Modeling Heterogeneous Embedded Systems with TTool

Daniela Genius*, Marie-Minerve Louërat*, François Pêcheux*
Ludovic Apvrille†, Haralampos Stratigopoulos*
* Sorbonne Université, LIP6, CNRS UMR 7606,
Email: first_name.last_name@lip6.fr
† Télécom ParisTech, Université Paris-Saclay, Institut Mines-Telecom
Email: first_name.last_name@telecom-paristech.fr

***Abstract*—Embedded systems are increasingly heterogeneous, comprising digital and analog integrated circuits, sensors, and actuators. This paper presents a first step towards an integrated modeling and simulation tool for verification and virtual prototyping of heterogeneous embedded systems on different abstraction levels.**

I. Introduction

The complexity of recent embedded systems pushes current design techniques to their limits. In particular, the space to be explored is getting larger. Model-oriented design of complex embedded systems is nowadays a current practice concerning software development; the hardware aspects of such systems are however less frequently designed using this kind of approach.

Many applications e.g. from robotics, automotive and autonomous systems require moreover heterogeneous modeling - including modeling of analog/mixed signal (AMS) and radio frequency (RF) features.

Nevertheless, the related work in the next section demonstrates the lack of an integrated tool offering at the same time heterogeneous system modeling with automated handling of synchronization issues between MoC, cycle and bit precise simulation, validation of the analog part, as well as formal verification of application code running on general purpose CPUs for the digital part.

II. Related Work

Well established tools like *Ptolemy II* [19][20], based upon a data-flow model, address heterogeneous systems by defining several sub domains [**?**]. Although hierarchy is provided, instantiation of elements controlling the time synchronization between domains is left to the responsibility of designers.

Metropolis [8] is based on a high level model and facilitates the separation of computation from communication concerns. Heterogeneous systems are taken into consideration, but heterogeneity can only be represented using processes, mediums, quantities and constraints. Hierarchical models are not allowed: all processes should be implemented in the same hierarchical level.

Metro II [**?**] introduces hierarchy and allows *Adaptors* for data synchronization as a bridge between the semantics of components belonging to different MoCs. The model designer still has the difficult task of implementing time synchronization by means of constraints, assertions, annotators and schedulers. As a common simulation kernel handles all process execution, MoCs are not well separated.

The above are not based on SystemC. Among the frameworks based on SystemC are HetSC [**?**], HetMoC [**?**] and ForSyDe [**?**], all having the disadvantage that instantiation of elements and controlling the synchronization have to be managed by the designer.

In the scope of [2], a mixed analog-digital systems proof-of-concept simulator has been developed [12], based on the SystemC AMS extension standard [17], [4]. Another simulator is proposed in [3]. Integration with software code for general-purpose CPUs and with an operating system is however not yet addressed in these approaches.

Outside the analog/mixed signal domain, UML/SysML based modeling techniques [24], [13] are popular with industry targeting embedded systems, but are still rarely used in the domain of heterogeneous system design. Furthermore, with few exceptions such as [22], [15], they do not lower the level of abstraction to cycle bit accurate level.

III. SystemC AMS Extensions

"SystemC AMS extensions" is a standard describing an extension of SystemC with AMS and RF features [23][17]. The usual approach for modeling the digital part of a heterogeneous system with SystemC [1] is to rely on the *Discrete Event* (DE) part of SystemC AMS extensions. The *Timed data Flow* (TDF) part adds support for signals where data values are sampled with a constant time step.

