Gabriela Nicolescu
Ian O'Connor
Christian Piguet *Editors*

# Design Technology for Heterogeneous Embedded Systems

Springer

# Design Technology for Heterogeneous Embedded Systems

Gabriela Nicolescu • Ian O'Connor •
Christian Piguet
Editors

# Design Technology for Heterogeneous Embedded Systems

Springer

*Editors*
Prof. Gabriela Nicolescu
Department of Computer Engineering
Ecole Polytechnique Montreal
2500 Chemin de Polytechnique Montreal
Montreal, Québec
Canada H3T 1J4
gabriela.nicolescu@polymtl.ca

Prof. Ian O'Connor
CNRS UMR 5270
Lyon Institute of Nanotechnology
Ecole Centrale de Lyon
av. Guy de Collongue 36
Bâtiment F7
69134 Ecully
France
ian.oconnor@ec-lyon.fr

Prof. Christian Piguet
Integrated and Wireless Systems Division
Centre Suisse d'Electronique et de
Microtechnique (CSEM)
Jaquet-Drotz 1
2000 Neuchâtel
Switzerland
christian.piguet@csem.ch

ISBN 978-94-007-1124-2 e-ISBN 978-94-007-1125-9
DOI 10.1007/978-94-007-1125-9
Springer Dordrecht Heidelberg London New York

Library of Congress Control Number: 2011942080



*Cover design*: VTeX UAB, Lithuania

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

# Contents

# List of Contributors

**El Mostapha Aboulhamid** Department of Computer Science and Operations Research, University of Montreal, 2920 Chemin de la Tour Montreal, Montreal, Canada H3T 1J4

**Gabriel Marchesan Almeida** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France

**Amara Amara** Institut Superieur d'Electronique de Paris (ISEP), 21 rue d'Assas, 75270 Paris, France

**Costin Anghel** Institut Superieur d'Electronique de Paris (ISEP), 21 rue d'Assas, 75270 Paris, France, costin.anghel@isep.fr

**David Atienza** Embedded Systems Laboratory (ESL), Faculty of Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, david.atienza@epfl.ch

**José L. Ayala** Embedded Systems Laboratory (ESL), Faculty of Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, jose.ayala@epfl.ch;
Department of Computer Architecture (DACYA), School of Computer Science, Complutense University of Madrid (UCM), Madrid, Spain, jayala@fdi.ucm.es

**Nadine Azemard** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France

**Cécile Belleudy** University of Nice-Sophia Antipolis, LEAT, CNRS, Bat. 4, 250 rue Albert Einstein, 06560 Valbonne, France, belleudy@unice.fr

**Marc Belleville** CEA Leti, MINATEC, Grenoble, France

**Pascal Benoit** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France, Pascal.Benoit@lirmm.fr

**Sébastien Bilavarn** University of Nice-Sophia Antipolis, LEAT, CNRS, Bat. 4, 250 rue Albert Einstein, 06560 Valbonne, France, bilavarn@unice.fr

**Dominique Borrione** TIMA Laboratory, 46 Avenue Félix Viallet, 38031 Grenoble Cedex, France, Dominique.Borrione@imag.fr

**Youcef Bouchebaba** STMicroelectronics, 16 Fitzgerald Rd, Ottawa, ON, K2H 8R6, Canada, youcef.bouchebaba@st.com

**Olivier Brousse** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France

**Jean-Frédéric Christmann** CEA-LETI, MINATEC, 17 avenue des Martyrs, 38054 Grenoble Cedex 9, France

**Fabien Clermidy** CEA Leti, MINATEC, Grenoble, France

**Cyril Condemine** CEA-LETI, MINATEC, 17 avenue des Martyrs, 38054 Grenoble Cedex 9, France, cyril.condemine@cea.fr

**Wim Dehaene** ESAT–MICAS, Katholieke Universiteit Leuven, Leuven, Belgium

**Jean-Luc Dekeyser** INRIA Lille Nord Europe–LIFL–USTL–CNRS, Parc Scientifique de la Haute Borne, Park Plaza–Batiment A, 40 avenue Halley, 59650 Villeneuve d'Ascq, France, jean-luc.dekeyser@lifl.fr

**Thibault Delavallée** Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, thibault.delavallee@uclouvain.be

**Damien Deleruyelle** IM2NP, Institut Matériaux Microélectronique Nanosciences de Provence, UMR CNRS 6242, Aix-Marseille Université, IMT Technopôle de Château Gombert, 13451 Marseille Cedex 20, France

**Alain Fourmigue** Department of Computer Engineering, Ecole Polytechnique Montreal, 2500 Chemin de Polytechnique Montreal, Montreal, Canada H3T 1J4, Alain.fourmigue@polymtl.ca

**Abdoulaye Gamatié** INRIA Lille Nord Europe–LIFL–USTL–CNRS, Parc Scientifique de la Haute Borne, Park Plaza–Batiment A, 40 avenue Halley, 59650 Villeneuve d'Ascq, France, abdoulaye.gamatie@lifl.fr

**Patrice Gerin** TIMA Laboratory, CNRS/Grenoble-INP/UJF, Grenoble, France, Patrice.Gerin@imag.fr

**Luiza Gheorghe** Department of Computer Engineering, Ecole Polytechnique Montreal, 2500 Chemin de Polytechnique Montreal, Montreal, Canada H3T 1J4

**Olivier Ginez** IM2NP, Institut Matériaux Microélectronique Nanosciences de Provence, UMR CNRS 6242, Aix-Marseille Université, IMT Technopôle de Château Gombert, 13451 Marseille Cedex 20, France

**Bruno Girodias** Department of Computer Engineering, Ecole Polytechnique Montreal, 2500 Chemin de Polytechnique Montreal, Montreal, Canada H3T 1J4

**Benoit Gosselin** Université Laval, Quebec, Canada, benoit.gosselin@gel.ulaval.ca

**Christoph Grimm** Institute of Computer Technology, Vienna University of Technology, Gußhausstraße 27-29/E384, 1040 Wien, Austria, grimm@ict.tuwien.ac.at

**S. Gyger** CSEM, Neuchâtel, Switzerland

**Jan Haase** Institute of Computer Technology, Vienna University of Technology, Gußhausstraße 27-29/E384, 1040 Wien, Austria, haase@ict.tuwien.ac.at

