (\eta_{ij} \times nR + 2 \times nl + (\eta_{ij} - 1) \times nL) \times \lambda. \quad (5)$$

Let nP_q be the number of phits of the q-th packet, going from module a (m_a), placed in τ_i, to module b (m_b), placed in τ_j. Considering that nl is equal to nL (which is the case for the NoC under analysis), then Eq. (6) represents the payload delay ($\text{d}P_{ijq}$)

$$\text{d}P_{ijq} = ((nP_q - 1) \times nl) \times \lambda. \quad (6)$$

The composition of Eqs. (5) and (6) generates Eq. (7) that allows computing the total q-th packet delay (d_{ijq}) for the 2D mesh NoC in this example

$$d_{ijq} = (\eta_{ij} \times (nR + nl) + nPq \times nl) \times \lambda. \quad (7)$$

5.4. Exemplifying the composition of communication models with application models

To estimate the energy consumption of a communication architecture and the application execution time, it is necessary to associate the application model with timing and energy models for the former. For instance, to estimate the energy consumption taking into account an application modeled by CWM, it is necessary to associate the EBitij concept developed in Section 5.2, with the volume of bits traversing a given edge of the CWG (i.e. the weight of the edge under consideration). In this sense, Eq. (8) gives the dynamic energy consumed by all communications from module a (m_a) to module b (m_b), considering that m_a and m_b are respectively mapped into tile i (τ_i) and tile j (τ_j)

$$\text{EBit}_{abij} = w_{ab} \times \text{EBit}_{ij}. \quad (8)$$

Let $|C|$ be the total number of communications described in some CWG. Then, Eq. (9) provides an estimation of the dissipated *NoC dynamic energy* (EDyNoC) considering all inter-module communications

$$\text{EDyNoC} = \sum_{q=1}^{|C|} \text{EBit}_{abij}(q). \quad (9)$$

Finally, to compute the total energy dissipation of the NoC (ENoC), the framework adds the static energy computed with Eq. (4) with the dynamic energy computed with Eq. (9), as described by Eq. (10)

$$\text{ENoC} = \text{EDyNoC} + \text{EStNoC}. \quad (10)$$

CWM does not capture application execution time (texec) information. In fact, this model only allows estimating the minimum time spent during communications, it does not consider contention effects nor the exact moments when packets are effectively transmitted. To obtain better estimations for texec, CAFES has available more powerful models, such as CDCM and ACPM, which enable the use of more elaborate timing models described elsewhere [20,21,18]. An example of how to compute texec using such models appears in Section 6 that explores an object recognition application modeled with CDCM.

6. Framework supporting tools and extensions

CAFES is a Java application that includes a set of generic classes and some algorithms specially implemented for allowing the construction of application tools. Section 6.1 explores the use of the framework for solving the mapping task and some other tools implemented in the current CAFES version. Section 6.2 discusses the current state of the CAFES algorithms library, a set of generic algorithms available for developing specific analyses over communication architectures. The environment provides some high-level estimation tools, as briefly exemplifies Section 6.3. The last Sections 6.4 and 6.5 respectively explore the relevant topic of synthetic application automatic generation and traffic scenarios to apply over synthetic or real application descriptions.

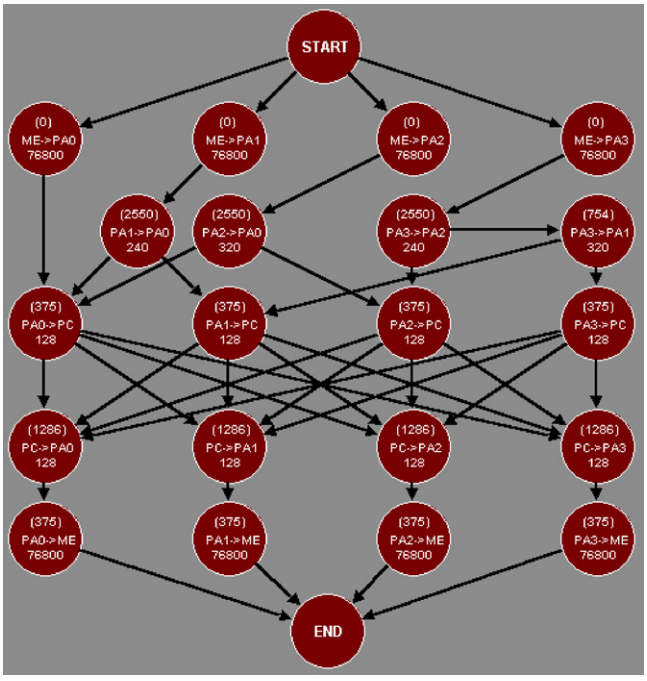


Fig. 10. A CDCM graphic description of a realistic object recognition application using CAFES. The application employs a distributed image segmentation algorithm.

