

Gaspard2: from MARTE to SystemC Simulation

Éric Piel

e.a.b.piel@tudelft.nl

Rabie Ben Atitallah

benatita@lifl.fr

Philippe Marquet

philippe.marquet@lifl.fr

Samy Meftali

samy.meftali@lifl.fr

Smaïl Niar

smaïl.niar@lifl.fr

Anne Etien

anne.etien@lifl.fr

Jean-Luc Dekeyser

jean-luc.dekeyser@lifl.fr

Pierre Boulet

pierre.boulet@lifl.fr

LIFL

UMR 8022 CNRS – USTL
59655 Villeneuve d’Ascq

INRIA Lille – Nord Europe

40 Avenue Halley
59650 Villeneuve d’Ascq

LAMIH

UMR 8530 CNRS – UVHC
59313 Valenciennes

Abstract

To efficiently use the tremendous amount of hardware resources available in next generation MultiProcessor Systems-on-Chip (MPSoC), new design methodologies and tools are needed to reduce the development complexity of such systems. In this paper, we present an efficient MPSoC design environment, Gaspard2. It uses the new standard MARTE (Modeling and Analysis of Real-Time and Embedded systems) profile for high level system representation. Gaspard2 is based on a Model-Driven Engineering (MDE) methodology allowing to move from the primary modeling form to an executable platform. In our work, we target SystemC code generation at the Timed Programmer View (PVT) level. To reach our objective, a compilation chain has been developed, made up of several model to model transformations.

Keywords: MPSoC, MARTE, Gaspard2, MDE, transformation, UML, SystemC

1. Introduction

At the same time as advances in technology allow to integrate more and more transistors on a single chip, the embedded system applications get always more sophisticated. Although these two evolutions fit well in term of computation power, the combination put a strong pressure on the designers’ capacity to design and verify the resulting very complex systems. To enhance the productivity, new design methodologies have to be adopted for the development of such large and complex Systems-on-Chip.

Nowadays, system design based on a high level modeling approach becomes increasingly attractive for different reasons. First, designers may attain higher levels of abstrac-

tion allowing to focus on a particular domain space. Second, such approach offers a visual expression form making easier to understand the system and to improve the communication between designers. The concepts and their relationship through which a model of the system is expressed are precisely specified in a metamodel. However, to get a graphical representation of these models, two alternatives are possible: define a graphical view for each concept of the metamodel or extend UML in a profile to support these concepts. The former solution is relatively tedious due to the need for a dedicated tool. The second solution aims to add semantics to UML [8] in order to adapt it to a specific domain and to use traditional modeling tools such as MagicDraw, Papyrus or RSA. MARTE (Modeling and Analysis of Real-Time and Embedded systems) [15] is a standard proposal of the Object Management Group (OMG). The primary aim of MARTE is to add capabilities to UML for model-driven development of real-time and embedded systems. UML provides the framework into which needed concepts are plugged. The MARTE profile enhances possibility to model software, hardware and relations between them. It also provides extensions to make performance and scheduling analysis and to take into account platform services (such as services provided by an operating system).

In this paper, we present our Gaspard2 design environment dedicated to MPSoC allowing to move from the high level MARTE description to an executable platform. Gaspard2 is based on Model-Driven Engineering [13] (MDE). This methodology revolves around two concepts: model and transformation. Data and their structures are represented in models, while the computation is done by transformations. In our framework, models are used to represent the system (application, architecture, and allocation). Transformations are employed to move from an abstract and contemplative model to a detailed and productive model. The set of transformations forms the compilation chain. In our

case, this chain converts the platform-independent MPSoC model into a SystemC [11] code.

The rest of this paper is organized as follows. An overview of related work on the modeling of SoC and automatic generation of simulation out of it is provided in section 2. Section 3 introduces the MARTE profile for MPSoC modeling and an overview of the extensions needed to target a low level platform. The SystemC platform that we are targeting is presented in details in section 4. Section 5 introduces the Gaspard2 environment for MPSoC design based on a Model-Driven Engineering methodology. This paper finishes with a case study illustrating the use of the models and the transformation in section 6 followed by a conclusion.

