



High level modeling of heterogeneous systems, analog/digital interfacing.

Fabio Cenni

► To cite this version:

Fabio Cenni. High level modeling of heterogeneous systems, analog/digital interfacing.. Other. Université de Grenoble, 2012. English. NNT : 2012GRENT009 . tel-00721972

HAL Id: tel-00721972

<https://theses.hal.science/tel-00721972>

Submitted on 31 Jul 2012

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Nano-Électronique et Nano-Techonologies**

Arrêté ministériel : 7 août 2006

Présentée par

Fabio CENNI

Thèse dirigée par **Emmanuel SIMEU**

préparée au sein du **Laboratoire “Techniques de l’Informatique et de la Microélectronique pour l’Architecture des systèmes intégrés” (TIMA) de Grenoble**

et de l’**École Doctorale d’Électronique, Électrotechnique, Automatique et Traitement du Signal (EEATS)**

Modélisation à haut niveau de systèmes hétérogènes, interfaçage analogique/numérique.

Thèse soutenue publiquement le **6 avril 2012**,
devant le jury composé de :

Mme. Nathalie JULIEN

Professeur, Université de Bretagne Sud, Lorient, Présidente

M. Christoph GRIMM

Professeur, Vienna University of Technology (TU Wien), Autriche, Rapporteur

M. Bertrand GRANADO

Professeur, ENSEA, Cergy-Pontoise, Rapporteur

M. Serge SCOTTI

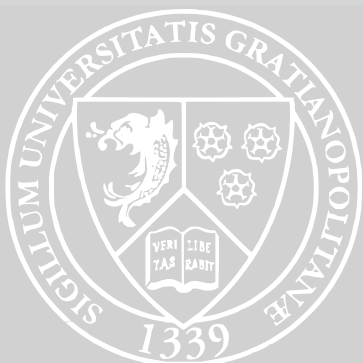
STMicroelectronics, Grenoble, Examineur

M. Laurent FESQUET

Maître de Conférences HDR, Grenoble Universités, Examineur

M. Emmanuel SIMEU

Maître de Conférences HDR, Grenoble Universités, Directeur de thèse



Abstract

The tremendous development of technology and methodology for electronic circuit design is leading to systems with high levels of complexity. Both the ever-shrinking electronic systems and improving efficiency of computer aided design (CAD) tools enable the design of extremely complex multi-tasking systems that are heterogeneous mixed-signal Systems on Chip (SoCs). The cohabitation of many physical domains such as mechanical, chemical, optical or magnetic in recent SoCs justifies the big effort that takes place in CAD tools for designing such systems. Nowadays, the design of the individual components is usually well understood and optimized through the usage of a diversity of CAD or Electronic Design Automation (EDA) tools, design languages, and data formats. These are based on applying specific modeling/abstraction concepts, description formalisms (also called Models of Computation (MoCs)) and analysis/simulation methods. The designer has to bridge the gaps between tools and methodologies using manual conversion of models and proprietary tool couplings/integrations, which is error-prone and time-consuming. The interaction among the huge quantity of individual components in recent systems is of vital interest for the overall system to function in compliancy with the specifications requested by the customer. At early stages of the design phase the interaction among the variety of Intellectual Properties (IPs) integrated on the system can only be validated through the simulation of a system model. Different levels of abstraction can be defined when modeling some IPs, a high abstraction means capturing only the gross behavior of the system and keeping a reduced accuracy of the model with respect to the real device. Analog and digital parts coexist in one system, it is necessary to model both of them for performing an overall simulation of the system. Furthermore digital parts involve processors with their embedded softwares. A crucial issue is nowadays becoming the validation of the interaction among AMS parts together with the digital and the embedded software.

For such a target a recent C++ based common design and simulation platform is establishing itself. Such a platform is based upon the SystemC (IEEE 1666) kernel and called SystemC AMS, it allows creating and refining a virtual prototype of the overall system on a high level of abstraction. This is done by supporting different description formalisms, also called Models of Computation (MoCs), for arbitrarily describing many types of behavior and interaction. This makes possible the exploration of different architecture options, estimation of the performance, validation of re-used parts, verification

of the interfaces between heterogeneous components and inter-operability with other systems.

This thesis deals with the modeling of analog and mixed-signal physically heterogeneous systems. In particular it is presented a study on different techniques for extracting behavioral models at different levels of abstraction and computational weights. Although the tools developed for behavioral model extraction are mostly based on the AMS extension of the SystemC kernel, the methodology can be applied to other Analog Hardware Description Languages (AHDL) such as Verilog-AMS and VHDL-AMS. These techniques are regrouped in three branches.

Firstly, a behavioral modeling technique from a schematic netlist entry description is studied and automatized in order to extract additional desired information under the guise of state space equations.

Secondly, techniques based on analytical fittings of frequency responses are explored for either reducing the model order or for identifying analytical simulateable models starting from analysis carried out with other application-specific CAD tools.

Finally, system identification techniques are examined for the extraction of black-box models from empirically obtained data, either from simulations of accurate models or from measure data. A proof-of-concept library implemented using SystemC AMS shows the applicability of the methodology and an LNA case study is developed for a power-minimization specific application.

The mixed nature of the Ph.D., both academic and industrial due to the collaboration with the STMicroelectronics enterprise, led to concentrate the modeling efforts on a CMOS image sensor case-study. This case-study aims at an overall simulation of an industrial platform for mobile application based on transaction-level modeling using the SystemC kernel. In order to do so, the analog/digital interfacing between the SystemC AMS MoCs and different types of SystemC-TLM coding styles is studied and adapted to the STMicroelectronics' proprietary TLM protocols as a proof of the applicability of the SystemC AMS based methodology to the industrial design flow. The image sensor model is abstracted at different levels using different SystemC AMS Models of Computation showing impressive simulation time performances with respect to the low level models (notably the VHDL-AMS based model). The virtual platform is currently being used for an early validation of the image correction algorithms and embedded software, this will result in an improved reliability of the product.

Keywords: Analog and Mixed-Signal (AMS) Design Flow and Methodology; Behavioral Modeling; AMS IP reuse; State Space Model; Reduced Order Modeling; Model of Computation (MoC); Multiphysical and AMS Systems on Chip (SoCs); OSCI SystemC AMS; VHDL-AMS; System Identification; Image Sensor; SystemC Transaction Level Modeling (TLM).

Acknowledgements

Completing a thesis is a challenge.

I'd like to express my gratitude to all the people that have remained close to me, both physically and emotionally as well, during these French years. In order to thank them all properly I desire to write in the language they respectively speak or better understand.

Comincerò col ringraziare i miei cari nella mia bellissima lingua madre, l'italiano.

Un ringraziamento speciale va alla mia meravigliosa famiglia quindi **Tonino**, **Nilde**, **Claudio**, **Sofia** e le mie due **Vittorie**. Grazie per non avermi mai fatto mancare il vostro incoraggiamento ed il vostro supporto.