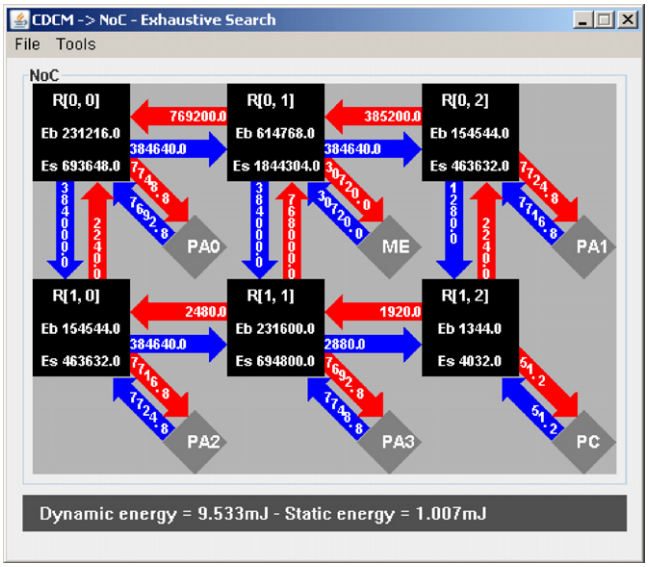


Fig. 11. An automatically generated mapping for the application of Fig. 10 on a 2×3 mesh topology NoC. Each channel (arrows) and router (larger rectangles) is marked with its estimated total dynamic energy consumption.

6.1. Application mapping task

The designer starts by choosing a model to describe the application behavior and selecting a target NoC from the available communication architectures list. He may use the default parameters or provide his own parameters that fit technological and geometrical aspects of the considered communication architecture. Then, using an internally available algorithm, the designer may extract an optimized mapping that reduces energy consumption and saves execution time.

As an example, consider an object recognition application using a 2D image segmentation technique. This application implements a distributed algorithm that splits a 640×480 pixels image into a 2D matrix of squares, associating each subarea to a processing element (PE). The example splits the image in four 76 800 pixels squares. Then, there are four PEs (PA0, PA1, PA2 and PA3) that process data from each square, a memory (ME) containing the whole image data and a central PE (PC) that coordinates and synchronizes system operation. The CAFES graphic (or textual) interfaces allow capturing the application description according to a given model. Fig. 10 illustrates this application modeled with CDCM, defined in Section 4.4. Each vertex different from START and END in the graph corresponds to a message exchanged through the NoC and contains the computation time, the source-target identifiers and the amount of transmitted bits.

By selecting a mapping algorithm available in CAFES and configuring it to minimize the dynamic and static energy consumption, the framework generates the mapping portrayed in Fig. 11. In this figure, large black squares represent NoC routers, while small gray squares represent processing modules. Inside each router, *Eb* represents the energy dissipated in buffers and *Es* the energy dissipated on other structures of the router (wires and combinational logic). In addition, links are annotated with the total energy consumed during application execution.

The mapping obtains the following set of pairs associating module to tiles: $\{(PA1, \tau_{[0,0]}), (ME, \tau_{[0,1]}), (PA0, \tau_{[0,2]}), (PA3, \tau_{[1,0]}), (PA2, \tau_{[1,1]}), (PC, \tau_{[1,2]})\}$. To each link in the communication architecture the mapping associates an estimation of dynamic energy consumption. Values are in nJ. The mapping interface also furnishes a global estimation for dynamic and static energy consumption (in mJ).

Once a mapping is found, the designer may estimate the application execution time (the texec value discussed in Section 5.4) by analyzing the latency report supplied at the end of the mapping task. Fig. 12 shows the latency report obtained for the object recognition application with the mapping of Fig. 11. Data on Fig. 12 assumes that routers operate instantaneously (number of cycles for routing $= 0$, see Fig. 4) and that processing starts injecting the first flit of messages in the NoC at the 0th clock cycle. Since the chosen NoC operates using wormhole routing, an n-flit message takes ($n +$ number of hops from source to target) cycles to traverse the NoC, including the local, router to PE links.

The report depicts data for each message, according to the computation and communication time, in clock cycles. The amount of computation clock cycles is furnished by the designer using the t parameter at each CDCG vertex. The communication clock cycles depend on the number of phits of each CDCG vertex and the number of cycles needed by the router to transmit each phit through a link. All these parameters are supplied in the *NoC Timing Parameters* field of the *Communication Infrastructure* interface as Fig. 4 depicts.

CAFES enables the simultaneous description of multiple independent applications operating on a same communication architecture, since all (mapping) tools support operation with disconnected graphs. In this case, each graph represents an application behavior and the set of all graphs represents a system running more than one application at a time. This feature may be used to search the optimum mapping of multiple applications that share the same tiles of a communication architecture. The operation of applications may either occur at different time periods, or they may concur for the same tile, i.e. operate partially or totally at the same time. This is the case of some embedded systems such as cellular phones containing more than one application running at the same time.