2. Related Work

Using UML standard for system design started in 1999 [6]. At the beginning, this approach did not achieve a great success due to the limited supporting tools and capabilities to target executable platform. As an example, it is possible to quote the UML profile for SystemC based on UML 1.4 [4] dedicated to SoC modeling. In an attempt to resolve this issue, the Object Management Group (OMG) devoted significant efforts which yielded the MDE initiative. This methodology provides a set of guidelines for translating from platform-independent models to platform-specific models.

Several propositions based on UML were later made. Early propositions have focused on the *model* side of this approach. We can mention SysML [10], from which MARTE has taken a certain number of concepts. It has the disadvantage of being too much generic, this means that there exist too many variation points to allow a complete specification of a system only at the model level. In the opposite, the standardization proposal by Fujitsu [9] about a UML profile for SoC tended to be too close from the implementation, providing concepts very tied to SystemC. None of those works have investigated how to reach the implementation from the specification.

More recently, there have been works also concerned by obtaining a realization out from the modeling of the SoCs, proposing *model transformations*. They highlighted the need for model notation to have an entirely *executable* semantics [7]. That is: not having any semantic variation points and containing information so that each part of the system can be completely realized. Unfortunately, so far the propositions [7, 1, 14] have been limited as direct transformations from UML profiles to SystemC simulations, taking little out of the powerful MDE approach.

In our project called Gaspard2 [16], we went further in the usage of MDE by abstracting more the level of specification of the SoC and leveraging the model transformations to target the multiple types of outputs that are needed during the SoC design.

3. The MARTE profile for MPSoC modeling

Before elaborating further on the generative side of our tool, permitting the generation of the entire simulation of the system, we will present what can be considered by the SoC designer as the *interface*. This interface is composed of the concepts that the designer has to grasp and manipulate to express the complete SoC. The major part of the interface is based on the MARTE profile. In the first part of this section we will articulate the advantages of MARTE for this particular domain of application, and which major concepts of the profile we rely on. Later in this section we will summarize the additional mechanisms Gaspard2 uses.

3.1. The MARTE profile as a base

The modeling of SoC in Gaspard2 is nearly entirely based on the MARTE profile (and of course on UML). One of the main advantages of MARTE is that it clearly distinguishes the hardware components from the software components. This is done via stereotypes provided by the Detailed Resource Modeling (DRM) package, in particular the *HwResource* and *SwResource* stereotypes. For SoC conception this separation is of prime importance as it is usual to create those two parts of the system simultaneously, by different teams. Moreover, this separation provides a flexible way to independently change the software part or the hardware part. For instance, this allows testing the software on different kind of hardware architecture, or to reuse an architecture (with little or no change) for different applications.

Compared to UML, with MARTE the hardware can be described with much more precision. The architecture of the hardware is described in a structural way. The notion of component permits to reuse a set of basic components in various places and to define the architecture hierarchically. In Gaspard2, only the *HW_Logical* sub-package is used. It allows to describe information about the kind of available components (*HwRAM*, *HwProcessor*, *HwASIC*, *HwBus*...), their characteristics, and how they are connected to each other. In Gaspard2 no use is done of the *HW_Physical* sub-package because when the system is entirely on one chip, synthesis tools can compute automatically an optimized layout.

Gaspard2 also leverages from the MARTE profile the *Alloc* package. As hardware and software are modeled independently, concepts to express how to link them together are necessary. The *Alloc* package provides the required concepts to associate a task or a data onto a processor or a memory. For now only the allocation of nature *spacialDistribution* is handled by Gaspard2. This is so because the semantics of the application, which we will describe later in this section, is simple enough so that a good scheduling is always possible to determine statically (at compilation time). In future versions of Gaspard2, the nature *timeScheduling* might be taken into account. This would allow the SoC de-

signer to force a specific order of task execution or memory allocation.