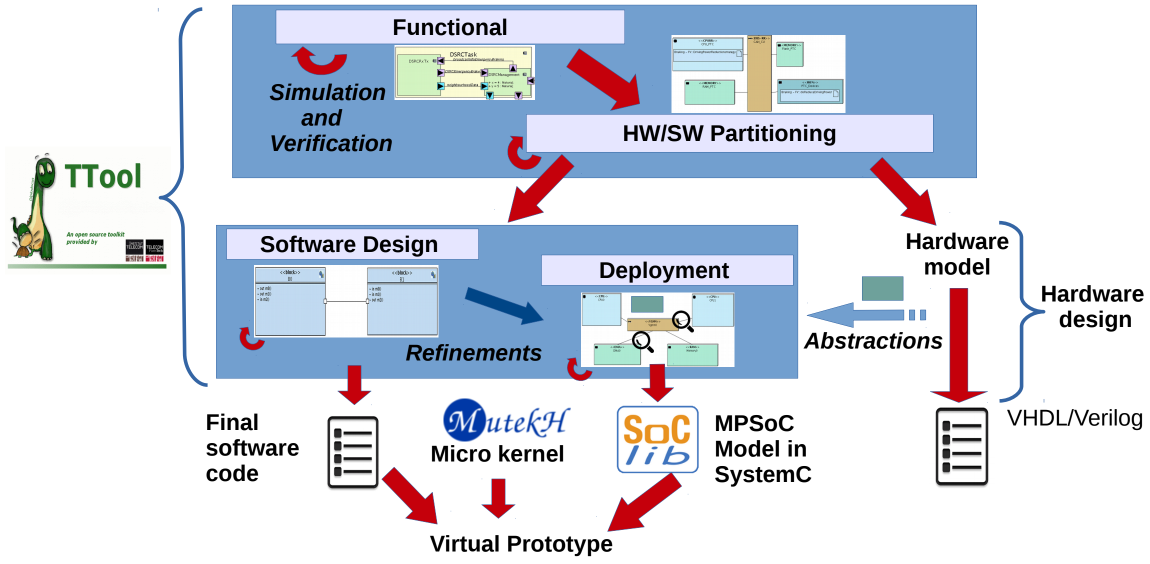

Fig. 1. Hardware/Software partitioning and Code generation for MPSoC platforms

A TDF module is described with an attribute representing the time step and a processing function. The time step is associated to a time period during which the processing function should be executed. The processing function corresponds to a mathematical function which depends on the module inputs and/or internal states. At each time step, a TDF module first reads a fixed number of samples from each of its input ports, then executes the processing function, and writes a fixed number of samples to each of its output ports.

A TDF port is described with three attributes:

- Tp represents the *time period*.
- R is the *rate* of data i.e. the number of read or written samples by a TDF port during each period.
- D models the *delay*, the number of samples of the TDF port when a simulation starts.

TDF modules can interact with the DE world (such as digital MPSoC platforms) using converter ports.

However, it is pointed out in [5] that it is hard to build a modeling environment synchronizing DE and TDF. Indeed, the TDF model of computation is based on the Synchronous Data Flow (SDF) formalism that considers models as a network of synchronous data flow blocks, and does not easily match the one of DE systems. Recent work [6] [10] models the interaction between TDF and DE by colored timed Petri Nets and checks causality issues between TDF and DE MoC automatically.

IV. High Level Modeling with TTool

TTool [7] is a SysML based, free and open-source software for model-based engineering of embedded systems at different abstraction levels: **functional**, **partitioning**, **software design**, **deployment**. The method associated to these levels [15] details how to take hardware/software partitioning decisions at a high level of abstraction (upper part of Figure 1), and to regularly validate these decisions during software development (middle part of Figure 1).

Models of platforms are described in the hardware/software partitioning and deployment stages. Tasks destined to be implemented in hardware are represented as *hardware accelerators*.

Models are thus composed of hardware and software parts. Software tasks for the partitioning model are captured within the functional abstraction level, and software tasks used in deployments are captured in the software design abstraction level. In both partitioning and deployment, the computation part of tasks is then deployed to processors (which can be hardware accelerators in the partitioning level), and the communication and storage part is deployed to buses and memories. An important advantage of TTool is that it offers an automated approach for formal verification and fast simulation on the three first levels of abstraction, for the digital part. Formal verification is based on internal model-checkers, or on external tools like UPPAAL [11].

On the lowest level (i.e. the deployment level), TTool offers the generation of a virtual prototype that can be simulated with a cycle bit accurate simulator for Multi Processor Systems on Chip (MPSoC) [14]. Processor models stem from the *SoCLib* [21] public domain library written in SystemC. SoCLib targets shared-memory *multiprocessor-on-chip system* architectures based on the *Virtual Component Interconnect* [**?**] standard which separates the components' function-