To exemplify multiple application descriptions, Fig. 13 presents four applications modeled with CWM. This case study assumes the simultaneous operation of all modules of the four applications in the same target architecture. As a result, the mapping algorithm tries to find groups of modules that minimize total energy consumption for each application.

Message		Clock cycles		
ID	Source -> Target	Computation	Start	End
0	ME -> PA0	0	0	76803
1	ME -> PA1	0	0	76803
2	ME -> PA2	0	0	76805
3	ME -> PA3	0	0	76803
4	PA1 -> PA0	2550	79353	79598
6	PA3 -> PA2	2550	79353	79596
5	PA2 -> PA0	2550	79355	79918
10	PA2 -> PC	375	80049	80182
8	PA0 -> PC	375	80293	80428
7	PA3 -> PA1	754	80346	80671
11	PA3 -> PC	375	81040	81171
9	PA1 -> PC	375	81046	81299
12	PC -> PA0	1286	82585	82720
13	PC -> PA1	1286	82585	82716
14	PC -> PA2	1286	82585	82718
15	PC -> PA3	1286	82585	82716
17	PA1 -> ME	375	83091	159894
19	PA3 -> ME	375	83091	236694
18	PA2 -> ME	375	83093	313494
16	PA0 -> ME	375	83095	390294

Total execution time: 390294 clock cycles

Fig. 12. Latency report of the object recognition application of Fig. 10, considering the mapping achieved using CAFES (depicted in Fig. 11). Each line represents a message containing an identification number (ID), source and target modules, the computation time which precedes the message dispatch and the instants when the message starts and ends its transmission. The difference from *Start* to *End* columns represents the message latency in clock cycles.

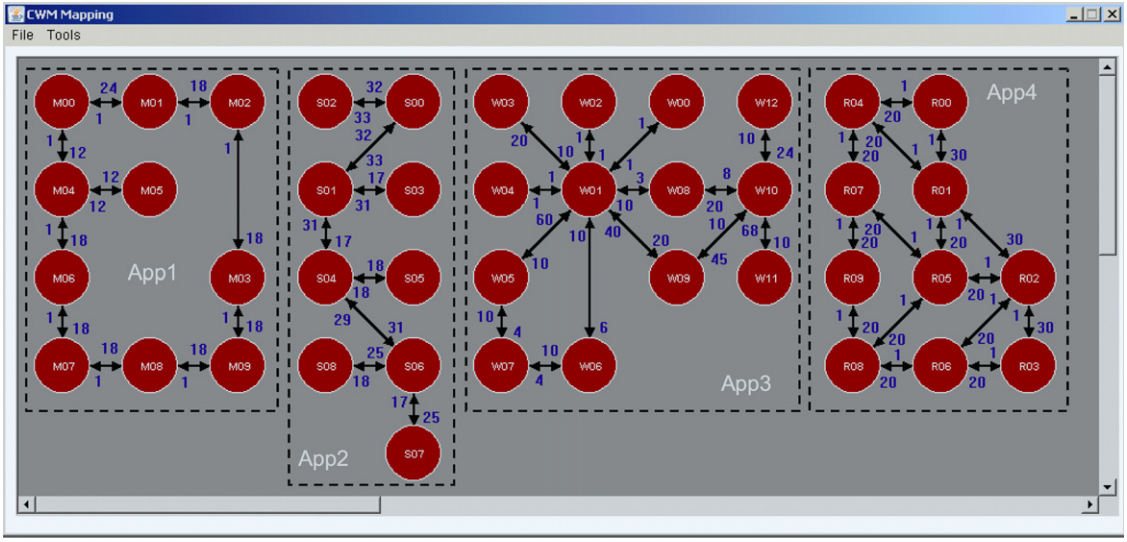


Fig. 13. A system composed by four applications, described using CWM.

S07	S03	S01	W12	W03	M08	M09
S06	S04	S00	W04	W00	M04	M03
S08	S05	S02	W09	W11	M05	M02
R00	R04	W02	W10	W08	M01	R03
R05	R01	W06	W01	M07	M00	R07
R08	R02	W07	W05	W02	M06	R09

Fig. 14. A mapping for the system composed by 4 simultaneously running applications, as described in Fig. 13, onto a 6 × 7 bidirectional 2D torus NoC.

Fig. 14 shows a mapping achieved by CAFES, considering a 6 × 7 bidirectional 2D torus NoC. In the Figure, each rectangle represents a tile of the target architecture, containing each an application module. Modules of the same application start with the same letter.