The last major part of MARTE on which Gaspard2 relies is the Repetitive Structure Modeling (RSM) annex. This package defines stereotypes and notations to describe in a compact way the regularity of a system's structure or topology. The structures considered are composed of repetitions of structural elements interconnected via a regular connection pattern. It provides the designer a way to express models efficiently and explicitly with a high number of identical components. In Gaspard2, the concepts brought by this package allow both to model concisely large regular hardware architectures, such as the more and more frequent multi-processor architectures, and the parallel applications. A detailed description about its usage and about using it with the allocation package can be found in [3].

The lower part of Figure 4 shows the model of the main component of a hardware architecture. It is constituted of 4 processing units (using the repetition concept), a RAM, a ROM, and a crossbar to interconnect those components together. The MIPS-PU component is not stereotyped *Hw-Processor* because this component is composed of both a processor and a cache. The *Reshape* connectors permit to specify which component is connected to which repetition of the *slave* port of the crossbar.

The other packages of the MARTE profile are not taken into account in the transformation chains of Gaspard2. The main reason is that our tool focuses only on one kind of systems, signal intensive processing applications on MPSoC, within the broad domain of systems that MARTE encompasses. The SoC designer can still use those concepts them in his or her model in order to explicitly describe or specify the system: they will simply be ignored by the transformations.

3.2. Introduced extensions

In order to generate a complete implementation of the model, a completely defined semantics and very precise details on the elementary component are necessary. As MARTE strives to be generic, for its best, so that it can accommodate a large set of needs, some of the needed information required for the implementation is not expressible only with the concepts provided. Therefore Gaspard2 introduces additional concepts and semantics to fill up this need in the particular domain of SoC modeling. This is done via a specific UML profile. This is a very good demonstration of the advantage of the profiling mechanism of UML: it allows to use standard concepts understood by a large set of people to model most of the system, and to introduce additional, very precise, concepts for the specific domain that a particular tool focuses on. We are presenting here the major two extensions introduced for the need of MPSoC automatic generation.

The first addition to MARTE concerns the semantics of the application modeling. In MARTE, as in UML, mostly

any kind of application can be specified but the behaviour of the application cannot be entirely defined, it is up to the programmers to code the precise behaviour of each component. In Gaspard2 we have restrained the domain of application which can be modeled to data-flow systematic applications, which is one of the major domains of application encountered in the field of MPSoC. So, in addition to the MARTE stereotypes *SwResource* (for the application components) and *FlowPort* (for all the ports), the semantics introduces a particular Model of Computation (MoC) based on ArrayOL [2]. This language permits to represent the application in a declarative way, well fitted to the graphical representation of UML, and is able to express all the data and task parallelism of an application.

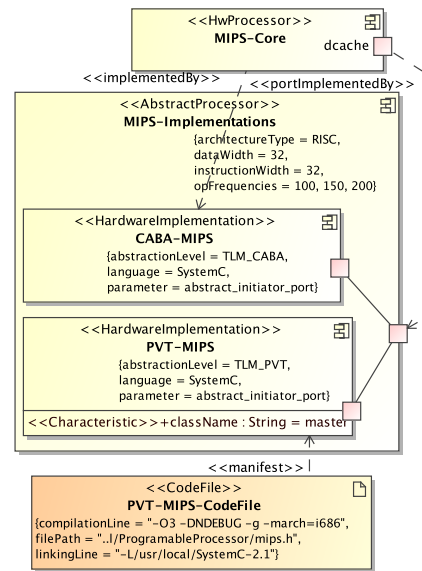


Figure 1. Deployment of a MIPS processor.