Un enorme grazie alla persona che mi è stata vicina più di tutti in questi ultimi anni, la mia dolce **Valentina**.

Grazie a tutti i miei amici di vecchia data ma anche più recenti, il vostro sostegno è stato e continua ad essere essenziale.

Doveroso è un ringraziamento a tutti i membri della mia famiglia allargata i quali contribuiscono a farmi sentire a casa ogni volta che torno in Romagna.

Posso affermare con sicurezza che questa esperienza mi ha veramente permesso di crescere da tutti i punti di vista e sono grato a tutti per avermi appoggiato in questa mia scelta. Purtroppo i contatti tra di noi sono diventati sempre meno frequenti ma *non vedo l'ora di riavvicinarmi al luogo in cui sono cresciuto, in mezzo a voi tutti ed alla mia gente.*

Permettez-moi maintenant de passer au français pour adresser mes remerciements les plus sincères à ceux qui ont rendu possible l'achèvement de mes travaux de thèse.

Un remerciement spécial va à mon directeur de thèse **Emmanuel** qui a toujours eu confiance en moi, me donnant ainsi la force et les motivations pour creuser et m'investir dans la recherche.

Un énorme merci va à **Serge**, mon tuteur en entreprise, pour m'avoir suivi et avoir guidé mon intégration dans ce monde énorme qui est STMicroelectronics. Son aide sous plusieurs aspects, tels que le relationnel aussi bien que le technique, a été d'incalculable valeur.

Je tiens à remercier les membres du jury qui ont bien voulu accepter de valoriser ce travail.

Mes remerciements s'adressent également aux acteurs de ma "deuxième vie".

Je tiens tout d'abord à remercier les amis Grenoblois ou naturalisés Grenoblois qui m'ont accompagné pendant ces années de découverte et d'ouverture d'esprit. Notamment depuis les temps internationaux rabotins je remercie Alejandro, Cécile et tous ceux qui ont participé à mon intégration initiale, en passant par mes amis italiens Alessandro, Adriano, Valentina.

Un mot aussi pour remercier les copains français, italiens et espagnols dont certains ont le plaisir de se confronter avec moi lors des matchs de foot en salle.

Les amis et collègues de l'ancienne et mémorable équipe RMS et du Laboratoire TIMA: Stephane, Laurent B., Ke, Asma, Rafik, Nourredine, Rshdee, Yoann, Louay, Mathieu, Franck, Maxime, Laurent F. j'espère de vous avoir tous cités.

En restant dans TIMA je voudrais remercier de tout cœur les supérieurs Salvador, Libor, Haralampos.

Merci aux protagonistes de ma vie industrielle à STMicroelectronics. Un gros merci aux collègues directs, mais surtout amis, Jean-Michel et Romain pour tout ce qu'on a passé ensemble et de leur sympathie.

Merci aux collègues de l'équipe SPG de ST: Maxime et Laurent, de l'équipe Imaging de ST: Stephane et Giuseppe et à ceux de ST-Ericsson: Philippe, Frédérique, et Karine.

Un gros merci à Nicola, Dario, Ioannis et Massimo pour l'amitié qui s'est installé entre nous renforcée par les réflexions parfois délirantes lors des repas à ST.

Of the many people who deserve thanks, I would like to say a word about the partners of the Beyond-DREAMS European Project. A special thank to the following for the nice and precious collaboration: Marie-Minerve, François and Antoine from LIP6, Paris; Martin from NXP, The Netherlands; Peter from Robert BOSCH, Germany; Karsten from Fraunhofer IIS/EAS, Germany and Gert-Jan from Dizain-Sync, The Netherlands.

If I have forgotten anyone, I apologize... Thank you all.



Contents

I. Positioning the context	3
1. Introduction	5
1.1. Introduction to physically heterogeneous systems and design methodologies . .	5
1.2. Motivation and research contributions	6
1.2.1. Motivation	6
1.2.2. Research contribution	9
1.3. Thesis structure	11
II. State-of-the-art of modeling-based design of heterogeneous systems	13
2. Design of heterogeneous embedded systems	15
2.1. Heterogeneities in embedded systems	15
2.1.1. Hardware and software heterogeneity	15
2.1.2. Heterogeneity of the architecture	15
2.1.3. Multi-physics heterogeneity	16
2.1.4. Electrical (digital/AMS/RF) heterogeneity	16
2.1.5. Summary and terminology	17
2.2. Design flow of heterogeneous mixed-signal systems	17
2.2.1. MEMS/AMS/RF design flow design flow	18
2.2.2. Currently open issues and conclusions	19
2.3. Modeling-based refinement of heterogeneous systems	20
2.4. State of the art of mixed (analog-digital) modeling languages/tools	21
2.4.1. Application of modeling languages to hardware design	22
2.5. Objective of the thesis	24
3. SystemC based environment for virtual prototyping	27
3.1. State of the art of AMS extensions to System C	27
3.2. SystemC AMS 1.0 OSCI standard model abstractions	32
3.3. SystemC AMS 1.0 OSCI standard models of computation	33
3.3.1. TDF modeling fundamentals	34
3.3.2. ELN modeling fundamentals	35
3.3.3. LSF modeling fundamentals	35
3.4. Interaction among SystemC AMS models of computation	36
3.4.1. Continuous-time (ELN or LSF) to/from discrete-time (TDF) or discrete-event (DE) domains	37
3.4.2. Continuous-time conservative to/from non-conservative domain (ELN/LSF)	38
3.4.3. Discrete-time domain to/from discrete event SystemC domain (TDF/DE)	38

3.5. SystemC TLM	39
3.6. Interaction between SystemC AMS models of computation and SystemC-TLM 2.0	41
3.6.1. Interfacing TDF/TLM2.0 AT	42
3.6.2. Interfacing TDF/TLM2.0 LT	42
III. The contribution	43
4. Specification and implementation of SystemC AMS extension libraries	45
4.1. Introduction to high-level modeling of AMS multi-physics systems	45
4.2. Contribution to the elaboration of knowledge-based high-level models from a netlist descriptive view	48
4.2.1. From the netlist to the state space equations with SystemC AMS	49
4.3. Contribution to the elaboration of knowledge-based models from simulation data	52
4.3.1. Modeling from frequency domain response curves for dynamic components	52
4.3.2. Macro-modeling of static components for SystemC AMS simulations . .	57
4.4. Contribution to the elaboration of black-box models from empirical data	58
4.4.1. System identification, model structures, parameters and criteria.	59
4.4.2. Proposed extension of SystemC AMS libraries for building identification models	63
4.5. Application of system identification techniques to the closed-loop control for power consumption optimization	67
4.5.1. Behavioral input/output model	68
4.5.2. LNA : Low Noise amplifier	70
4.5.3. Envelope Detector	71
4.5.4. Nonlinear performance prediction model	72
4.5.5. Adaptive logical control	73
4.5.6. LNA performance modes	74
4.5.7. Simulation results	75
4.5.8. Conclusions	79
4.6. SAW based chemical sensor case study	80
4.6.1. Overview of the SAW device	81
4.6.2. Behavioral modeling of SAW sensors	81
4.6.3. Laplace transfer function model of the SAW sensor	83
4.6.4. Low-level Verilog-A / Cadence-Spectre based modeling	85
4.6.5. High-level SystemC / SystemC AMS based modeling	97
4.6.6. Conclusions	101
4.7. Summary	102
5. Industrial case study: CMOS video sensor	105
5.1. CMOS video sensor and SystemC AMS models	105
5.1.1. VHDL-AMS model	106
5.1.2. SystemC AMS ELN-TDF model	107
5.1.3. SystemC AMS TDF models	109
5.2. SystemC AMS TDF fastest model	111
5.2.1. Input image builder	111
5.2.2. Lens effect model	112
5.2.3. Bayer filter model	113