6.2. CAFES algorithms library

CAFES has a set of libraries that provides algorithms applicable to the evaluation and synthesis of intrachip communication architectures. Examples of these are *simulated annealing* (SA) and *tabu search* (TS) and some specific algorithms related to the communication architecture or the desired design task, like the heuristic algorithms *largest communication first* (LCF) and *greedy incremental* (GI) [22]. The designer can use the desired resource library to help him in the design task in view, or even to evaluate the quality of each algorithm to achieve the desired task. For instance, Fig. 15 illustrates partial results of [22], where different algorithms, implemented inside the framework, are explored to evaluate the quality of synthetic and real applications mapping onto a large quantity of NoC sizes against the computational complexity, in terms of memory usage and computation time.

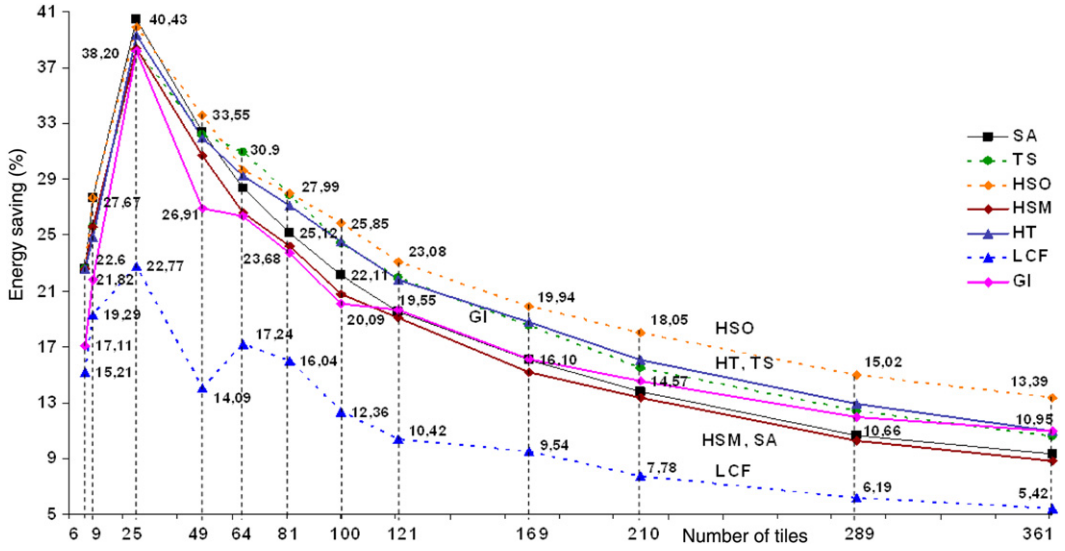


Fig. 15. Percentage of energy saving for mappings achieved by seven algorithms, considering a large variety of applications and NoC sizes. Figure extracted from [22].

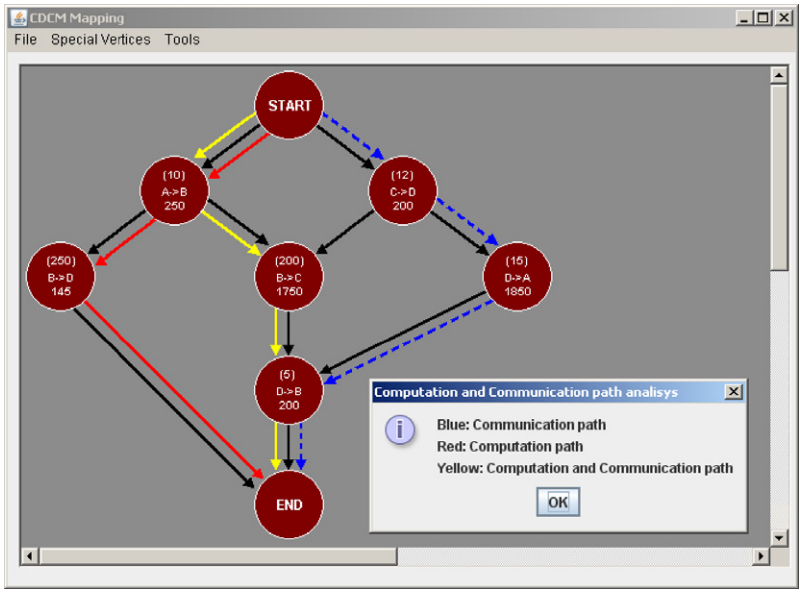


Fig. 16. Example results of using the computation and communication path analysis tool applied to a CDCM synthetic application[1].

6.3. High-level estimations

Depending only on the employed application description model, CAFES allows to estimate some communication and/or computation bottlenecks in early design stages, as depicted in Fig. 16.