The second extension of Gaspard2 over the MARTE profile concerns the description of the elementary components: the components which are the basic blocks of all the other components. To transform the high abstraction level models into simulation code, very detailed deployment information must be provided. In particular, every elementary component must be linked to an existing code, for both the hardware and the application. A specificity of the SoC design is that one functionality can be implemented in different ways. This is necessary for testing the system with different tools, or at different levels of abstraction, along the design phase. Moreover, this is also necessary to facilitate application component reuse on different hardwares: for instance an application component can be optimized for a specific processor, or written as a hardware accelerator. Therefore we have introduced the concept of *AbstractImplementation* which expresses one hardware or software functionality, independently of the compilation target. It contains one or several *Implementations*, each one being used to define a specific implementation at a given simulation level and programming language. Figure 1 shows an example of an *Ab-*

stractImplementation of a MIPS processor which contains two *Implementations* at the CABA and PVT levels written in SystemC. The *Implementation* which fits best the target will be used to reify the component during the compilation phase. This automatic selection allows to generate the exact same SoC model at different simulation levels. Additionally, the concept of *CodeFile* is used to specify the code and compilation options required.

4. SystemC target platform

In our work, we focus on the use of Transaction Level Modeling (TLM) as a viable approach for accelerating MP-SoC simulation. In TLM, a set of abstraction levels simplifying the description of inter-module communication is defined. The simulation time is reduced by augmenting communication operation granularity. Consequently, signal handshaking, as used in cycle accurate level, is replaced in TLM by transactions using objects (i.e. words or frames) over communication channels.

In fact, TLM refers to a taxonomy of several abstraction levels. Each level differs from the others in the degree of functional or temporal details it expresses. In order to allow reliable design exploration, the simulation generated by our framework is at the so called *Programmer's View* (PV) level. In the PV level the hardware architecture is specified for both processing and communication parts, and some arbitration of the communication infrastructure is applied. At this level, application tasks are compiled for the host machine to benefit from high simulation speedup (avoiding the instruction interpreter). For performance estimation, this level is annotated with timing specification (PVT).

5. Gaspard2 environment

The Gaspard2 environment uses model transformations in order to implement the MPSoC compilation flow. Transformations can be chained: the output model of a transformation is used as the input of an other one. They are employed to move from an abstract model to a more detailed model. Those intermediary models, respecting a pre-defined metamodel, provide strongly documented “synchronization points” in the compilation flow. Consequently, each transformation is independent from the others (they only depend on the pre-defined metamodel). This organization facilitates the reuse of transformations while compiling towards different simulation levels. Moreover the compilation flow must adapt easily to the evolutions of the fast evolving SoC domain. Being able to write transformations with a declarative language increases their maintainability and simplifies their modification. The declarative approach specifies that each transformation is separated into a set of rules. Each rule is associated to an explicit set of input and output patterns. In Gaspard2, the engine we used to execute

the transformations is a framework which permits to have a declarative structure using Java.

Our Gaspard2 environment, thanks to the MDE approach, provides great flexibility on the compilation chain. Thus, designers can couple additional tools, or target different platforms. For instance we have also carried out the generation of VHDL implementation on FPGA and of synchronous language code from the same chain (Fig. 2).

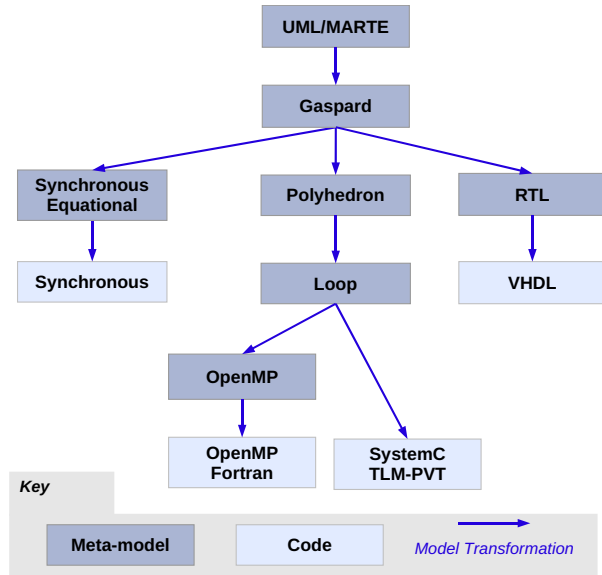


Figure 2. The Gaspard2 transformation chain.