**Mian Muhammad Hamayun** TIMA Laboratory, CNRS/Grenoble-INP/UJF, Grenoble, France, Mian-Muhammad.Hamayun@imag.fr

**Da He** C-LAB, Paderborn University, Paderborn, Germany

**Yannick Hervé** Université de Strasbourg, Strasbourg, France, yrv@wanadoo.fr; Simfonia SARL, Strasbourg, France

**L. Labrak** CNRS UMR 5270, Lyon Institute of Nanotechnology, Ecole Centrale de Lyon, av. Guy de Collongue 36, Bâtiment F7, 69134 Ecully, France

**Walid Lafi** CEA-LETI, MINATEC, 17 avenue des Martyrs, 38054 Grenoble Cedex 9, France

**Didier Lattard** CEA-LETI, MINATEC, 17 avenue des Martyrs, 38054 Grenoble Cedex 9, France, didier.lattard@cea.fr

**Yusuf Leblebici** Microelectronic Systems Laboratory (LSM), Faculty of Engineering, EPFL, Lausanne, Switzerland, yusuf.leblebici@epfl.ch

**Yves Leduc** Advanced System Technology, Wireless Terminal Business Unit, Texas Instruments, Villeneuve-Loubet, France, y-leduc@ti.com

**Jean-Didier Legat** Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, jean-didier.legat@uclouvain.be

**Arnaud Legendre** Simfonia SARL, Strasbourg, France

**Igor Loiselle** Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, igor.loiselle@uclouvain.be

**Philippe Manet** Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, philippe.manet@uclouvain.be

**J.-M. Masgonty** CSEM, Neuchâtel, Switzerland

**Philippe Maurine** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France

**Samy Meftali** INRIA Lille Nord Europe–LIFL–USTL–CNRS, Parc Scientifique de la Haute Borne, Park Plaza–Batiment A, 40 avenue Halley, 59650 Villeneuve d'Ascq, France, samy.meftali@lifl.fr

**Nathalie Messina** Advanced System Technology, Wireless Terminal Business Unit, Texas Instruments, Villeneuve-Loubet, France

**Fabian Mischkalla** C-LAB, Paderborn University, Paderborn, Germany

**M. Morgan** CSEM, Neuchâtel, Switzerland

**Katell Morin-Allory** TIMA Laboratory, 46 Avenue Félix Viallet, 38031 Grenoble Cedex, France, Katell.Morin-Allory@imag.fr

**Wolfgang Mueller** C-LAB, Paderborn University, Paderborn, Germany

**Christophe Muller** IM2NP, Institut Matériaux Microélectronique Nanosciences de Provence, UMR CNRS 6242, Aix-Marseille Université, IMT Technopôle de Château Gombert, 13451 Marseille Cedex 20, France, christophe.muller@im2np.fr

**J.-L. Nagel** CSEM, Neuchâtel, Switzerland

**Gabriela Nicolescu** Department of Computer Engineering, Ecole Polytechnique Montreal, 2500 Chemin de Polytechnique Montreal, Montreal, Québec, Canada H3T 1J4, gabriela.nicolescu@polymtl.ca

**I. O'Connor** CNRS UMR 5270, Lyon Institute of Nanotechnology, Ecole Centrale de Lyon, av. Guy de Collongue 36, Bâtiment F7, 69134 Ecully, France, ian.oconnor@ec-lyon.fr

**Yann Oddos** TIMA Laboratory, 46 Avenue Félix Viallet, 38031 Grenoble Cedex, France, Yann.Oddos@imag.fr

**Frédéric Pétrot** TIMA Laboratory, CNRS/Grenoble-INP/UJF, Grenoble, France, Frederic.Petrot@imag.fr

**Pierre Paulin** STMicroelectronics, 16 Fitzgerald Rd, Ottawa, ON, K2H 8R6, Canada

**V. Peiris** CSEM, Neuchâtel, Switzerland

**C. Piguet** Integrated and Wireless Systems Division, Centre Suisse d'Electronique et de Microtechnique (CSEM), Jaquet-Drotz 1, 2000 Neuchâtel, Switzerland, christian.piguet@csem.ch

**Diego Puschini** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France; CEA Leti, MINATEC, Grenoble, France

**Imran Rafiq Quadri** INRIA Lille Nord Europe–LIFL–USTL–CNRS, Parc Scientifique de la Haute Borne, Park Plaza–Batiment A, 40 avenue Halley, 59650 Villeneuve d'Ascq, France, imran.quadri@lifl.fr

**Bettina Rebaud** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France; CEA Leti, MINATEC, Grenoble, France

**Michel Robert** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France

**Bertrand Rousseau** Université catholique de Louvain (UCL), Louvain-la-Neuve, Belgium, bertrand.rousseau@uclouvain.be

**D. Séverac** CSEM, Neuchâtel, Switzerland

**Gilles Sassatelli** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France

**Mohamad Sawan** École Polytechnique de Montréal, Montreal, Canada

**Arvind Sridhar** Embedded Systems Laboratory (ESL), Faculty of Engineering, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, arvind.sridhar@epfl.ch

**Lionel Torres** LIRMM, UMR 5506, CNRS–Université Montpellier 2, 161 rue Ada, 34095 Montpellier Cedex 5, France

**Yves Vanderperren** ESAT–MICAS, Katholieke Universiteit Leuven, Leuven, Belgium, yves.vanderperren@ieee.org

**Guy Waltisperger** CEA-LETI, MINATEC, 17 avenue des Martyrs, 38054 Grenoble Cedex 9, France

**Jérôme Willemin** CEA-LETI, MINATEC, 17 avenue des Martyrs, 38054 Grenoble Cedex 9, France

# Chapter 1
# Introduction

**G. Nicolescu, I. O'Connor, and C. Piguet**

Heterogeneous is an adjective, derived from Greek ("heteros", 'other' and "genos", 'kind'), used to describe an object or system that is composed of a number of items that are different from one another. It is the antonym of homogeneous, which signifies that the constituent parts of a system are of identical type. For example system heterogeneity in distributed systems refers to the existence, within the system, of different types of hardware and software, while data heterogeneity, in computing, refers to a mixing of data from two or more sources, often in two or more formats.

In fact, the term "heterogeneous" has many meanings, which originate to a large extent from the multiple fields of knowledge of the people that use the term. This book focuses on heterogeneity in the embedded systems and system on chip field, such that while we cannot pretend to establish an exhaustive list of the meanings of the term in this field, we can at least give our point of view.