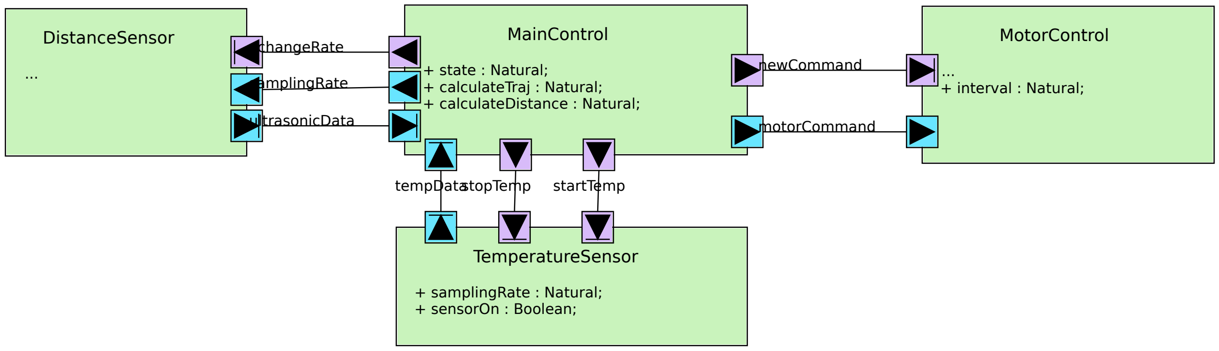

Fig. 2. Functional model of the rover

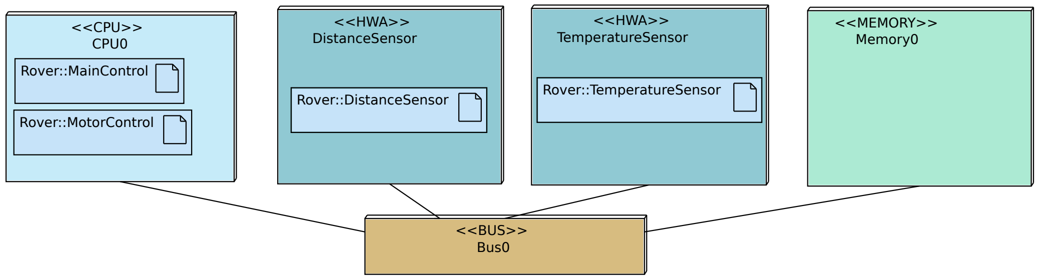

Fig. 3. Hardware/Software partitioning of the rover

ality from their communication. The lower left part of Figure 1 shows the generation of software code, cross-compiled for a general purpose processor and running under the MutekH [9] micro kernel on the SoCLib virtual prototype of the (purely digital) MPSoC. The initial Deployment diagram did not offer the possibility to show hardware, neither digital nor analog. The lower right part of the figure thus shows that we augment the Deployment Diagram with abstractions found for these hardware modules.

Several case studies have been performed in order to explain how the abstraction levels of TTool relate, e.g. an automotive obstacle detection [14] and a rover [16]. While sensors, GPS, radar, etc. can be approximately modeled with highly abstracted digital blocks, the accuracy of our models and verification would benefit from more realistic models taking into account the AMS part.

V. Integration of Analog Components

SystemC AMS uses the Timed Data Flow (TDF) Model of Computation (MoC) which is based on the timeless Synchronous Data Flow (SDF) [**?**]. So-called converter ports serve as interface between the TDF and DE MoC, raising potential causality issues. We adopted the solution described in [**?**].

A. *Contribution*

Our initial idea was to add analog components to the partitioning level, treating them at the same, very abstract, level as the (digital) hardware accelerators. After the partitioning stage, just like for such digital components, analog components could be developed independently from the software tasks. Yet, the development of analog components must be compatible with the partitioning decisions that can be revised in the next abstraction levels (software design, deployment).

We have augmented the graphical interface of TTool with the possibility to describe SystemC AMS blocks with their DE, TDF and converter ports. However, the behavior of the analog blocks, i.e. the *processing* function, must be provided directly in SystemC AMS since the abstraction of the behavior of these components is not likely to be easily modeled in a UML/SysML way. For the moment, the SystemC AMS code is entered in an editor associated to the module.

We are currently implementing the generation of SystemC AMS (TDF only) code of the components as well as the top cells from these mixed graphical/textual descriptions. This generation will have three phases:

1) In a first phase, we aim at generating fully functional SystemC AMS top cells and TDF models of analog components by limiting to platforms without any software part and without preexisting SoCLib components. For validation purpose, we have selected some simple existing SystemC AMS platforms from [3]: source-rectifier-sink, vibration sensor, etc. which we try to reproduce by our generator.