An overview of our compilation chain is available in Fig. 2. The *UML/MARTE* model, which represents the high level MPSoC model is first directly converted to the domain specific model (*Gaspard*). It is then transformed into the *Polyhedron* model, where the application is reshaped and split on the different processing units. This derived model is then transformed into the *Loop* model, where the application is represented according to a traditional nested loop schema. While those transformations are shared, the later transformations are specific to the simulation level targeted. They generate SystemC simulation code at the PVT level.

6. Use case

In order to give more concrete description on our usage of MARTE and also to validate our methodology and model transformations, we are providing here an example of MP-SoC modeling with MARTE and generation of SystemC simulation. The modeled application is an H.263 [5] video encoder. It aims at compressing a video stream originally represented as a sequence of QCIF pictures (176 × 144 pixels in YCbCr format). Due to space restriction, we can not show the entire model of the application, however interested readers can obtain a much more complete description of this use case in [12].

Figure 3 presents the model of the second and third levels of hierarchy of this application. The *H263Encoder* component reads one picture decomposed in three parts (corresponding to the three ports *lumin*, *cbin*, *crin*) and, via *Tilers*, it links the processing of block of pixels to the *H263MacroBlock* component. The result of the processing is then reassembled via other *Tilers* and provided as output. In order to process the whole video sequence the *H263Encoder* is instantiated at the first level of the application with an infinite repetition. The component *H263MacroBlock* is constituted of three sub-components pipelined to process the picture through the three stages of the H.263 encoder algorithm.

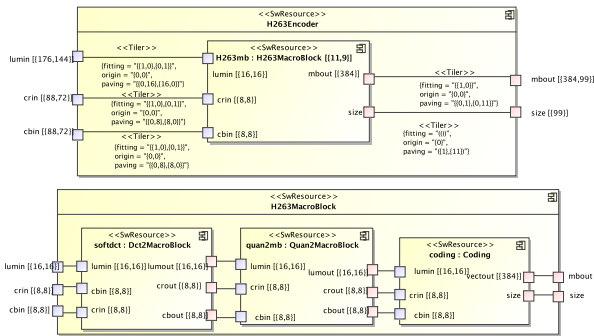


Figure 3. The main application components of the H.263 encoder.

The hardware architecture which executes the application is a SoC with 4 MIPS processors and shared memory. Part of the model of this architecture is presented in Figures 1 and 4. Using the allocation and distribution mechanisms of the MARTE profile the mapping of the application on the hardware architecture was specified. Figure 4 shows the part of the model specifying the distribution of the 99 instances of *H263MacroBlock* onto the four MIPS processors. Informally explained, the distribution linearizes the 11×9 instances and maps them on the array of processors via a modulo function. Consequently, 25 instances are assigned to each processor except the fourth one which receives only 24 instances.

Every elementary component has been deployed on an *AbstractImplementation* which contains at least one implementation in SystemC at the PVT level (for the hardware components) or one C or C++ implementation (for the software components).

Once the system has been entirely modeled, the UML file is exported and is used as input of the Gaspard2 tool chain. The user selects which compilation chain should be executed among the four available. All the transformations are automatically executed, generating the various intermediary models, until reaching the target implementation. In the screenshot on Figure 5, the left panel shows all the generated models (UML, Gaspard2, Polyhedron, Loop, and the SystemC/PVT source code). On the right panel, the UML