5.2.4. Video timer model	114
5.2.5. Pixel module	115
5.2.6. ADC bank module	116
5.3. CIS modeling styles performance comparison	118
5.4. CIS model integration in different SystemC-based platforms	119
5.5. Beyond-DREAMS OSCI SystemC TLM 2.0 platform integration	120
5.5.1. SystemC TLM 2.0 proof-of-concept platform	120
5.5.2. SystemC AMS/SystemC OSCI TLM 2.0 platform simulation results . .	123
5.6. WASABI SystemC bit-cycle accurate platform integration	123
5.6.1. Modeling the whole system	125
5.6.2. Modeling of the SystemC / SystemC AMS interfacing	126
5.7. STMicroelectronics CATSEYE SystemC TLM platform integration	128
5.7.1. Digital to analog control	129
5.7.2. Analog to digital information passing	129
5.8. ST-Ericsson SystemC TLM mobile platform integration	130
5.9. Conclusion and future works	133
6. Conclusions and perspectives	135
A. Technical Annex	139
A.1. SytemC TDF based system identification	139
A.2. SystemC AMS extraction of a Laplace transfer function-based fitted model . .	149
A.3. C-code for the MIPS32 embedded firmware	153
List of Figures	157
List of Tables	161
List of Acronyms	163
Bibliography	167
 IV. Résumé de la thèse en français	 177
Résumé	179
Introduction et état de l'art	181
1. Contributions à la recherche	184
2. Structure de la thèse	186
Environnement de modélisation et simulation SystemC	187
3. SystemC AMS et ses modèles de calcul	187
4. SystemC TLM	188
Contribution à la modélisation de haut niveau de systèmes hétérogènes	189
5. Introduction et regroupement à partir des connaissances de départ	189
6. Contribution à l'élaboration de modèles de haut niveau basés sur la connaissance à partir d'une netlist	191

7.	Contribution à l'élaboration de modèles basés sur la connaissance à partir de données de simulation	192
7.1.	Macro modélisation de composants statiques pour la simulation en SystemC AMS	192
7.2.	Macro modélisation de composants dynamiques pour la simulation en SystemC AMS	192
8.	Contribution à l'élaboration de modèles boîte noire à partir de données empiriques	193
9.	Application des techniques d'identification de système à la commande en boucle fermée pour l'optimisation de consommation de puissance. Cas d'étude d'un LNA	193
10.	Cas d'étude de la conception de l'interface microélectronique pour un capteur chimique SAW	194
11.	Conclusions	195
Cas d'étude industriel : capteur d'images CMOS		197
12.	Motivation	197
13.	Modélisation du CIS	197
14.	Comparaison de performances entre les modèles du CIS	199
Conclusions		201
Author's curriculum vitæ		205

Part I.

Positioning the context

Chapter 1.

Introduction

1.1. Introduction to physically heterogeneous systems and design methodologies

The tremendous development of the last decades in terms of technology and methodology for electronic circuit design is leading to systems with high levels of complexity. The shrinking of electronic systems from the point of view of the technology node together with the efficiency of the computer aided design (CAD) tools allow to design and conceive extremely complex multi-tasking systems in little area.

On the one hand, from the electronics point of view, the system composition is more and more split in two parts devoted to two different tasks: the thinking part and the interacting one. The *thinking* part is typically the core of the electronics of the system with the highest fill factor, that is, it contains the most part of electronic devices, consumes the smallest part of the power and occupies the smallest percentage of the system area. This is obtained thanks to the small quantity of energy exchange taking place among the different devices of the thinking part. The *interacting* part is consecrated to the interfacing with the real world, which means sensing and controlling real physical quantities. The interfacing with digital world requires sensors and actuators dealing with analog signals therefore digital to/from analog converters are needed. When dealing with physical quantities it is well known that, the more impacting effect we desire the higher the energy involved hence larger area is used.

On the other hand, the research on materials and other scientific fields allowed developing miniaturized sensors and actuators in a variety of physical domains (mechanical, chemical, optical, magnetic). Many multi-domain sensors are embedded in nowadays systems thus opening the way to different physical natures integrated in one chip: physically heterogeneous systems on chip (SoCs). When it is not feasible to fabricate a SoC for a particular application, typically because of the heterogeneity of the technological process, an alternative is a system in package (SiP) that comprises a number of dies in a single package. Examples of the aforementioned physically heterogeneous systems can be found in different market shares such as the automotive with sensor networks based on micro electro-mechanical systems (MEMSs) mainly used for mechanical sensor, or the pharmaceuticals with bio-analysis autonomously performed by devices called “laboratories on chip” (lab-on-a-chip or LOC), or the consumer electronics with the market-pulling embedded devices like smartphones and tablet personal computers. Figure 1.1 shows the case of a technology push (mainly driven by internal R&D activities) and a market pull (driven by external forces) as two distinguished phenomena, it is intentionally left to the reader’s consideration whether the interaction exists and, if it is the case, how interwoven it is.

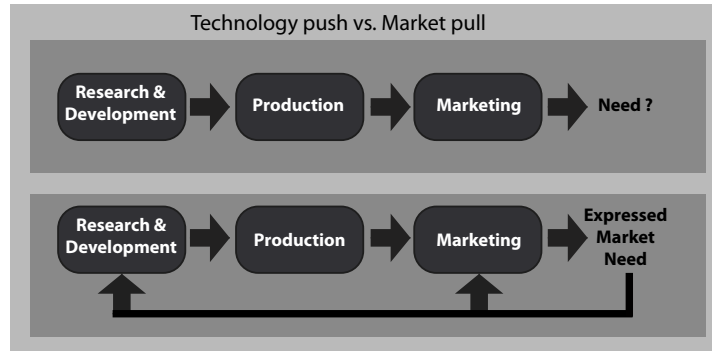


Figure 1.1.: Technology push and market pull.