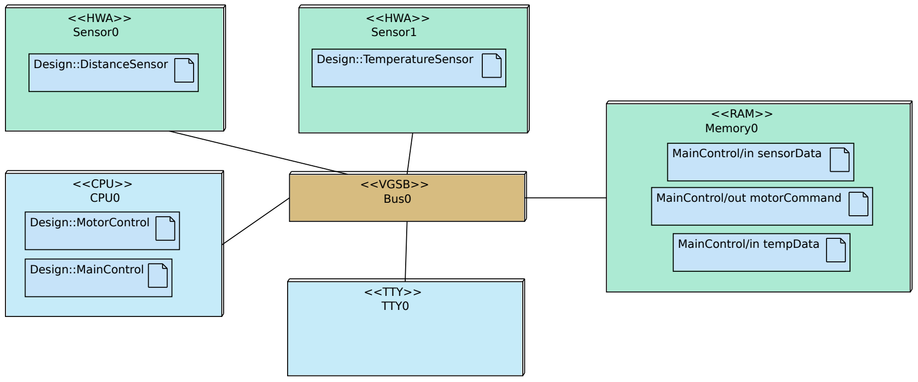

Fig. 4. Virtual Prototype

2) In a second phase, we intend to combine the SystemC AMS TDF part with the digital MPSoC platform (i.e. components using the DE MoC). The automated generation and configuration of purely digital versions of such a platform has already been addressed in previous contributions [14]. The (huge) remaining work is to combine this infrastructure with the analog part. On the software side, this phase will restrict to executing some assembler instructions directly loaded into memory, without the use of an operating system.
3) In a third phase, we intend to run larger scope software, requiring an operating system and linker script. We plan to reuse the SoCLib models of digital components, requiring an adaptation to the VCI standard.

B. Representing Analog Components in the MPSoC

The basic idea for the second and third phase is to extend the principle of "hardware accelerators" of the functional level of TTool to represent the integration of analog/mixed components. For each component described in SystemC AMS, a cycle-bit accurate interface will be generated. Only these interfaces will be "visible" for the MPSoC designer.

The separation of digital and analog design in TTool in different panels allows us to maintain formal verification (with UPPAAL, e.g. deadlocks and starvation) for the software part of the digital platform. The schedulability of the analog part is validated using the schedulability check of SystemC AMS [17].

VI. Case Study

A rover system meant to assist rescuers to find victims in debris is used as a case study. The rover features several sensors including distance and temperature sensors. Depending on the distance measured by the ultrasonic distance sensor, the sampling rate of the temperature sensor is adapted and the motor receives commands to speed up or slow down. The rover was initially described for a purely digital implementation, code generated for software tasks deployed on a SoC [16].

Figure 2 presents the functional model that abstract sampling rates of sensors with numerical (integer) values (type *Natural*). This model contains the two sensor blocks and the motor control, managed by a central control block communicating by channels (blue arrows) and through events (pink arrows). The current model is unable to express the analog nature of the temperature and distance signals. Moreover, the concrete values of the sampling rates cannot be given at this level.

In Figure 3, *MotorContol* and *MainControl* are mapped to the general purpose CPU, while the sensors are considered as *hardware accelerators*, without yet further specifying their (digital or analog) nature. Figure 4 shows the virtual prototype, where the software tasks, *MotorControl* and *MainControl*, are mapped to a general purpose processor using the method described in [14], The temperature and distance sensors are represented as *hardware accelerator* blocks on the partitioning level as well as in the overview of the virtual prototype.

A complete SystemC AMS model is out of the focus of this paper. Let us consider the distance sensor. Figure 5 shows the state machine on the partitioning (purely functional) level. Figure 6 shows the representation of the distance sensor as SystemC AMS cluster. Note that modeling is much more detailed. In fact, the sensor is modeled as a SystemC AMS cluster containing two distinct TDF blocks, one for the ultrasonic sensor itself and one for the analog to digital (AD) converter. The latter features three ports: an input port for the voltage issued from the ultrasonic sensor, an input converter port for sampling rate control, and an output converter port from TDF to the DE domain for the outgoing bit stream.