The first and most obvious meaning is that more than one physical domain is involved in the functionality of the system. A domain represents a distinct part of the design space in which system components exist, and defines a set of common terminology, information on functionality and requirements for valid use. Indeed, a physical domain is based on the nature of the exchange of energy (i.e. power) used

G. Nicolescu
Department of Computer Engineering, Ecole Polytechnique Montreal, 2500 Chemin de Polytechnique Montreal, Montreal, Québec, Canada H3T 1J4
e-mail: gabriela.nicolescu@polymtl.ca

I. O'Connor (✉)
CNRS UMR 5270, Lyon Institute of Nanotechnology, Ecole Centrale de Lyon, av. Guy de Collongue 36, Bâtiment F7, 69134 Ecully, France
e-mail: ian.oconnor@ec-lyon.fr

C. Piguet
Integrated and Wireless Systems Division, Centre Suisse d'Electronique et de Microtechnique (CSEM), Jaquet-Drotz 1, 2000 Neuchâtel, Switzerland
e-mail: christian.piguet@csem.ch

G. Nicolescu et al. (eds.), *Design Technology for Heterogeneous Embedded Systems*,
DOI 10.1007/978-94-007-1125-9_1, 

in hardware component functionality, and in communication between the components. The term "domain" here is synonymous with discipline and stems from the use of the term in the sense of "field of knowledge". Some examples of physical domains are: electrical, mechanical, optical, thermal, hydraulic (using variations in the power of physical quantities such as charge, force, photon flux, temperature and volume flow as part of their functionality and/or as means of conveying information from one component to another).

Another meaning of "heterogeneous" concerns the target technology as the final fabric onto which all hardware components of the system will be deployed, and on which all component and system performance metrics ultimately depend. It can be defined as an association of raw materials (defining a set of material parameters), tools and process techniques (defining process parameters, including limits to physical geometries). This association leads to the technological fabrication process and the use of more than one basic material (silicon, III-V, organic, etc.) for the functional devices, whether by co-integration techniques (planar SoC[1] or stacked, i.e. 3D integrated circuits or 3DIC) or bonding (SiP[2]). When applied to SoC/SiP, it is implicit that one of the domains is electrical, and that one of the materials is silicon. From the technological viewpoint, many choices, issues and tradeoffs exist between the various packaging and integration techniques (above IC, SiP, heterogeneous integration, bulk, etc.).

While the general benefits of heterogeneous integration appear to be clear, this evolution represents a strong paradigm shift for the semiconductor industry. Moving towards diversification as a complement to the scaling trend that has lasted over 40 years is possible because the integration technology (or at least the individual technological steps) exists to do so. However, the capacity to translate system drivers into technology requirements (and consequently guidance for investment) to exploit such diversification is severely lacking. Such a role can only be fulfilled by a radical shift in design technology to address the new and vast problem of heterogeneous system design. However, the design technology must also remain compatible with standard "More Moore" flows, which are geared towards handling complexity both in terms of detail as device dimensions shrink (silicon complexity) and in terms of sheer scale of systems as the number of devices in the system grows (system complexity). Indeed, the micro-electronics industry, over the years and with its spectacular and unique evolution, has built its own specific design methods while focusing mainly on the management of complexity through the establishment of abstraction levels. Today, the emergence of device heterogeneity requires new approaches enabling the satisfactory design of heterogeneous embedded systems for the widespread deployment of such systems.

Other meanings of "heterogeneous" relate to the design process involved before the object exists physically (specification, synthesis, simulation, verification). In fact, the term was first commonly used to describe systems based on (digital) hardware and software, which can be more generally defined as system description

[1] System on Chip.

[2] System in Package.

using more than one level of abstraction—both for hardware and signal description. This is essentially driven by the need to handle the massive complexity of SoC/SiP by simplifying assumptions, and which in turn drives many of the requirements for modeling, design and simulation techniques of non-digital hardware. In fact in addition to the notion of physical domain, the use of abstract domain types in the design process is necessary to completely define the sphere of operation of a component. Multiple abstract domains exist and are either based on the intrinsic nature of the component functionality and communication, or on the way that this nature is described (usually in terms of reducing the description content to a strict minimum as a means of optimizing simulation time).

Specific branches of heterogeneity can also be identified, concerning the use of more than one behavioral domain (such as causal, synchronous, discrete, signal-flow, conservative), all of which find varying degrees of accuracy of data and of time. Various data domains exist, such as untyped objects or tokens, enumerated symbols, real values, integer values, logic values. The means of taking time into account also leads to various time-domains (e.g. continuous-time, discrete-time) and abstractions (cycle accurate, transaction accurate). While the temporal analysis of systems is necessary, it is not the only analysis domain where relevant information is to be found concerning performance metrics of (particularly continuous-time) components: analyses include time-domain, frequency-domain, static-domain, modulated time/frequency-domain. Generally, the heterogeneity of abstractions and models implies a heterogeneity of tools: a set of tools are required for a complete design flow.

Mastering heterogeneity in embedded systems by design technology is one of the most important challenges facing the semiconductor industry today and will be for several years to come. This book, compiled largely from a set of contributions from participants of past editions of the Winter School on Heterogeneous Embedded Systems Design Technology (FETCH), proposes a necessarily broad and holistic overview of design techniques used to tackle the various facets of heterogeneity in terms of technology and opportunities at the physical level, signal representations and different abstraction levels, architectures and components based on hardware and software, in all the main phases of design (modeling, validation with multiple models of computation, synthesis and optimization). It concentrates on the specific issues at the interfaces, and is divided into two main parts. The first part examines mainly theoretical issues and focuses on the modeling, validation and design techniques themselves. The second part illustrates the use of these methods in various design contexts at the forefront of new technology and architectural developments. The following sections summarize the contributions of each chapter.

## 1 Methods, Models and Tools

A model is a representation of a complex system and/or its environment. The system to be designed is a collection of communicating components that are organized

through its architecture for a common purpose. The architecture defines the way in which the components are organized together to form a system. In order for each component to carry out a useful role within the system, it must be connectible in some way to other elements. Its functionality is thus defined by the conjunction of a particular behavior and an interface for communication. The functionality is manifest through the states of the component, which can evolve over time. The functionality of the component can be quantified by parameters and performance metrics.

Modeling is the first step of any design and/or analysis activity related to the architecture and function of a system. In general, a model is associated with an abstraction level and represents the set of properties specific to this level. The model enables a hierarchical design process where (i) the entire system and its environment are first represented at an abstract level, and (ii) model transformations refine this design and add detail as necessary to solve the design problem. The first part of this book will cover these various necessary phases: starting with the means of specifying design intent, the use of abstraction levels and refinement strategies are then described, ending with the two main activities in the design process, simulation (analysis) and design (synthesis).