Resuming, increasing **complexity** and both significant **multi physics domain heterogeneity** and **analog/digital heterogeneity** of recent SoCs and SiPs are accompanied by other open issues given by advances in manufacturing technology and semiconductor market dynamism. These issues are: increasing **environment awareness** for low power devices typically battery-supplied or harvesting of environmental energy; increasing **re-use of subsystems** for reducing the design efforts thus reducing the time to market; increasing impact of **modern silicon technologies** due to the physical effects taking place in ever shrinking transistor size.

1.2. Motivation and research contributions

1.2.1. Motivation

The design flow of nowadays multi-domain mixed-signal systems is scattered among different design methodologies supported by diverse computer-aided design (CAD) and electronic design automation (EDA) tools. The design of heterogeneous systems is still a highly manual work and not as automatized / standardized as the digital design flow. An intuitive and partial explanation to this delay of analog on the digital is that AMS systems are concerned by a high variety of physical phenomena that have to be interpreted and modeled for being understood and therefore mastered, while digital systems are artificial artifacts thus easily formalizable. Logic synthesis and place & route tools assist the design of digital systems while heterogeneous design requires a multidisciplinary approach. A multidisciplinary approach requires the definition of different description formalisms also called models of computation (MoCs) together with analysis and simulation methods. Such formalisms/analysis and simulation methods are provided by CAD/EDA tool vendors and consist of specialized simulators for different physical disciplines and levels of abstraction together with other capabilities (netlist, graphical user interface, etc.).

For example the design of a typical micro-electro-mechanical system like a three-axial accelerometer [Tan 08] requires:

- Optimization and characterization of the micro-mechanical resonator and (separately) the electro-static field distribution of the comb drive structures, which are driving and sensing the movements of the flexible structure. This is done with the help of a Finite Element Analysis tool (e.g. ANSYSTM, COVENTORTM) from mechanical engineering.

- Simulation of the whole system on the circuit level, taking into account the coupling between mechanical and electrostatic domain within the MEMS transducer and feedback from the analog and digital driving and sensing circuits. For this, behavioral simulators and modeling languages like VHDL-AMS from electrical engineering are employed.
- Layout of the mechanical structure and of the electronic circuits. This is carried out with the help of IC layout tools.

On the one hand, when an **accurate heterogeneous simulation** is needed, a co-simulation, defined as a simulation performed by launching two different simulation kernels, allows interfacing the variety of domain-specific tools and simulators above listed, thus allowing the overall heterogeneous simulation. However, this process is far to be seamless and straightforward. It is not the CAD tool vendors' priority to provide easily interfaceability to other tiers simulator, especially if it is the case of competitors in advance in other engineering fields or if their suites of EDA tools already feature a similar proprietary capability to the user, even if this capability is not enough. The designers are forced to bridge the gaps between tools and methodologies using manual conversion of models for achieving proprietary tool coupling and tool integrations. Obviously these stratagems are error-prone and often unreliable because of the high subjectivity of their outcome. As short-term axe of research, efforts in the direction of an easy and user-transparent interaction between simulators should be put. In the long term, new design methods and integrated tool chains are needed to support the whole process of specification, design, integration, verification and validation of the components of a complex AMS system.

On the other hand, going hand in hand with the increasing complexity of nowadays systems it is the capability of performing an efficient **overall system verification** early in the design flow. Virtual prototypes of the components of the system, both the digital and the heterogeneous parts, will allow performing the verification of the system, which would provide many benefits such as architecture exploration, performance estimation, validation of reused parts, verification of the interfaces between RF, analog and digital parts, early verification of the embedded software development together with its eventual debugging, verification of the interoperability with other systems, and assessment of the impact of future working environments and new generations of technologies. The task of modeling is therefore crucial in the design flow of embedded electronics and modeling languages must be appropriately chosen mainly basing on the type of sub-systems it is needed to model and at what level of accuracy / abstraction.

From the **digital side**, the top-down design flow from Register Transfer Level (RTL) descriptions to gate-level implementation by using the VHDL [IEEE 09c] and Verilog [IEEE 04] Hardware Description Languages (HDLs) is pretty well established and used by the majority of digital hardware designers. The relevance of increasing portions of embedded software in recent systems led to the need for modeling it, hence pushing the research towards system-level design languages. SystemC [IEEE 09a] and SystemVerilog [IEEE 09b] certainly are widely affirmed for providing the system-level view. SystemC is an industry-standard language for electronic system-level (ESL) design based on the C++ programming language, it is promoted by the Open SystemC Initiative (OSCI) (from December 2011 OSCI and *Accellera* merged and became *Accellera Systems Initiative* [Initiative 12]) and is an IEEE standard (known as IEEE 1666-2005) since 2005. It allows describing systems at more abstract level (but even at the RTL level) basing on the discrete event driven kernel by giving the possibility to model both the hardware and the software components. In the recent years the C language surrounded by all

the C-like dialects has established itself as the solution for developing the software/firmware of embedded processors, it follows that system level architectural descriptions of such processor should be described by means of the C language. It is not the case for SystemVerilog for example, where the programming languages used for the architectural description and the embedded software are not both based on the C language. In 2008 OSCI also released the SystemC Transaction-level Modeling Standard, TLM-2.0 [OSCI 09] aimed at enabling SystemC model interoperability and reuse at the transaction level, providing an essential ESL framework for architecture analysis, software development, software performance analysis, and hardware verification.

When it comes to the **analog and mixed signal** concurrent design such as RF, MEMS, analog to digital (ADC) or digital to analog conversions (DAC), extensions to the existing modeling languages have appeared. VHDL and Verilog have shown their AMS extensions named VHDL-AMS (standard IEEE 1076-2008)[IEEE 07] and Verilog-AMS with its Verilog-A analog kernel [Accellera 09]. These languages are really hardware description oriented and lacks in modeling formalisms for describing AMS systems at a behavioral or functional level. Furthermore they offer no capability of modeling for embedded software and available simulators are too slow for even think to simulate a complex state-of-the-art heterogeneous SoC. Other high level tools like Matlab/Simulink do offer an alternative for functional and behavioral modeling but lack in direct connection with the flow to the hardware implementation and there is no way to simulate the SW. It has been logical to provide the SystemC framework with AMS extension since, it is being widely accepted and adopted as a versatile C++ based language commonly used by system designers, software engineers, and hardware designers. The AMS working group (AMSWG) of the OSCI has being involving its efforts in the definition of modeling paradigms, syntax and interfacing layer to the SystemC event-driven kernel, together with the submission for standardization of the SystemC AMS [OSCI 10, Vachoux 05] extensions since a few years now. This simulation language is now able to model complex heterogeneous systems with the help of different MoCs for a description of each individual component.