It should be noted that simulation rate change is not yet implemented, but that this will be possible with SystemC AMS v2.0. We thus do not yet model the *changeRate* event between *MainControl* and *DistanceSensor* in Figure 2. As an approximation, we implement the sampling rate as an integer value *s* to be read on the *soclibIn* port and which determines the number of iterations in a *for* loop: every *s* samples read from *sensorIn* a bit vector representing the current sample is written to *soclibOut*, the others are ignored. We assume a common rate of 1 for all ports of a module. The *maxVoltage* value can either be generated directly from TTool as done here or passed as a parameter.

The corresponding code will then be generated by TTool and the simulator solves the synchronization issues as mentioned before. Figure 7 finally shows the SystemC AMS code for the converter block of the distance sensor that we aim to generate from the SysML description. Delays of zero and rates of one are not shown.

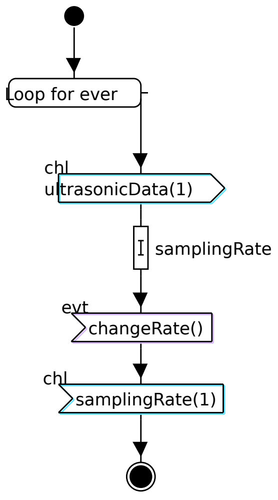

Fig. 5. Functional model of the distance sensor

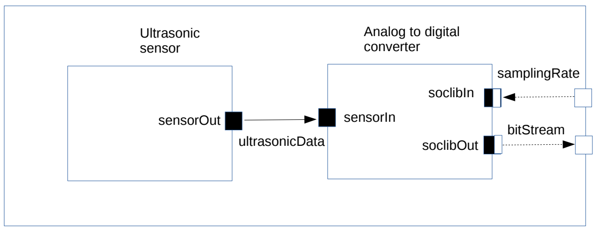

Fig. 6. SystemC AMS representation of the distance sensor

VII. Conclusion and Perspectives

We outline a method to integrate analog components into a multi-level modeling tool for complex embedded systems. [5] describes a formalization of the conversion between the digital and the AMS part in the form of timed colored Petri Nets. Thus, the results of [?] on co-simulation of analog and digital SoCLib components for virtual prototyping can be generalized.

The work is still at its beginning; we are currently working on the automatic generation of purely TDF platforms from SysML designs. The next step is the integration with the digital MPSoC. Once this step is achieved, we will be able to simulate more complex heterogeneous systems with extended software parts running on (digital) general-purpose processors and (light) operating systems.

In the spirit of SoCLib, we have also started to define a library of basic analog components that can be parametrized and configured to some degree. This will be another challenging task: each analog component has specific features like e.g. equations for which it is much harder to determine common aspects than for their digital counterparts.

References

[1] SystemC. In *http://www.systemc.org*.

[2] Beyond Dreams (Design Refinement of Embedded Analogue and Mixed-Signal Systems). 2008-2011.

[3] Heterogeneous Inception. In *https://www-soc.lip6.fr/trac/hinception*, 2012-2015.

[4] Accellera systems initiative. *SystemC AMS extensions Users Guide, Version 1.0.*

[5] L. Andrade, T. Maehne, A. Vachoux, C. Ben Aoun, F. Pêcheux, and M.-M. Louërat. Pre-Simulation Formal Analysis of Synchronization Issues between Discrete Event and Timed Data Flow Models of Computation. In *Design, Automation and Test in Europe, DATE Conference*, Mar. 2015.

[6] L. Andrade Porras. *Principles and implementation of a generic synchronization interface between SystemC AMS models of computation for the virtual prototyping of multi-disciplinary systems.* PhD thesis, Université Pierre et Marie Curie, 2016.

[7] L. Apvrille. TTool, an open-source toolkit for the modeling and verification of embedded systems. In *http://ttool.telecom-paristech.fr/*.

[8] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone, and A. L. Sangiovanni-Vincentelli. Metropolis: An integrated electronic system design environment. *IEEE Computer*, 36(4):45–52, 2003.

[9] A. Becoulet. Mutekh. http://www.mutekh.org.

[10] C. Ben Aoun. *Principles and Realization of a Virtual Prototyping Environment for Composable Heterogeneous Systems.* PhD thesis, Université Pierre et Marie Curie, 2017.

[11] J. Bengtsson and W. Yi. Timed automata: Semantics, algorithms and tools. In *Lecture Notes on Concurrency and Petri Nets*, pages 87–124. W. Reisig and G. Rozenberg (eds.), LNCS 3098, Springer-Verlag, 2004.