## *1.1 Specifications*

The design of embedded systems is not an isolated hardware design activity, and must take into account higher-level component descriptions, including software. The highest abstraction level for the description of a system is that used by the specifications.

In Chap. 2, the authors make the case that a general purpose modeling language (UML[3]) can be customized to the specific purpose of modeling electronic systems. The chapter covers recent advances of the UML language applied to SoC and hardware-related embedded systems design through several examples of specific UML profiles relevant to SoC design. Linking UML to existing hardware/software design languages and simulation environments is a key point to the discussion, and is illustrated through a concrete example of a UML profile for hardware/software co-modeling and code generation.

In Chap. 3, the authors show how it is possible to separate the key concerns of data and control to achieve an executable specification flow. The benefits of this technology-independent methodology are demonstrated through the validation and verification of complex systems accommodating a mixture of hardware and software, analog and digital, electronic and micromechanics components.

[3] Unified Modeling Language (http://www.uml.org).

## 1.2 Modeling, Abstraction and Reuse

Due to the sustained and exponential growth in complexity in embedded SoC architectures, exploration and performance estimation, particularly early in the design cycle, is becoming an increasingly difficult challenge. With the advent of the nanotechnology era, billions of transistors are available to form high-performance systems, and it is impossible to consider the transistor as the building block. It is widely acknowledged that to solve this problem, a continuum of model abstraction levels must be established during the course of the design of a system. A model at a higher abstraction level will hide certain characteristics of the object that it models, either in terms of performance metrics, or in terms of internal architecture, such that simulation at higher levels of abstraction can perform early validation of software. Models can be refined (i.e. made more accurate) by moving down the continuum of abstraction levels, implying that additional model characteristics must be defined. Through the selected design methodology (design space exploration, optimization, trial and error, etc.) performance metrics can be specified and/or architectural variants defined.

In Chap. 4, the authors propose a general autonomous integrated system model with functional, technological and structural scalability in mind. The model handles extensions to the scale of the system in distributed MPSoC systems, as well as to the dynamic nature of the system in reconfigurable and autonomous computing. The authors wrap up by considering more long term trends towards multi-agent bio-inspired approaches.

Chapter 5 covers techniques to achieve the goal of fast hardware/software system simulation on MPSoCs[4] for design choices and design validations. The authors argue that native simulation (rather than instruction-set simulation for example) is a promising solution to simulate software that can be linked to abstract hardware simulators for both functional and temporal hardware/software system validation.

Chapter 6 presents a high level component based approach for expressing system reconfigurability in SoC co-design. The authors firstly present a generic model for reactive control in SoC co-design. This allows the integration of control at different abstraction levels, in particular at the higher abstraction levels, as well as reconfigurability features. The work is validated through a case study using the UML MARTE[5] profile for the modeling and analysis of real-time embedded systems.

## 1.3 Simulation and Validation of Complex Systems

The refinement of models describing complex systems has to be validated, which is usually realized using simulation. For this it is necessary to define global executable models which require a model integration strategy. The strategy is based on

[4] Multi-Processor System on Chip.

[5] Modeling and Analysis of Real-Time and Embedded Systems (http://www.omgmarte.com).

the establishment of interfaces accommodating the various models types. Another possibility for validation is formal verification. This implies that all the models must be expressed using a given formalism, such that the set of properties can be verified.

The tradeoffs involved in validation are particularly evident in Chap. 7, which presents a hybrid platform composed of a simulation tool and a testbed environment to facilitate the design and improve test accuracy of new wireless protocols. The complexity of these protocols requires fast and accurate validation methodologies, and the authors combine the flexibility of simulation-based approaches with the speed and capability of testing designs in real life settings using testbed platforms.

Chapter 8 examines property-based verification, and its dynamic extension as applied to large systems that defeat formal verification methods. The authors describe a verification system in which temporal properties are automatically translated into synthesizable IPs,[6] while resulting monitors and generators are automatically connected to the design under verification.

## *1.4 Design, Optimization and Synthesis*

Methodologies for the design of heterogeneous embedded systems using advanced technologies are critical to achieve, such that functionality can be reliably guaranteed, and such that performance can be optimized to one or more criteria. The establishment of methodologies is increasingly complex, since it has to cope with both very high-level system descriptions and low-level aspects related to technology and variability. Particular issues concern the formalization of global specifications and their continuous validation during design, hardware/software co-development, clarifying interaction between multiple concerns and physical domains. Tooling to support design methodologies also now inevitably at some stage uses numerical optimization. For hardware/software heterogeneity, a major challenge is the efficient mapping of software applications onto parallel hardware resources. This is a non-trivial problem because of the number of parameters to be considered for characterizing both the applications and the underlying platform architectures. For physical domain heterogeneity, the main issue is to build appropriate "divide and conquer" partitioning strategies while allowing the exploration of tradeoffs spanning several domains or abstraction levels.

Chapter 9 covers the major low-level issues (dynamic and static power consumption, temperature, technology variations, interconnect, reliability, yield), and their impact on high-level design, such as the design of multi-supply voltage, fault-tolerant, redundant or adaptive chip architectures. The authors illustrate their methodology through three heterogeneous multi-processor based systems: wireless sensor networks, vision sensors and mobile television.

In Chap. 10, the authors propose a new framework for the mapping of applications onto a many-core computing platform. This extensible framework allows

[6] Intellectual Property.

the exploration of several meta-heuristics, the addition of new objective functions with any number of architecture and application constraints. Experimental results demonstrate the relevance of using new meta-heuristics, and the power of the parallel framework implementation by significantly increasing the explored solution space.

The Functional Virtual Prototyping methodology is covered in Chap. 11, where a virtual prototype is defined as a model composed of multiple abstraction levels of a multi-domain system. The authors argue that this methodology can enable the formalization, exchange, and reuse of design knowledge, and that its widespread use could solve several issues in complex systems design, particularly for collaboration between multiple and geo-distributed design groups.

Chapter 12 addresses design complexity as related to multi-physics systems with a methodology based on hierarchical partitioning and multi-level optimization. The authors show how an optimization problem including cross-domain variables can be formulated to enable the exploration of trade-offs at an early design stage. The methodology is put into practice with an experimental framework, and is demonstrated with the optimization of the fill-factor and response speed in an active pixel sensor.