Behavioral modeling of analog and mixed-signal such as RF or MEMS components is more and more topical in the industry as part of the design process of integrated systems, since it allows simulating the entirety of a complex heterogeneous system. The above-mentioned standardized mixed-signal and mixed-technology behavioral HDLs are facilitated by several commercial and open source simulators. A behavioral model is the description of a component as input-output behavior augmented with major non-idealities of real implementations, but without requiring a complete description of all implementation details. The purpose is mostly to verify the correct functionality of the system wherein the device is operating in acceptable CPU times by replacing the target devices with behavioral models.

It is of common sense to consider the design process of technological systems as a V-shaped model (Figure 1.2) wherein the design has a top-down approach starting from the business case to the implementation passing through the system specifications. The verification has a bottom-up approach because it is performed step by step first verifying the functioning of the device inserted in the sub-subsystem then verifying the sub-subsystem's functioning inserted in the subsystem and so on, while keeping back-annotating information for reviewing/modifying the design at any time an issue is found out.

Behavioral modeling can greatly simplify the design process in both sides of the V-model, however, a wrong choice of the model correctness/accuracy can also lead to oversimplification, messing up the whole process:

- **During top-down design.** Here the intent is not really to design in the classic acceptance of the word, that is, start from system specifications and descend with the automated synthesis to the layout of the system, although some studies are ongoing in this direction [der Plas 02]. The intent is rather to take advantage of behavioral modeling of digital and AMS components of the system for helping the designers to perform architectural exploration and to map top-level performance specifications onto different blocks trading off different performances and implementation costs (e.g. area, power).
- **During bottom-up verification** the same behavioral models used for refining the system in sub-systems and components can be used for verify the correct functioning of the system. For representing a real interest the model accuracy is improved by enriching them through details coming from the low-level implementation. The low-level implementation could be a transistor-level design before the layout thus the extraction of parasitic, or an already back-ended chip together with simulations including parasitic effects, or a characterization through measures performed on a physical prototype. Of course details fall apart when gradually raising the level of abstraction.

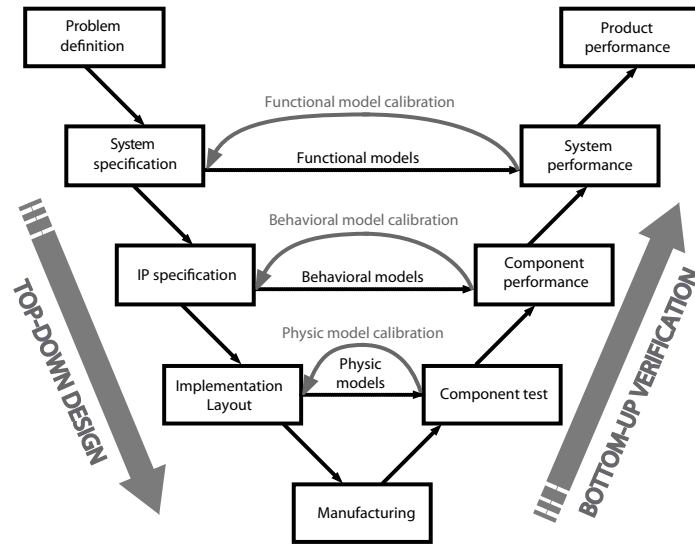


Figure 1.2.: V-model of the design process of a technological system.

Different techniques and strategies for modeling such components can be found depending on the actual need linked to the position in the V-model of the product design (Figure 1.2).

1.2.2. Research contribution

When dealing with the diversity of natures and domains involved in heterogeneous SoCs, it is hard to figure out how analog behaviors can be captured and modeled for representing the functioning of the target sub-systems or components. The modeling formalisms / paradigms that are available change for each modeling language or framework. Different techniques for

different types of behavioral modeling have been studied and qualified as modeling techniques for different requirements. Analog and mixed signal components may be mastered at different levels of knowledge, it is typically reasonable to resume two cases as follows:

- The structure of the device is known and the physical laws that govern the behavior of the device are mastered. An analytical inputs/outputs equation, sometimes involving state variables, can therefore be determined. This is called “modeling of knowledge” (violet left branch in Figure 1.3), the possibility of disposing of a system exhaustive knowledge is nowadays less and less probable because of the increasing complexity/coupling of the devices.
- The physical laws that define the behavior of the device are not *a priori* known and only experimental data are available. The device is either physically available (green right branch in Figure 1.3) (delivered in the form of a prototype IP or a pre-compiled simulateable model, in both cases, it is considered as a black box) or a fine description of it (blue center branch in Figure 1.3) issued from structural analysis performed by *ad-hoc* domain-specific simulator/tool (FEM for instance) is known. In both cases a mathematical model has to be built for describing the behavior of the device in order to perform simulations.

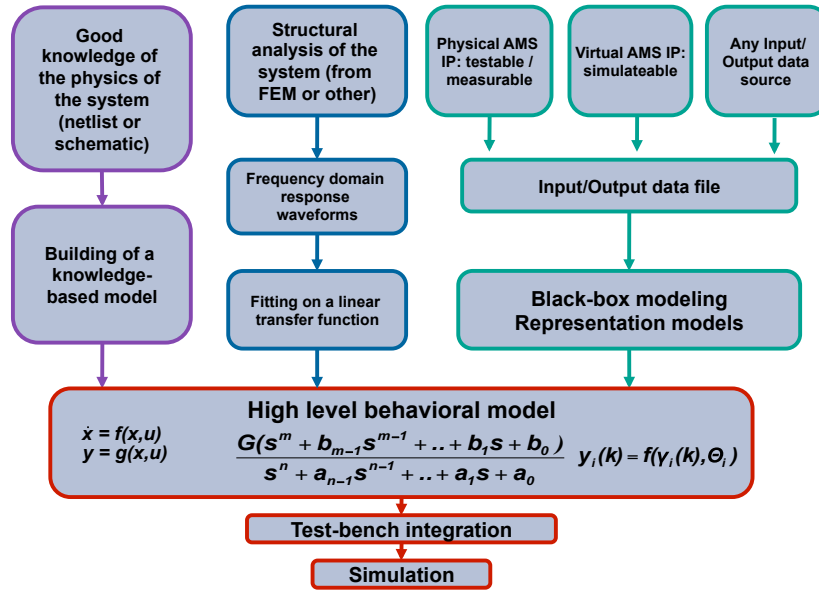


Figure 1.3.: Behavioral modeling cases for heterogeneous mixed-signal devices.