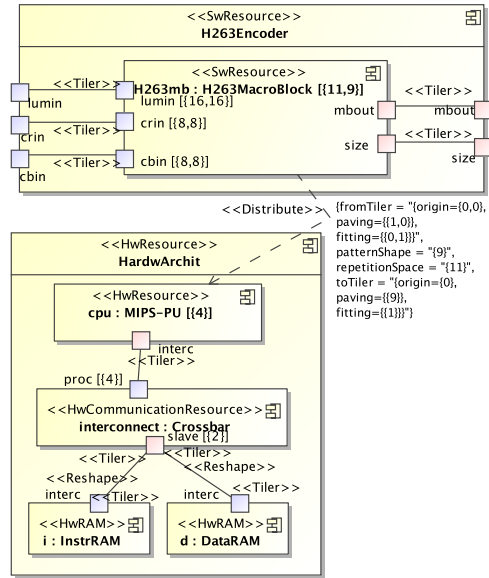


Figure 4. Allocation of the main application component on the processors.

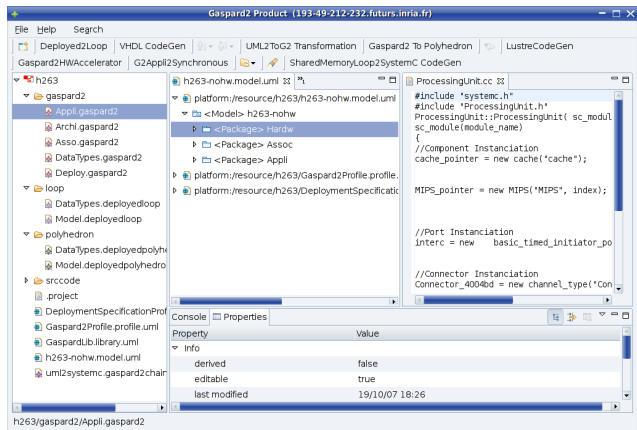


Figure 5. The Gaspard2 environment.

model is represented by an inheritance tree, and a part of the simulation source code is listed.

The complete generation of the source code, its compilation and the execution of the simulation constitute the first part of the validation of our method. As the model is expressed at a high level of abstraction, modification of the system can be very easily done. This allows the designer to test various configurations and find which one is the best. For instance, we have changed the number of processors in the system. This consists in modifying two numbers in the model: in the *shape* associated to the MIPS processor component, and in the *shape* associated to the ports of the crossbar component. Then the simulation source code is re-generated, and the simulation of the new configuration can be executed.

In order to validate the simulation, we have also written manually a simulation of the system at a lower level of ab-

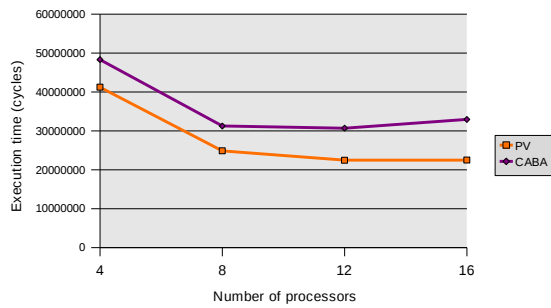


Figure 6. Number of simulated cycles depending on the amount of processors, at the PVT and CABA abstraction levels.

straction (the so called Cycle-Accurate Byte-Accurate level, CABA). Figure 6 presents the number of simulated cycles used for the encoding of one picture when the architecture has 4, 8, 12, and 16 processors for the two levels of abstraction. About 15% to 30% of under-estimation can be noted on the PVT simulation, but the relative order between the configurations is respected. In particular, both simulations show that a 16-processor architecture is slower than a 8-processor one (this is due to the conflicts for accessing the RAM). This confirms the correctness of the generated simulation.

7. Conclusions

In this paper we have presented the Gaspard2 chain generating a SystemC simulation at a high level of abstraction for an MPSoC model. In the Gaspard2 environment the modeling is based on the MARTE profile with some extensions necessary to completely specify the system and remove any semantic variation point. The compilation of the system into a simulator is done via a chain of model transformations. This work has been validated by the modeling of a H.263 encoder parallelized on a shared memory MP-SoC architecture. In particular we have highlighted the ease of exploring various configurations thanks to the high level modeling.