## 2 Design Contexts

From the miniaturization of existing systems (position sensors, labs on chip, etc.) to the creation of specific integrated functions (memory, RF tuning, energy, etc.), nanoscale and non-electronic devices are being integrated to create nanoelectronic and heterogeneous SoC/SiP and 3DICs. This approach will have a significant impact on several economic sectors and is driven by:

- the need for the miniaturization of existing systems to benefit from technological advances and improve performance at lower overall cost,
- the potential replacement of specific functions in SoC/SiP with nanoscale or non-electronic devices (nanoswitches, optical interconnect, magnetic memory, etc.),
- the advent of high-performance user interfaces (virtual surgical operations, games consoles, etc.),
- the rise of low-power mobile systems (communications and mobile computing) and wireless sensor networks for the measurement of phenomena inaccessible to single-sensor systems.

The second part of this book will cover design contexts from two points of view: firstly, how new technologies impact the design process, and secondly, how novel distributed autonomous sensor systems can be designed.

## 2.1 Designing with Emerging Technologies

New integration and fabrication technologies continually stretch the limits of design technology. 3D integration, consisting of stacking many chips vertically and connecting them together using Through Silicon Vias (TSVs) is a promising solution for heterogeneous systems, providing several benefits in terms of performance and cost. For example the first 3D processor (the 2008 Rochester Cube[7]) runs at 1.4 GHz and has abilities that the conventional planar chip cannot reach. However, the design process faced many difficulties because of the complexity of the design, such as ensuring synchronized operation of all of the layers and seamless inter-layer communication. 3D chips also demonstrate significant thermal issues (and consequently higher static power and lower reliability) due to the presence of processing units with a high power density, which are not homogeneously distributed in the stack.

The scaling of silicon technology and the integration of novel functional materials are also enabling exploration for alternative concepts for information processing, storage and communication in computing platforms. Adoption of such new technologies will only occur if significant improvement in the conventional compute figure of merit (number of operations per second·Watt·mm$^3$). As well as information technology, emerging technologies are also solving issues in biomedical applications, through the development of efficient low-power interface circuits between living objects and data gathering.

Chapter 13 analyzes both near-term and long-term technology alternatives for memory and logic. Scaling of conventional circuits is considered, as well as the development of novel logic circuits based on carbon nanotubes for reconfigurable circuits, nanowire crossbar matrices for memory, and graphene nanoribbons. Hybrid molecular-CMOS architectures, the most likely first step towards alternative architectures, are also discussed.

In Chap. 14, a new approach for the thermal control of 3D chips is discussed. The authors use both grid and non-uniform placement of TSVs as an effective mechanism for thermal balancing and control in 3D chips. A large part of the chapter is dedicated to the mathematical modeling of the material layers and TSVs, including a detailed calibration phase based on a real 5-tier 3D chip stack with several heaters and sensors to study the heat diffusion.

Chapter 15 covers 3D integration solutions for heterogeneous systems, with an overview of 3D manufacturing technologies and related concerns. An outlook to some potential applications is given, with particular focus on 3D MPSoC architectures for compute intensive systems.

In Chap. 16, the authors focus on alternative devices capable of switching between two distinct resistive states for non-volatile memories. In particular, the materials and their ability to withstand a downscaling of their critical dimensions are described. Particular attention is also given to the models describing the operation

[7] V.F. Pavlidis, E.G. Friedman, Three-Dimensional Integrated Circuit Design, Morgan Kaufman, 2009.

of such memory cells, and their implementation in electrical simulators to evaluate their robustness at the architectural level.

Chapter 17 presents dedicated circuit techniques and strategies to design and assemble dense embedded microsystems target towards bioelectrical signal recording applications. Efficient interface circuits to measure the weak bioelectrical signal from several cells in the cortical tissues are covered, and high-fidelity data-reduction strategies are demonstrated. Since power issues are predominant in in-vivo applications, particular attention is paid to on-chip power management schemes based on automatic biopotential detection, as well as low-power design techniques, ultra-low-power neural signal processing circuits, and dedicated implementation strategies enabling high multi-channel neural recording microsystem integration density.

## 2.2 *Designing Smart, Self-powered Radio Systems—Extreme Heterogeneity*

Perhaps the most extreme cases of heterogeneity in embedded systems today are in the field of sensor networks with wireless communication between nodes. Energy-autonomy is key to the deployment of such distributed sensor systems, since the sensor nodes have batteries or energy harvesters and must therefore be designed to consume very little power. Understanding at an early design stage the impact of design choices on power is a critical design step. Moreover, the energy harvesting system itself is a particularly heterogeneous system, including energy harvester, battery, antenna, sensors and electronic blocks. The main issues for this kind of system are the energy converter efficiency for small power transfer, the load consumption (RF,[8] sensors) in active and standby mode, and the embedded power management.

The application domains of such complex systems can be found in military, security, or high reliability systems such as aerospace and automotive. Meeting the joint constraints of specific requirements, multiple standards, low volume, high-reliability and cost mean that reconfigurable or reprogrammable hardware is the favored approach in this area.

Chapter 18 is focused on the integration of an energy- and data-driven platform for autonomous systems. The authors describe a global system description and specification, as well as the principles of three energy-harvesting techniques (mechanical vibrations, thermal flux, and solar radiation). The use of multiple sources for energy harvesting is shown to be feasible, when strategies for power management are considered with a focus on power path optimization.

In Chap. 19, the authors present a power model suited for multiprocessor power management based on the study of a video decoder application. The execution of this application on the multiprocessor, and the impact of the memory architecture on the energy cost, is analyzed in detail, as well as a power strategy suited to video processing based on the selection of operating points of frequency and voltage.

[8] Radiofrequency.

Chapter 20 covers the use of high-performance reconfigurable platforms in software-defined radio for multi-standard applications. Additional flexibility is explored through novel techniques of dynamic partial reconfiguration.

In Chap. 21, the authors propose an approach for complete system simulation and power estimation of wireless sensor networks, ultimately enabling sensor-node optimization at the architecture level. The authors argue that the accurate estimation of the power consumption of network nodes, requires both accurate and efficient modeling of the communication infrastructure and the architecture of the node. To achieve these goals, the developed simulation framework includes an Instruction Set Simulator and uses Transaction Level Modeling (TLM) with SystemC as the basis for simulation.

**Acknowledgements** We would like to take this opportunity to thank all the contributors to this book for having undertaken the writing of each chapter from the original winter school presentations and for their patience during the review process. We also wish to extend our appreciation to the team at Springer for their editorial guidance as well of course as for giving us the opportunity to compile this book together.