Starting from the application graphs, CAFES furnishes critical path estimations for communication and computation. These are high-level estimations, independent of the target architecture. This independence derives from the fact that the high abstraction level overlooks module placement. Consequently, the communication path is underestimated, because it may change, depending on the distance between communicating modules after mapping. On the other hand, the computation path does not depend on module placement, only on the sum of all computation figures of component vertices. Therefore, high-level estimations for computation paths are more accurate. Fig. 16 shows a synthetic application described with CDCM, after applying the computation and communication path analysis tool of CAFES.

Inside Fig. 16, black arrows represent communication dependence, while blue, red and yellow arrows are communication, computation and overall critical paths, respectively.[1]

6.4. Graph generation tools to automate synthetic application building

Several test scenarios are usually necessary to validate system implementations. In many cases, developing a sufficiently large number of real applications to test the system may be unfeasible or too costly as a first approach. Thus, the availability of procedures to generate synthetic applications is useful, if not mandatory. Several works like [19–21,15,18,30,22] have benefited from the use of

[1] For readers with the black and white versions of the paper, blue arrows are dotted, red corresponds to dark grey arrows and yellow corresponds to white arrows.

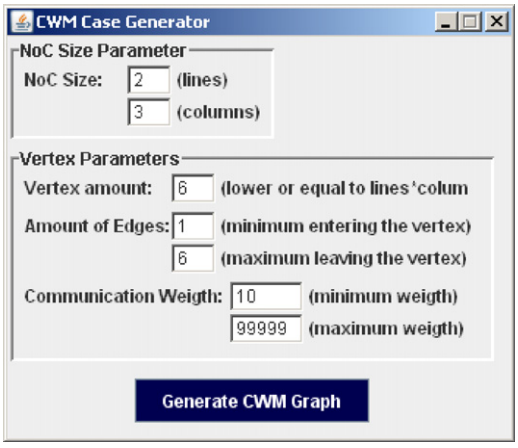


Fig. 17. Graphic interface providing access to the CAFES tool for automatic application graph generation.

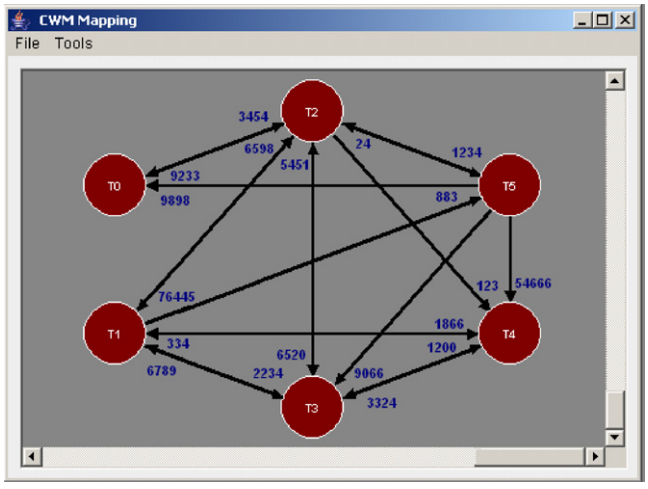


Fig. 18. CWG containing the automatically generated synthetic application described in Fig. 17.

tools for automatic application graph generation. Accordingly, the CAFES framework has available a set of graph generation tools to automate the process of building synthetic applications. CAFES can produce application graphs specific for a selected application model. The tools are similar to that proposed in [5], where it is possible to parameterize several application aspects. Application graph generation tools in CAFES have distinct interfaces for each application model. As an example, Fig. 17 illustrates the use of the tool for automatic generation of applications graphs modeled using CWG.

Here, it is possible to parameterize the number of vertices and edges, and the communication volume of CWG vertices. This tool also allows parameterizing the number of lines and columns of the communication architecture, enabling to evaluate NoCs with different dimensions and occupation rates.

Fig. 18 shows a randomly generated graph that respects the data distribution constraints described in Fig. 17.

6.5. Automatic traffic generation tool

NoCs generated by the ATLAS framework [24] are usable as communication architectures in CAFES. Designers select a NoC in the *Target Communication Architecture* field (see Fig. 3). To estimate execution time and energy by simulation, it is necessary to stimulate the NoC inputs with application data traffic. This may be achieved by producing a NoC testbench from the application graph, using a straightforward translation of the graph into a VHDL or SystemC behavioral description. CAFES can automatically generate VHDL testbenches with an internal tool whose output depends on the application model as well as on the target communication architecture.