[12] K. Einwich. *SystemC AMS PoC2.1 Library, COSEDA, Dresden*, 2016. http://www.coseda-tech.com/systemc-ams-proof-of-concept/.

[13] A. Gamatié, S. L. Beux, É. Piel, R. B. Atitallah, A. Etien, P. Marquet, and J.-L. Dekeyser. A model-driven design framework for massively parallel embedded systems. *ACM Trans. Embedded Comput. Syst*, 10(4):39, 2011.

[14] D. Genius and L. Apvrille. Virtual yet precise prototyping: An automotive case study. In *ERTSS'2016*, Toulouse, Jan. 2016.

[15] D. Genius, L. W. Li, and L. Apvrille. Model-Driven Performance Evaluation and Formal Verification for Multi-level Embedded System Design. In *Conférence on Model-Driven*

```
template<int NBits>
class adconverter : public sca_tdf::sca_module {

public:
typedef sc_dt::sc_int<NBits> data_type;// bit vector

sca_tdf::sca_de::sca_in<double> sensorIn; // TDF input port
sca_tdf::sca_de::sca_in<data_type> soclibIn; // TDF input converter port
sca_tdf::sca_de::sca_out<data_type> soclibOut; // TDF output converter port

converter<NBits>::adconverter(sc_core::sc_module_name nm)
: sensorIn("in"), soclibIn("soclibIn"), soclibOut("soclibOut") {}

protected:
void set_attributes() {
  sensorIn.set_rate(1);
  sensorIn.set_delay(0);
  soclibIn.set_rate(1);
  soclibIn.set_delay(0);
  soclibOut.set_rate(1);
  soclibOut.set_delay(0);
}

void processing() {
  data_type res;
  int s =  soclibIn.read();
  int step;
  using namespace std;
  for(step=0; step < s; step++){
    double in = sensorIn.read();
    }
  sc_dt::sc_int<NBils> res = lround((in/maxVoltage)*((1<<(NBits-1))-1));
  soclibOut.write(res);
  }
}
```

Fig. 7. SystemC AMS model of the analog AD converter

Engineering and Software Development (Modelsward'2017), Porto, Portugal, Feb. 2017.

[16] D. Genius, L. W. Li, and L. Apvrille. Multi-level Latency Evaluation with an MDE Approach. In *Conferénce on Model-Driven Engineering and Software Development Modelsward'2018 (to appear)*, Funchal, Portugal, Jan. 2018.

[17] IEEE. *IEEE Std 1666.1 standard.* https://standards.ieee.org/findstds/standard/1666.1-2016.html, January 2016.

[18] K. Latif, M. Selva, C. Effiong, R. Ursu, A. Gamatie, G. Sassatelli, L. Zordan, L. Ost, P. Dziurzanski, and L. S. Indrusiak. Design space exploration for complex automotive applications: An engine control system case study. In *Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, RAPIDO '16, pages 2:1–2:7, NY, USA, 2016. ACM.

[19] E. A. Lee. Disciplined heterogeneous modeling. In D. Petriu, N. Rouquette, and O. Haugen, editors, *Proceedings of the ACM/IEEE 13th International Conference on Model Driven Engineering, Languages, and Systems (MODELS)*, pages 273–287. LNCS 6395, Springer-Verlag, Oct. 2010.

[20] C. Ptolemaeus, editor. *System Design, Modeling, and Simulation using Ptolemy II.* Ptolemy.org, 2014. http://ptolemy.org/books/Systems.

[21] SocLib consortium. The SoCLib project: An integrated system-on-chip modelling and simulation platform. Technical report, CNRS, 2003. www.soclib.fr.

[22] S. Taha, A. Radermacher, and S. Gérard. An entirely model-based framework for hardware design and simulation. In *DIPES/BICC*, volume 329 of *IFIP Advances in Information and Communication Technology*, pages 31–42. Springer, 2010.

[23] A. Vachoux, C. Grimm, and K. Einwich. Analog and mixed signal modelling with systemC-AMS. In *ISCAS (3)*, pages 914–917. IEEE, 2003.

[24] J. Vidal, F. de Lamotte, G. Gogniat, P. Soulard, and J.-P. Diguet. A co-design approach for embedded system modeling and code generation with UML and MARTE. In *DATE*, pages 226–231. IEEE, 2009.