# Part I
# Methods, Models and Tools

# Chapter 2
# Extending UML for Electronic Systems Design: A Code Generation Perspective

**Yves Vanderperren, Wolfgang Mueller, Da He, Fabian Mischkalla, and Wim Dehaene**

## 1 Introduction

Larger scale designs, increased mask and design costs, 'first time right' requirements and shorter product development cycles motivate the application of innovative 'System on a Chip' (SoC) methodologies which tackle complex system design issues.[1] There is a noticeable need for design flows towards implementation starting from higher level modeling. The application of the Unified Modeling Language (UML) in the context of electronic systems has attracted growing interest in the recent years [16, 35], and several experiences from industrial and academic users have been reported [34, 58].

Following its introduction in 1995, UML has been widely accepted in software engineering and supported by a considerable number of Computer Aided Software Engineering (CASE) tools. Although UML has its roots in the software domain, the Object Management Group (OMG), the organization driving the UML standardization effort [44, 45], has turned the UML notation into a general-purpose modeling language which can be used for various application domains, ranging from business process to engineering modeling, mainly for documentation purposes. Besides the language complexity, the main drawback of such a broad target is the lack of

[1] While the term 'SoC' is commonly understood as the packaging of all the necessary electronic circuits and parts for a system on a *single* chip, we consider the term in larger sense here, and cover electronic systems irrespective of the underlying implementation technology. These systems, which might be multi-chip, involve several disciplines including specification, architecture exploration, analog and digital hardware design, the development of embedded software which may be running on top of a real-time operating system (RTOS), verification, etc.

Y. Vanderperren (✉) · W. Dehaene
ESAT–MICAS, Katholieke Universiteit Leuven, Leuven, Belgium
e-mail: yves.vanderperren@ieee.org

W. Mueller · D. He · F. Mischkalla
C-LAB, Paderborn University, Paderborn, Germany

G. Nicolescu et al. (eds.), *Design Technology for Heterogeneous Embedded Systems*,
DOI 10.1007/978-94-007-1125-9_2, 

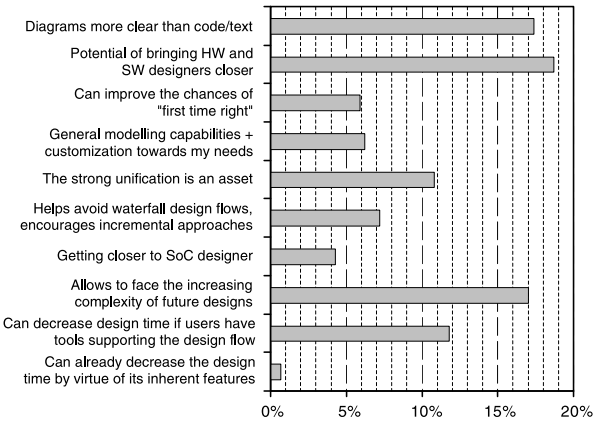


**Fig. 2.1** Positive aspects of UML [62]

sufficient semantics, which constitutes the main obstacle for real engineering application. Therefore, application specific customizations of UML (UML profiles), such as the System Modeling Language (SysML) [38] and the UML Profile for SoC [42], are of increasing importance. The addition of precise semantics allows for the automatic generation of code skeleton, typically C++ or Java, from UML models.

In the domain of embedded systems, the complexity of embedded software doubled every 10 months in the last decades. Automotive software, for instance, may exceed several GBytes [22]. In this domain, complexity is now successfully managed by model-based development and testing methodologies based on MATLAB/Simulink with highly efficient C code generation. Unfortunately, the situation is more complex in electronic systems design than in the embedded software domain, as designers face a combination of various disciplines, the coexistence of multiple design languages, and several abstraction levels. Furthermore, multiprocessor architectures have become commonplace and require languages and tool support for parallel programming. In this multi-disciplinary context,

UML has great potential to unify hardware and software design flows. The possibility to bring designers of both domains closer and to improve the communication between them was recognized as a major advantage, as reported by surveys conducted during the UML-SoC Workshops at the Design Automation Conference (DAC) in 2006 (Fig. 2.1) and 2007 [63]. Additionally, UML is also perceived as a means to manage the increasing complexity of future designs and improve their specification. UML diagrams are expected to provide a clearer overview compared to text.

Significant issues remain, however, such as the perceived lack of maturity of tool support, the possible difficulty of acceptance by designers due to lack of knowledge, and the existence of different UML extensions applicable to SoC design but which are not necessarily compatible [62].

A detailed presentation of the UML is beyond the scope of this chapter and we assume that the reader has a basic knowledge of the language. The focus of this

chapter is the concrete application of UML to SoC design, and follows the following structure. The next section introduces basic concepts of the UML extension mechanism, how to define a UML profile. Thereafter, we present some UML profiles relevant for UML for SoC and embedded systems design before we introduce one application in the context of SystemC/C++ co-simulation and -synthesis [27].

## 2 Extending UML

A stereotype is an extensibility mechanism of UML which allows users to define modeling elements derived from existing UML classifiers, such as classes and associations, and to customize these towards individual application domains [45]. Graphically, a stereotype is rendered as a name enclosed by «...». The readability and interpretation of models can be highly improved by using a limited number of well defined stereotypes. Additionally, stereotypes can add precise meanings to individual elements, enabling automatic code generation.

For instance, stereotypes corresponding to SystemC constructs can be defined, such as «*sc_module*», «*sc_clock*», «*sc_thread*», «*sc_method*» etc. The individual elements of a UML model can then be annotated with these stereotypes to indicate which SystemC construct they correspond to. The resulting UML model constitutes a first specification of a SystemC model, which can then be automatically generated. The stereotypes give to the UML elements the precise semantics from the target language (SystemC in this case).

As an example, Fig. 2.2 represents a Class Diagram with SystemC-oriented stereotypes. It corresponds to the simple bus example delivered with SystemC, with master, slave, and arbiter classes stereotyped as «*sc_module*» and connected to a bus. Modules are connected by a directed association with stereotype «*connect*». We introduce this stereotype as an abstraction for a port with associated interface where the flow points into the direction of the interface. An alternative and more detailed representation of the bus connection is provided by the explicit definition of the interface via a separate element with stereotype «*sc_interface*» (Fig. 2.3). Such examples illustrate how stereotypes add necessary interpretations to UML diagrams. A clear definition and structure of stereotypes is of utmost importance before applying UML for effective documentation and efficient code generation.

UML is defined on the basis of a metamodel, i.e., the UML language is itself described based on a model. This approach makes the language extremely flexible, since an application specific customization can be easily defined by the extension of that metamodel trough the definition of stereotypes. In theory, the principle of an application specific customization of UML through a so-called *UML profile* through stereotypes is simple. Considering a specific application domain, all unnecessary parts of the UML metamodel are stripped in a first step. In a second step, the resulting metamodel is extended. This mainly means the definition of a set of additional stereotypes and tagged values, i.e., stereotype attributes. In further steps, useful graphical icons/symbols, constraints, and semantic outlines are added.