As perspective of this work, first of all it could be interesting to write another transformation chain, targeting a SystemC simulation at lower level of abstraction. A major part of this work should be feasible by reusing the current transformation chain. We are planning to extend the application semantics in order to be able to model a bigger domain of application. Finally, we are also considering to automatize the design-space exploration, using the result of the generated simulation to determine which part of the input model to modify, until the results are compliant with a set of requirements defined by the designer.

References

- [1] M. Alanen, J. Lilius, I. Porres, D. Truscan, I. Oliver, and K. Sandstrom. Design method support for domain specific soc design. In *MBD-MOMPES '06: Proceedings of the Fourth Workshop on Model-Based Development of Computer-Based Systems and Third International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MBD-MOMPES'06)*, pages 25–32, Washington, DC, USA, 2006. IEEE Computer Society.
- [2] P. Boulet. Array-OL revisited, multidimensional intensive signal processing specification. Research Report RR-6113, INRIA, Feb. 2007.
- [3] P. Boulet, P. Marquet, E. Piel, and J. Taillard. Repetitive Allocation Modeling with MARTE. In *Forum on specification and design languages (FDL'07)*, Barcelona, Spain, Sept. 2007. Invited Paper.
- [4] F. Bruschi and D. Sciuto. SystemC based design flow starting from uml model. In *Proceedings of ESCUG*, 2002.
- [5] G. Cote, B. Erol, M. Gallant, and F. Kossentini. H.263+: video coding at low bit rates. *IEEE Trans. On Circuits And Systems For Video Technology*, Nov. 1998.
- [6] G. Martin. UML and VCC. White paper, Cadence Design Systems, 1999.
- [7] K. D. Nguyen, Z. Sun, P. S. Thiagarajan, and W.-F. Wong. Model-driven SoC design via executable UML to SystemC. In *RTSS '04: Proceedings of the 25th IEEE International Real-Time Systems Symposium (RTSS'04)*, pages 459–468, Washington, DC, USA, 2004. IEEE Computer Society.
- [8] Object Management Group, Inc., editor. *UML 2 Superstructure (Available Specification)*. <http://www.omg.org/cgi-bin/doc?ptc/2004-10-02>, Oct. 2004.
- [9] Object Management Group, Inc., editor. *UML Extension Profile for SoC RFC*. <http://www.omg.org/cgi-bin/doc?realtime/2005-03-01>, Mar. 2005.
- [10] Object Management Group, Inc., editor. *Final Adopted OMG SysML Specification*. <http://www.omg.org/cgi-bin/doc?ptc/06-0504>, May 2006.
- [11] Open SystemC Initiative. SystemC. <http://www.systemc.org/>.
- [12] E. Piel. *Ordonnancement de systèmes parallèles temps-réel, De la modélisation à la mise en œuvre par l'ingénierie dirigée par les modèles*. Thèse de doctorat (PhD Thesis), Laboratoire d'informatique fondamentale de Lille, Université des sciences et technologies de Lille, France, Dec. 2007.
- [13] Planet MDE. Model Driven Engineering, 2007. <http://planetmde.org>.
- [14] E. Riccobene, P. Scandurra, A. Rosti, and S. Bocchio. A model-driven design environment for embedded systems. In *DAC '06: Proceedings of the 43rd annual conference on Design automation*, pages 915–918, New York, NY, USA, 2006. ACM.
- [15] L. Rioux, T. Saunier, S. Gerard, A. Radermacher, R. de Simone, T. Gautier, Y. Sorel, J. Forget, J.-L. Dekeyser, A. Cucuru, C. Dumoulin, and C. Andre. MARTE: A new profile RFP for the modeling and analysis of real-time embedded systems. In *UML-SoC'05, DAC 2005 Workshop UML for SoC Design*, Anaheim, CA, June 2005.
- [16] WEST Team LIFL, Lille, France. Graphical array specification for parallel and distributed computing (GASPARD-2). <http://www.lifl.fr/west/gaspard/>, 2005.