To exemplify the application graph translation into a behavioral VHDL description, refer to the CDCG description in Fig. 10. Fig. 19 depicts part of the VHDL obtained after feeding this CDCG as input to the CAFES *automatic traffic generation tool*. Each VHDL process emulates the communication and computation of an application module placed inside a tile according to the mapping of Fig. 11. All computations and communications wait until their respective dependences are solved. Next, the computation and communication occur and the corresponding dependence flag signal is set to true, enabling all communications depending on it. For instance, module PA1, which is placed into tile with coordinates [0, 2] ($\tau_{[0,2]}$), remains waiting for the sent_ME_PA1_1 flag be true, then executes the computation time (25 ns), sends 240 phits to the module placed in tile $\tau_{[0,0]}$ (PA0) and notifies all other modules that this communication occurred, by setting the sent_PA1_PA0_1 flag true.

```
. . .
PA1_02: process -- Module PA1 mapped in tile R[0,2]
begin
  loop
    wait until sent_ME_PA1_1 = true;
    Computation(Time := 25ns);
    Communication(TargetLine := 0, TargetColumn := 0, NumberOfPhits := 240);
    sent_PA1_PA0_4 <= true;

    wait until sent_PA1_PA0_4 = true and sent_PA3_PA1_7 = true;
    Computation(Time := 5ns);
    Communication(TargetLine := 1, TargetColumn := 2, NumberOfPhits := 128);
    sent_PA1_PC_9 <= true;

    wait until sent_PC_PA1_13 = true;
    Computation(Time := 5ns);
    Communication(TargetLine := 0, TargetColumn := 1, NumberOfPhits := 76800);
    sent_PA1_ME_17 <= true;
  end loop;
end process;
. . .
```

Fig. 19. Partial VHDL description automatically generated from the CDCG in Fig. 10. It emulates the behavior of tile $\tau_{[0,2]}$ in the mapping of the object recognition application, depicted in Fig. 11.

7. Conclusions and ongoing work

This work described CAFES, a framework for application modeling and design of infrastructure communication architectures. It is an extensible open-source framework that integrates models, tools and a synthesis flow for system design, having NoCs as target architectures. Several models used to describe applications and target architectures are built inside the framework. Using these models, several tools as traffic generators, mappers and estimators are implemented. The extensibility of the framework allows inserting new application and NoC models, tools and other facilities, enabling to fulfill the design of a diversity of NoC-based systems existing today.

Results achieved attest that using the framework facilities a designer may significantly reduce the energy consumption and latency of the target application with an acceptable design time. CAFES enables to estimate the energy consumption and latency at high and low abstraction levels, depending on the desired accuracy and available design time. The available description models allow capturing various applications features, and an inter-module conversion tool enables verifying the capability of each application description model. Also, it permits assessing the precision of results achieved with each model.

The implementation of the partitioning and dynamic mapping synthesis tasks on CAFES is an ongoing work. Also, a more thorough integration of CAFES with synthesis tools for communication infrastructure generation is underway. The inclusion of support to heterogeneous MPSoC designs is another direction where work in CAFES occurs now. This implies adding more input information to the framework, such as the set of processing elements and their individual features. Also, it would be interesting to inform the clock frequency associated to each tile or to each router and PE forming a tile. With this last type of information it is possible to support heterogeneous GALS MPSoCs, a current limitation of CAFES.

Acknowledgment

The authors acknowledge the support of the Conselho Nacional de Desenvolvimento Cientfico e Tecnolgico (CNPq-Brazil) through research grants 308924/2008-8, 141247/2005-3, 301599/2009-2, 309255/2008-2, 306178/2009-5, 485315/2007-6 and 312485/2009-3.