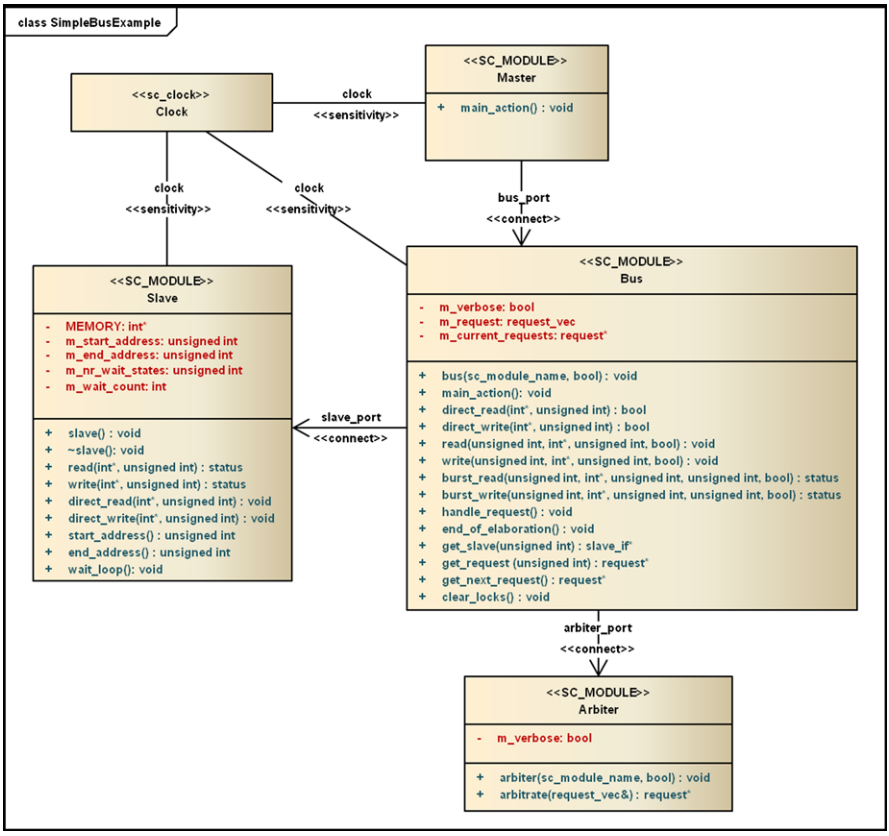


**Fig. 2.2** UML simple bus class diagram

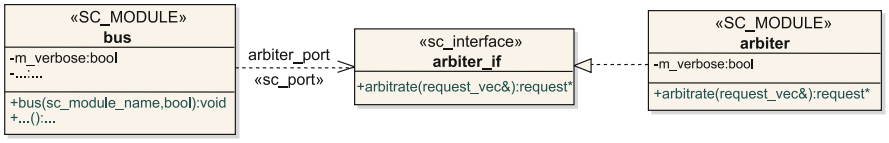


**Fig. 2.3** UML arbiter interface

In practice, the first step is often skipped and the additional semantics weak, leaving room for several interpretations.

Definitions of stereotypes are often given in the form of a table. In most cases, an additional set of Class Diagrams is given, as depicted for example in Fig. 2.4, which shows an excerpt from the UML profile for SoC [42], which will be further discussed in Sect. 3.1. The extended UML metaclass *Port* is indicated by the keyword «*metaclass*». For its definition, the stereotype *SoCPort* is specified with the keyword «*stereotype*» and linked with an extension relationship (solid line link with

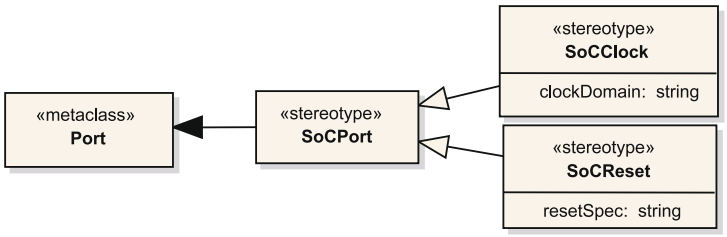


**Fig. 2.4** UML stereotype definition

black head). The two other extensions, *SoCClock* and *SoCReset*, are simply specified with their tagged values as generalizations of *SoCPort*. After having defined those extensions, the stereotypes «*SoCPort*», «*SoCClock*», and «*SoCReset*» can be applied in Class Diagrams.

Several UML profiles are available as OMG standards and applicable to electronic and embedded systems modeling, such as the UML Testing Profile [39], the UML Profile for Modeling Quality of Service (QoS) and Fault Tolerance Characteristics and Mechanisms [40], the UML Profile for Schedulability, Performance and Time (SPT) [41], the UML Profile for Systems Engineering (which defines the SysML language) [38], the UML Profile for SoC [42], and MARTE (Modeling and Analysis of Real-Time Embedded Systems) [43].

The following sections will focus on the most important ones in the context of SoC design.

## 3 UML Extensions Applicable to SoC Design

### *3.1 UML Profile for SoC*

The UML profile for SoC was initiated by CATS, Rational (now part of IBM), and Fujitsu in 2002. It is available as an OMG standard since August 2006 [42]. It targets mainly Transaction Level Modeling (TLM) SoC design and defines modeling concepts close to SystemC. Table 2.1 gives a summary of several stereotypes introduced in the profile and the UML metaclasses they extend.

The SoC profile introduces *Structure Diagrams* with special symbols for hierarchical modules, ports, and interfaces. The icons for ports and interfaces are similar to those introduced in [23]. Annex A and B of the profile provide more information on the equivalence between these constructs and SystemC concepts. Automatic SystemC code generation from UML models based on the SoC Profile is supported by tools from CATS [12] and the UML tool vendor ArtisanSW [19].