Different techniques for the modeling have been analyzed and the constructs automatized for obtaining behavioral models at different levels of abstraction and computational weights, with particular focus on the SystemC AMS analog and mixed signal extension framework. For obtaining this, many reduced order modeling techniques are studied and implemented for the refinement of the model-based top-down design or bottom-up verification. Different case studies and different models of computation offered by SystemC AMS (but also Verilog-AMS and VHDL-AMS), are analyzed for proving the efficiency of the proposed methodology. The analog / digital interfacing issue is also faced and application-specific solutions are shown for the interfacing to the SystemC OSCI transaction level modeling TLM-2.0 standard. In particular,

taking advantage of the industrial collaboration with STMicroelectronics, the methodology for AMS components is applied to refined models of a CMOS video sensor. The AMS sensor itself communicates with transaction-level communication protocols proprietary of STMicroelectronics. It is claimed and proved that the methodology would aid in both shrinking the time to market by anticipating development phases (embedded software development/debug among others) and in augmenting the robustness of commercialized products by reducing their bug-proneness.

1.3. Thesis structure

Chapter one has been introducing the framework and the motivations of this work. Chapter two will first give a brief overview of the different meaning of the word heterogeneity that appears when dealing with embedded systems. Chapter two will then show the typical industrial flow of heterogeneous embedded system, in the acceptance of electrical heterogeneity, the problems and the techniques for modeling and simulation of heterogeneous systems. The objective of the thesis will be defined here. Chapter three will deal with the C++ based SystemC framework and its extensions / formalization for analog and mixed signal and transaction level modeling. The SystemC AMS models of computation and their interaction will be detailed together with the interaction with the digital transaction level modeling. Chapter four gives my contribution to the study of modeling techniques for abstracting reduced-order, higher-level models from the circuit or transistor level both for design and test purposes. In Chapter four an application of the *system identification* technique will aim at the closed-loop control of a target device for a power consumption optimization, a Low Noise Amplifier is shown as a case-study. Chapter four will also show how the behavioral modeling is applied in the case-study of the design of a Surface Acoustic Wave-based chemical sensor. Chapter five will deal with the embedded analog and mixed-signal industrial case study, the CMOS image sensor. Chapter six will then give the conclusions of my work and future work perspectives.

Part II.

State-of-the-art of modeling-based design of heterogeneous systems

Chapter 2.

Design of heterogeneous embedded systems

In this chapter a description of the design flow for heterogeneous systems is given. First, when talking about heterogeneous embedded systems many thoughts could come to one's mind. In section 2.1 an overview of the main kinds of heterogeneity is provided. The focus will then move on the heterogeneity of multi-physical domains and the heterogeneity between analog and mixed-signal systems and purely digital systems. The design flow typically used for analog and mixed-signal systems is described in section 2.2 together with its issues and limitations. Section 2.3 deals with the techniques for modeling and refinement for the model-based design of heterogeneous systems. Section 2.4 gives an overview of the state-of-the-art modeling languages and tools for mixed-signal systems. Finally the last section reminds the objective of the thesis.

2.1. Heterogeneities in embedded systems

The heterogeneity of embedded systems may be highlighted/schematized by illustrating the overall system as a block diagram. The blocks are used for distinguishing the fields/domains that induced us to define the system as heterogeneous.

2.1.1. Hardware and software heterogeneity

This kind of heterogeneity is intended when, within a system, the architecture has to be designed by considering if the tasks have to be performed by a dedicated hardware module or by loading/running the related algorithms on a standard processor. It is an intrinsic aspect of every state-of-the-art electronic system. In the particular subject of embedded systems, special care must be dedicated to the partitioning of hardware and software parts with the intention to use Application-Specific Integrated Circuit (ASIC) parts whether high performances are required [Wolf 94], techniques for the HW-SW co-design can be found in literature and nowadays tools and design methodologies allow developing concurrently both the HW and the SW rather than developing the SW once the HW is available.

2.1.2. Heterogeneity of the architecture

This kind of heterogeneity refers to the architecture of processors from the point of view of the digital/hardware architect. Node level heterogeneous architectures (also known as *heterogeneous computing*) refers to the use of different processing cores to maximize performance

[Brodtkorb 10]. Compared to traditional symmetric multi-processor (SMP) systems, they offer high peak performance and are energy and/or cost efficient. The use/study of these architectures is indeed a hot topic due to deployment of fine-grained parallelism in high-performance computing, as well as the introduction of parallelism in workstations. Typically, three commonly found architectures are predominant, Figure 2.1 shows the compositions of the three architectures:

- Cell Broadband Engine Architecture (CBEA), a traditional CPU core and eight single instruction multiple data (SIMD) acceleration cores.
- Architecture based on graphics processing units (GPUs), a standard multiple core CPU combined with a GPU with 30 highly multi-threaded SIMD accelerator cores.
- Architecture based on field programmable gate arrays (FPGAs), a standard multiple core CPU combined with an FPGA containing an array of logic blocks.

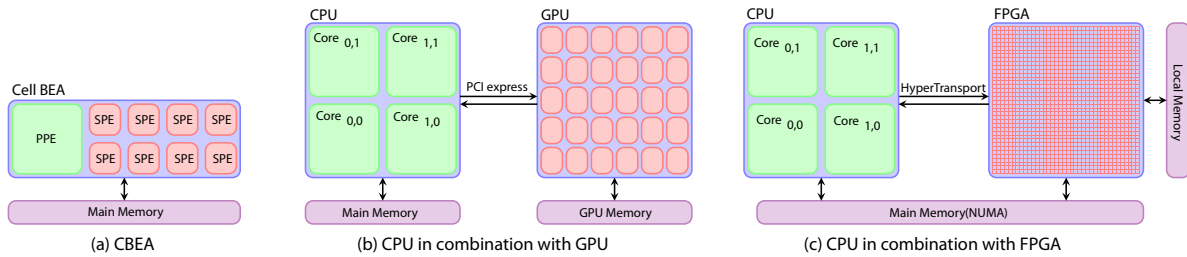


Figure 2.1.: The Cell Broadband Engine is a heterogeneous chip (a), a CPU in combination with a GPU is a heterogeneous system (b), and a CPU in combination with an FPGA is also a heterogeneous system (c).

2.1.3. Multi-physics heterogeneity

In nowadays systems, the coexistence of multiple physical domains or multiple simultaneous physical phenomena (electromagnetism, chemistry, biology, mechanics, fluidics, thermo-dynamics, optics) on the same system is a reality, this aspect is emphasized in embedded systems since, compared to workstations, they have to interact with the real world by sensing physical quantities by means of sensors and acting by controlling actuators/motors. In addition to the meaning: **more than one physical domain** (electrical, mechanical, optical, chemical ...), the adjective “heterogeneous” may also be used for identifying **more than one technological fabrication process** that is different basic materials (silicon, III-V, organic ...) or different co-integration techniques: planar or stacked SoC or bonding (SiP) [O’Connor 07].