References

[1] A. Aswatha, T. Basavaraju, A. Kalpana, Efficient power modeling for on-chip global interconnects, in: 51st Midwest Symposium on Circuits and Systems, MWSCAS, August 2008, pp. 458–461.
[2] H. Blume, J.V. Livonius, L. Rotenberg, T.G. Noll, H. Bothe, J. Brakensiek, Performance and power analysis of parallelized implementations on an MPCore multiprocessor platform, in: International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation, IC-SAMOS, July 2007, pp. 74–81.
[3] D. Burger, J.R. Goodman, Billion-transistor architectures, in: Guest Editors' Introduction, IEEE Computer 30 (9) (1997) 46–48.
[4] W. Dally, B. Towles, Route packets, not wires: on-chip interconnection networks, in: Design Automation Conference, DAC, June 2001, pp. 684–689.
[5] R. Dick, D. Rhodes, W. Wolf, TGFF: task graphs for free, in: International Workshop on Hardware/Software Codesign, CODES/CASHE, March 1998, pp. 97–101.
[6] E. Dolif, M. Lombardi, M. Ruggiero, M. Milano, L. Benini, Communication-aware stochastic allocation and scheduling framework for conditional task graphs in multi-processor systems-on-chip, in: Seventh ACM & IEEE International Conference on Embedded Software, EMSOFT, October 2007, pp. 47–56.
[7] S. Evain, J.P. Diguet, D. Houzet, μspider: a CAD tool for efficient NoC design, in: 22nd Norchip Conference, November 2004, pp. 218–221.
[8] M. Ghadiry, M. Nadi, D. Rahmati, New approach to calculate energy on NoC, in: International Conference on Computer and Communication Engineering, ICCCE, May 2008, pp. 1098–1104.
[9] F. Glover, M. Laguna, Tabu Search, Kluwer Academic Publishers, 1997, 412 pp.
[10] C. Hsieh, M. Pedram, Architectural energy optimization by bus splitting, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 21 (4) (2002).
[11] J. Hu, R. Marculescu, Energy-aware mapping for tile-based NoC architectures under performance constraints, in: Asia and South Pacific Design Automation Conference, ASPDAC, January 2003, pp. 233–239.
[12] J. Hu, R. Marculescu, Energy-aware communication and task scheduling for network-on-chip architectures under real-time constraints, Design, Automation, and Test in Europe (DATE) (2004) 234–239.
[13] ITRS International Technology Roadmap for Semiconductors, 2008, Update—overview. Available at: http://www.itrs.net/Links/2008ITRS/Update/2008_Update.pdf (captured in October 2009, 2008).
[14] A. Jalabert, et al., XpipesCompiler: a tool for instantiating application specific networks on chip, Design, Automation, and Test in Europe, DATE (2004) 884–889.
[15] S. Johann Filho, A. Aguiar, C. Marcon, F. Hessel, High-level estimation of execution time and energy consumption for fast homogeneous MPSoCs prototyping, in: IEEE/IFIP International Workshop on Rapid System Prototyping, RSP, June 2008, pp. 27–33.
[16] Kurt Keutzer, S. Malik, A.R. Newton, J.M. Rabaey, A. Sangiovanni-Vincentelli, System level design: orthogonalization of concerns and platform-based design, IEEE Transactions on Computer-Aided Design of Circuits and Systems 19 (12) (2000).
[17] Y. Krasteva, F. Criado, E. de la Torre, T. Riesgo, A fast emulation-based NoC prototyping framework, in: International Conference on Reconfigurable Computing and FPGAs, ReConFig, December 2008, pp. 211–216.
[18] M. Kreutz, C. Marcon, N. Calazans, A. Susin, Energy and latency evaluation of NoC topologies, in: IEEE International Symposium on Circuits and Systems, ISCAS, May 2005, pp. 5866–5869.
[19] C. Marcon, A. Borin, A. Susin, L. Carro, F. Wagner, Time and energy efficient mapping of embedded applications onto NoCs, in: Asia and South Pacific Design Automation Conference, ASPDAC, January 2005, pp. 33–38.
[20] C. Marcon, N. Calazans, F. Moraes, A. Susin, I. Reis, F. Hessel, Exploring NoC mapping strategies: an energy and timing aware technique, Design, Automation, and Test in Europe (DATE) (2005) 502–507.
[21] C. Marcon, M. Kreutz, A. Susin, N. Calazans, Models for embedded application mapping onto NoCs: timing analysis, in: IEEE/IFIP International Workshop on Rapid System Prototyping, RSP, June 2005, pp. 17–23.
[22] C. Marcon, E. Moreno, N. Calazans, F. Moraes, Comparison of NoC mapping algorithms targeting low energy consumption, IET Computers & Digital Techniques 2 (6) (2008) 471–482.
[23] C. Marcon, J. Palma, N. Calazans, F. Moraes, A. Susin, R. Reis, Modeling the traffic effect for the application cores mapping problem onto NoCs, in: VLSI-SoC: From Systems to Silicon, vol. 240, Springer, 2007, pp. 179–194 (Chapter 12).
[24] F. Moraes, N. Calazans, A. Mello, L. Möller, E. Moreno, ATLAS—an environment for NoC generation and evaluation. Available online at: http://www.inf.pucrs.br/~gaph/ATLASHtml/ATLASIndex_us.html (captured on April.22.09).
[25] F. Moraes, N. Calazans, A. Mello, L. Möller, L. Ost, HERMES: an infrastructure for low area overhead packet-switching networks on chip, Integration, the VLSI Journal 38 (1) (2004) 69–93.
[26] S. Murali, G. De Micheli, SUNMAP: a tool for automatic topology selection and generation for NoCs, in: Design Automation Conference, DAC, 2004, pp. 914–914.
[27] S. Murali, G. De Micheli, Bandwidth-constrained mapping of cores onto NoC architectures, Design, Automation, and Test in Europe (DATE) (2004) 896–901.
[28] L. Ost, A. Mello, J. Palma, F. Moraes, N. Calazans, MAIA—a framework for networks on chip generation and verification, in: Asia and South Pacific Design Automation Conference, ASPDAC, June 2005, pp. 18–21.
[29] G. Palermo, C. Silvano, PIRATE: a framework for power/performance exploration of network-on-chip architectures, Lecture Notes in Computer Science (2004) 521–531.
[30] J. Palma, L. Indrusiak, F. Moraes, A. Ortiz, M. Glesner, R. Reis, Inserting data encoding techniques into NoC-based systems, in: IEEE Computer Society Annual Symposium on VLSI, May 2007, pp. 299–304.
[31] S. Samii, et al., Cycle-accurate test power modeling and its application to SoC test architecture design and scheduling, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 27 (5) (2008) 973–977.
[32] B. Talwar, et al., Latency, power and performance trade-offs in network-on-chips by link microarchitecture exploration, in: International Conference on VLSI Design, January 2009, pp. 163–168.
[33] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, S. Borkar, An 80-tile sub-100-W teraFLOPS processor in 65-nm CMOS, IEEE Journal of Solid-State Circuits 43 (1) (2008) 29–41.
[34] P. van Laarhoven, E. Aarts, Simulated Annealing: Theory and Applications, Kluwer Academic Publishers, 1987, 204 pp.