**Table 2.1** Examples of stereotypes defined in the UML profile for SoC

| SoC model element | Stereotype | UML metaclass |
|---|---|---|
| Module | SoCModule | Class |
| Process | SoCProcess | Operation |
| Data | Data | Class |
| Controller | Controller | Class |
| Protocol Interface | SoCInterface | Interface |
| Channel | SoCChannel | Class |
| Port | SoCPort | Port/Class |
| Connector | SoCConnector | Connector |
| Clock Port | SoCClock | Port |
| Reset Port | SoCReset | Port |
| Data Type | SoCDataType | Dependency |

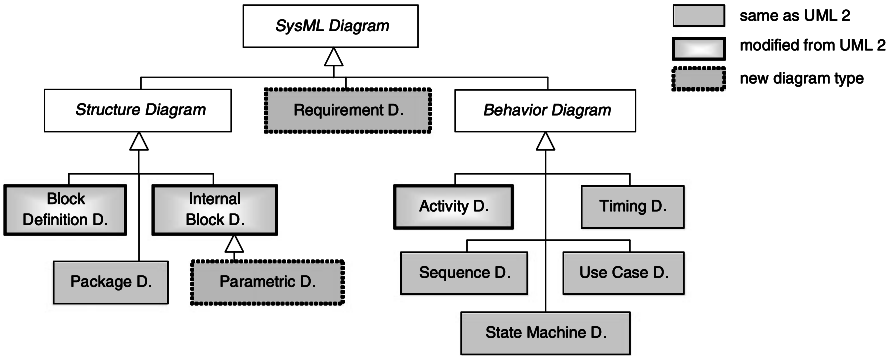


**Fig. 2.5** Architecture of SysML

## *3.2 SysML*

SysML is a UML profile which allows modeling systems from a domain neutral and Systems Engineering (SE) perspective [38]. It is the result of a joint initiative of OMG and the International Council on Systems Engineering (INCOSE). The focus of SE is the efficient design of complex systems which include a broad range of heterogeneous domains, including hardware and software. SysML provides opportunities to improve UML-based SoC development processes with the successful experiences from the SE discipline [59]. Strong similarities exist indeed between the methods used in the area of SE and complex SoC design, such as the need for precise requirements management, heterogeneous system specification and simulation, system validation and verification. The architecture of SysML is represented on Fig. 2.5. The main differences are summarized hereafter:

- **Structure:** SysML simplifies the UML diagrams used to represent the structural aspects of a system. It introduces the concept of *block*, a stereotyped class which

describes a system as a structure of interconnected parts. A block provides a domain neutral modeling element that can be used to represent the structure of any kind of system, regardless of the nature of its components. In the context of a SoC, these components can be hardware or software based as well as analog or digital.

- **Behavior:** SysML provides several enhancements to Activity Diagrams. In particular, the control of execution is extended such that running actions can be disabled. In UML, the control is limited to the determination of the moment when actions start. In SysML a behavior may not stop itself. Instead it can run until it is terminated externally. For this purpose SysML introduces *control operators*, i.e., behaviors which produce an output controlling the execution of other actions.
- **Requirements:** One of the major improvements SysML brings to UML is the support for representing requirements and relating them to the models of a system, the actual design and the test procedures. UML does not address how to trace the requirements of a system from informal specifications down to the individual design elements and test cases. Requirements are often only traced to UML use cases but not to the design. Adding design rationale information which captures the reasons for design decisions made during the creation of development artifacts, and linking these to the requirements help analyze the consequences of a requirement change. SysML introduces for this purpose the *Requirement Diagram*, and defines several kinds of relationships improving the requirement traceability. The aim is not to replace existing requirements management tools, but to provide a standard way of linking the requirements to the design and the test suite within UML and a unified design environment.
- **Allocations:** The concept of *allocation* in SysML is a more abstract form of deployment than in UML. It is a relationship established during the design phase between model elements. An allocation provides the generalized capability to a source model element to a target model element. For example, it can be used to link requirements and design elements, to map a behavior into the structure implementing it, or to associate a piece of software with the hardware deploying it.

SysML presents clear advantages. It simplifies UML in several aspects, as it actually removes more diagrams than it introduces. Furthermore, SysML can support the application of Systems Engineering approaches to SoC design. This feature is particularly important, since the successful construction of complex SoC systems requires a cross-functional team with system design knowledge combined with experienced SoC design groups from hardware and software domains which are backed by an integrated tool chain. By encouraging a Systems Engineering perspective and by providing a common notation for different disciplines, SysML allows facing the growing complexity of electronic systems and improving communication among the project members.

However, SysML remains a semi-formal language, like UML. Although SysML contributes to the applicability of UML to non-software systems, it remains a semi-formal language since it lacks associated semantics. For instance, SysML blocks allow unifying the representation of the structure of heterogeneous systems but

have weak semantics, in particular in terms of behavior. As another example, the specification of timing aspects is considered out of scope of SysML and must be provided by another profile. The consequence is a risk of discrepancies between profiles which have been developed separately. SysML can be customized to model domain specific applications, and in particular support code generation towards SoC languages. First signs of interest in this direction are already visible [21, 33, 49, 59, 64].

SysML allows integrating heterogeneous domains in a unified model at a high abstraction level. In the context of SoC design, the ability to navigate through the system architecture both horizontally (inside the system at a given abstraction level) and vertically (through the abstraction levels) is of major importance. The semantic integrity of the model of a heterogeneous SoC could be ensured if tools supporting SysML take advantage of the allocation concept in SysML and provide facilities to navigate through the different abstraction layers into the underlying structure and functionality of the system. Unfortunately, such tool support is not yet available at the time of this writing.

## *3.3 UML Profile for MARTE*

The development of the UML profile for MARTE (Modeling and Analysis of Real-Time and Embedded systems) was initiated by the ProMARTE partners in 2005. The specification was adopted by OMG in 2007 and has been finalized in 2009 [43]. The general purpose of MARTE is to define foundations for the modeling and analysis of real-time embedded systems (RTES) including hardware aspects. MARTE is meant to replace the UML profile for Schedulability, Performance and Time (SPT) and to be compatible with the QoS and SysML profile, as conceptual overlaps may exist.

MARTE is a complex profile with various packages in the areas of core elements, design, and analysis with a strong focus on generic hardware/software component models and schedulability and performance analysis (Fig. 2.6). The profile is structured around two directions: the modeling of features of real-time and embedded systems, and the annotation of the application models in order to support the analysis of the system properties.

The types introduced to model hardware resources are more relevant for multi-chip board level designs rather than for chip development. The application of MARTE to SystemC models is not investigated, so that MARTE is complimentary to the UML profile for SoC. MARTE is a broad profile and its relationship to the RTES domain is similar to the one between UML and the system and software domain: MARTE paves the way for a family of specification formalisms.

## *3.4 UML Profile for IP-XACT*

IP-XACT was created by the SPIRIT Consortium as an XML-based standard data format for describing and handling intellectual property that enables automated con-