2.1.4. Electrical (digital/AMS/RF) heterogeneity

This heterogeneity refers to the coexistence of purely digital parts, analog and mixed-signal (AMS) parts and radio frequency (RF) signals, in the same SoC. In such systems the heterogeneity is given by the electrical coexistence of purely analog systems with analog inputs/outputs (amplifiers e.g. operational amplifiers, analog filters, or RF devices such as low noise amplifiers or mixers), together with mixed-signal systems where digital inputs influence the outputs

(sensors or actuators) and purely digital synchronous (internal functioning triggered by clock edges) or asynchronous.

2.1.5. Summary and terminology

The listing of types of heterogeneity just depicted is a first classification from the point of view of the system composition. If one wants to consider also other heterogeneities related to the design flow of such systems other senses of heterogeneity can be found:

- heterogeneity of the design processes involved before the object exists physically (specification, synthesis, simulation, verification).
- heterogeneity of the types of signal description (continuous and discrete time and value, cycle and bit accuracy).
- heterogeneity of the models of computation (dataflow, sequential processes, discrete event, etc.).

In this thesis it will be focused on “**heterogeneous systems**” (in the sense described in section 2.1.3) referring to different physical domains involved. Analogously, it will also be focused on “**mixed-signal (AMS) systems**” (in the sense described in section 2.1.4) referring to different electrical natures involved.

Heterogeneous systems, according to the definition given right above, do not explicitly imply that both analog and digital signals are present in the system. Despite this, on the one hand, it is intuitive that node level SoCs contain a substantial part of digital electronics for processing data and taking decisions. On the other hand, when retrieving information from other physical domains, such as a vibration in the case of a MEMS, it is almost obliged to pass through an analog signal that will be converted to a digital data by means of an Analog to Digital Converter (ADC). Resuming, for a matter of clarity, in our scope, an heterogeneous system only defines multi-physics phenomena occurring in it even if, in node level embedded systems, electrically mixed-signals are implicitly involved.

The aim of this work is to deal with modeling and simulation of these two types of heterogeneities, for instance, when a system presents both electrical and multi-physics heterogeneity it will be called “**heterogeneous mixed-signal system**” or “**heterogeneous AMS system**”.

2.2. Design flow of heterogeneous mixed-signal systems

In this section the typical design flow of heterogeneous mixed-signal systems is discussed.

In the following it is considered the case of a MEMS-based wireless sensor network (WSN) node as a good representative of a heterogeneous mixed-signal system. First, the overall specifications are defined and a partitioning in functional units (analog, RF, digital, software) is done. Second, the functional units are refined by system architects, basing on their experience with similar products, providing a tentative architecture of each functional units taking care of respecting the required performances. The overall specifications are then mapped on the macro-blocks that compose the architecture. Different designer teams take care separately of

the design flows of the digital hardware and software, AMS and MEMS parts, each one with its specifications.

A possible scenario could be the one of Figure 2.2 where the design flow of a WSN node designed and commercialized by a MEMS manufacturer is shown. For making the analyzed case as generic as possible it is assumed that third-party manufacturer are involved. In particular, *Manufacturer1* provides the digital HW and teams A and B are teams following different flows, A for the design and B for the verification. *Manufacturer2* provides the RF transceiver and analog-to-digital converters, as for the previous manufacturer, team A is in charge of the design and B of the the verification. For what concerns the MEMS design/verification, they are internally performed by team A and B while the software team develops the embedded software for the third-party hardware. The integration is then performed by assembly engineers that are in charge of the integration and verification of the functioning pre/post layout.

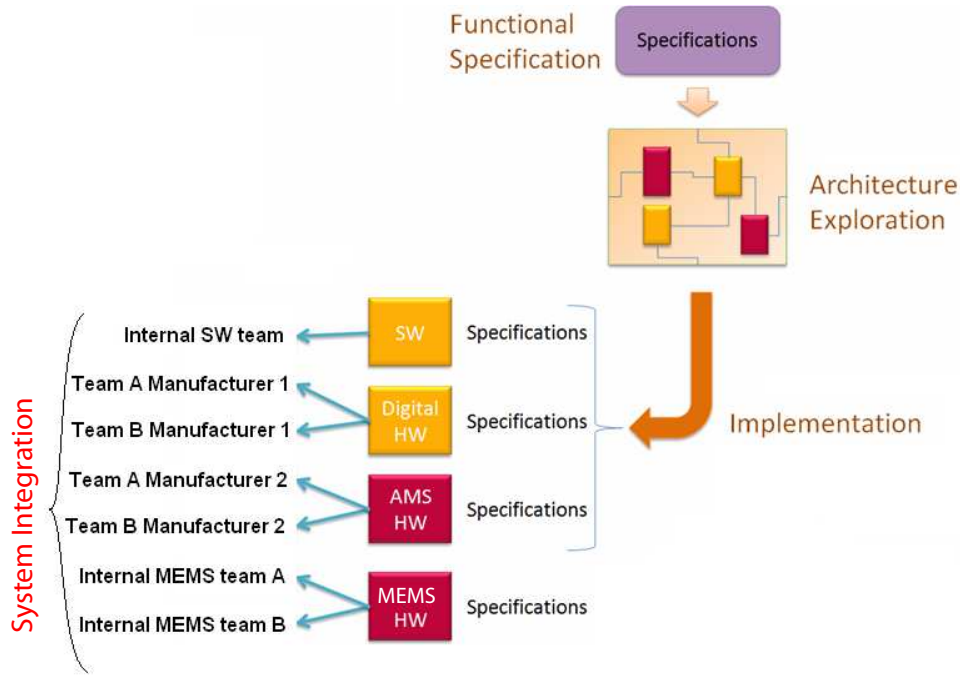


Figure 2.2.: Many flows co-existing for the design flow of WSN node.

2.2.1. MEMS/AMS/RF design flow design flow

For the MEMS part, an accelerometer for instance, basing on the required specifications, if currently available devices do not meet the requirements, a device architecture/typology is chosen among the variety of sensors produced by the MEMS provider, and modifications are necessary. A new MEMS hardware is needed and traditionally two engineering teams collaborate to the design of MEMS. One team uses a Finite Element Analysis (FEA) based computer-aided design (CAD) tool (such as CoventorWareTM[Khankhua 11] ANSYSTM[Malik 08]) to create the electro-mechanical model. The other team focuses on the electronics frontend and uses an EDA tool from electronic CAD vendors (such as CadenceTM, MentorGraphicsTM or SynopsysTM) for performing transistor-level simulations. For verifying the interfacing of both the mechanical and the electrical aspects, behavioral models of the MEMS are developed

using Analog Hardware Description Languages (AHDLs) (such as VHDL-AMS) extracting the behavior from the accurate FEM characterization. This allows to take into account the coupling between mechanical and electrostatic domain occurring within the MEMS and the analog and digital driving and sensing circuits. The layout of the mechanical structure and the electronics circuit is carried out through the help of IC layout tools. Different tools are required and it is difficult to optimize the performances because of the disparities of tools. This makes existing MEMS design-methodologies inefficient and leads to extensive and time-consuming design iterations.