[35] T. Ye, L. Benini, G. De Micheli, Analysis of power consumption on switch fabrics in network routers, in: Design Automation Conference, June 2002, pp. 524–529.

César Augusto Missio Marcon received the bachelor's degree from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in Electrical Engineering in 1989, the M.Sc. degree in Computer Science in 1992, also from UFRGS, and the Ph.D. degree in Computer Science in 2005, from the same University. He is currently an Assistant Professor at the Catholic University of Rio Grande do Sul (PUCRS). His research interests include intrachip communication networks, embedded system design and implementation, and computer-aided design techniques and tools.

Ney Laert Vilar Calazans received the bachelor's degree from the Federal University of Rio Grande do Sul (UFRGS), Brazil, in Electrical Engineering in 1985, the M.Sc. degree in Computer Science in 1988, also from UFRGS, and the Ph.D. degree in Microelectronics in 1993, from the Université Catholique de Louvain (UCL), Belgium. He is currently a Professor at the Catholic University of Rio Grande do Sul (PUCRS). His research interests include intrachip communication networks, non-synchronous circuit design and implementation, and computer-aided design techniques and tools. Professor Calazans is a member of the Brazilian Computer Society, SBC.

Edson Ifarraguirre Moreno was born in Porto Alegre, Brazil, in 1976. He received the bachelor's degree in Computer Science from the Catholic University of Rio Grande do Sul (PUCRS) in 2001. He is currently a Ph.D. candidate at the same university. His doctoral research focuses on networks on chip (NoCs) and multiprocessor systems on chip (MPSoCs) design. His research interests also include high level hardware and embedded systems modeling, computer aided design techniques and tools and multicore programmability.

Fernando Gehm Moraes received the Electrical Engineering and M.Sc. degrees from the Universidade Federal do Rio Grande do Sul (UFRGS), Porto Alegre, Brazil, in 1987 and 1990, respectively. In 1994 he received the Ph.D. degree from the Laboratoire d'Informatique, Robotique et Microélectronique de Montpellier (LIRMM), France. He is currently a Professor at the Catholic University of Rio Grande do Sul (PUCRS). He has authored and co-authored 12 peer reviewed journal articles in the field of VLSI design, comprising the development of networks on chip and telecommunication circuits. He has also authored and co-authored more than 140 conference papers on these topics. His research interests include intrachip communication networks (NoCs), and MPSoC design. Professor Moraes is a member of the IEEE and of the Brazilian Computer Society, SBC.

Fabiano Passuelo Hessel is an Associate Professor of Computer Science at Pontifical Catholic University of Rio Grande do Sul (PUCRS), Brazil. He received his Ph.D. in Computer Science from Université Joseph–Fourier, TIMA laboratory, France. He is the head of the Embedded Systems research group. He was the Associate Editor of the ACM Transactions on Embedded Computer Systems—Special Issue on Rapid System Prototyping. He has several publications in prestigious conferences and journals, book chapters and books. His research interests are embedded real-time systems, real-time operating systems and MPSoC systems.

Altamiro Amadeu Susin received the bachelor's degree in Electrical Engineering and M.Sc. degrees from Universidade Federal do Rio Grande do Sul (UFRGS), Brazil, in 1972 and 1977, respectively. Since 1968 he worked in the Data Centers of two local Universities. In 1981 he got his Dr. Ing. degree from the Institut National Polytechnique de Grenoble, France. He is presently a professor at the Electrical Engineering Department of UFRGS, in charge of Digital Systems Design disciplines at the graduate and undergraduate levels. He is also a member of the Computer Science Graduate Program of UFRGS, where he is responsible for VLSI Architecture courses. His main research interests are Integrated Circuits Architecture, Embedded Systems, Signal Processing. Prof. Susin has published more than one hundred technical papers in these domains. He has been responsible for several R&D projects funded by public agencies and/or industries.