For the AMS/RF part, it is almost the same flow as for the MEMS part but FEM tools are replaced with special AMS/RF simulators such as SPICE or Agilent ADSTM. Analog HDLs are nevertheless used for performing simulations of the surrounding electronics with lower details that, using transistor level simulators, would have demanded too high simulation times.

2.2.2. Currently open issues and conclusions

The above-described flow is well established but unlike the design flow of digital parts it is not seamless and straightforward, the centralization of the information is lost as soon as the design flow is separated in many/non-communicating discipline-specific (AMS/RF/MEMS) design flows. From this point of view a way to unify the exchange of specifications between the architecture and the implementation level should be introduced.

With this in mind, another point is that the main lack of the design methodology is the ability to validate the system at the application level through simulations for embedded heterogeneous AMS systems. A high level modeling language and methodology is needed for enabling the **modeling** of the digital HW and SW and the analog with dedicated models of computations (MoC).

In phase of top-down design, in order to improve the quality and speed by allowing architects to analyze several scenarios exploring solutions and early performances (architecture exploration), it is proposed to use reliable high level models of sub-blocks (provided by the MEMS sensor team for instance) for simulating the system at application level. These models would be delivered by the teams supplying the IP using a common high level modeling language (possibly different models at different levels of details). Architects, with the overall specifications in mind, would inter-exchange the IP models potentially coming from different suppliers by testing them and commissioning modifications for fitting the required performances.

During bottom-up verification, once implementation is achieved and post-layout simulations with parasitics effects performed, the higher abstraction level models are updated with detailed parameters for reflecting the real device as accurately as possible. As opposed to the digital top-down only methodology, heterogeneous mixed systems need also a bottom-up methodology to propagate low-level properties/parasitics at system level. The verification is performed at each level for assuring that performances meet the expectations at each level of abstraction, normally the verification covers only the target cases of performances under analysis.

Concerning digital ICs design flow, when technology changes, the system described at RTL is coded in VHDL (studies are ongoing for obtaining the VHDL from C++ [Shcherbakov 11] and the implementation is directly synthesized and the netlist generated, place & route and timing analysis tools are used by the backend designers for producing the masks needed for the

fabrication of the chip. Unlike the digital design, when the technology changes less reuse of AMS/MEMS IPs is possible, nevertheless, if the technology is not changing it is typically a matter of resizing components or changing parameters, despite this, a portion of the design flow must be performed the same. The concept here is that from a low-level point of view no reuse of AMS/MEMS IPs is possible but from a high-level point of view analog behavioral models can be reused “as they are” or adjusted for performing a first top level architecture exploration and feasibility study.

Synthesis in the AMS domain is not feasible yet even if studies are ongoing [der Plas 02, O’Connor 06, Mitea 11], it is claimed that the top-design will be based on the refinement of models enabled by different tools for different levels of abstraction, and an analog and mixed-signal extension of SystemC will play a key role for high level AMS systems modeling.

2.3. Modeling-based refinement of heterogeneous systems

As already mentioned, for the design of digital systems, top-down design is state-of-the-art. The integration of analog/mixed-signal subsystems, which are mostly designed bottom-up, into a digitally dominated top-down flow is still a challenge. In the ideal case, top-down refinement begins with an executable specification of the intended behavior at system level. Refinement of the executable specification is part of the architecture exploration. The refinement process consists of a stepwise approach of replacing the blocks in the system with more accurate (less abstract) models.

Figure 2.3 shows the levels of abstraction in which the design flow or design view can be exploded. The root is the overall specification then functional units are identified wherein the functions digital/A(MS)/interfaces are separated and ulteriorly divided in architectural blocks. Together with the levels of abstraction the languages used for describing the models are shown as well. The AMS extensions to the SystemC locates at a higher abstraction level than the one offered by VHDL-AMS

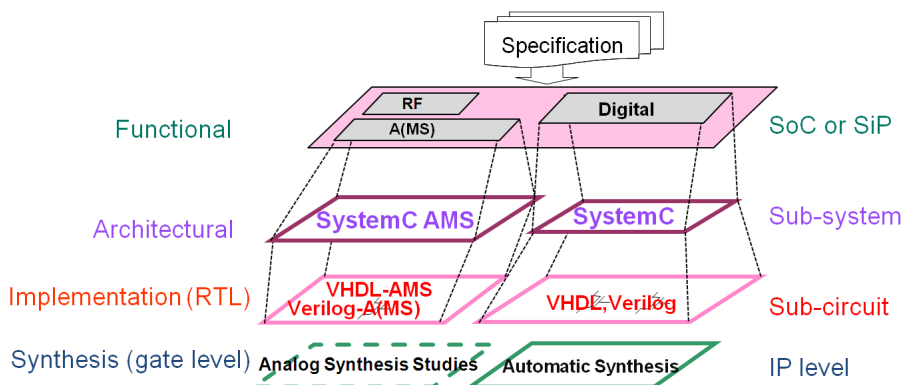


Figure 2.3.: Explosion in levels of abstraction of a heterogeneous analog and mixed-signal system.

The importance of modeling at every level of abstraction for interpreting the functioning of transistor/circuits/digital has been discussed. Models are everywhere in real world and it has been mentioned above that behavioral modeling would help at the system view for exploring

the design space and better understanding at early stages how components would interact. The top-down is not only the paradigm for designing ICs but it is a paradigm for real life. For most of the things in our daily life first the overall vision is captured then internal structures can be unrolled and developed and details unveiled.

A study about the formalization of the different abstraction levels for the AMS design flow has been carried out in [Paugnat 11]. A mapping of the levels of abstraction with the SystemC AMS MoCs is proposed and justified by means of a use case example.

2.4. State of the art of mixed (analog-digital) modeling languages/tools

In order to retrace the evolution of the modeling languages/tools that address heterogeneity in system design one important effort has been the development of the *Ptolemy II* software environment [Brooks 10]. The Ptolemy approach is based on the hierarchical composition of models, each one being possibly modeled using a different Model of Computation (MoC). A MoC defines a modeling formalism, i.e., a graphical or textual language with its syntactical elements, a set of syntax rules and a set of computational rules that define the semantics of the model. Such models are also called executable models as they can be used for simulation, synthesis, or formal proof. The Ptolemy II environment supports several MoCs, among others:

- **Discrete Event (DE) MoC:** is based on a discrete representation of time, which is suitable for modeling the behavior of digital and sampled systems. The computational rules for such MoCs are formally defined in the form of an event-driven logic simulator.
- **Continuous-Time (CT) MoC:** is based on a continuous representation of time, which is suitable for modeling the behavior of analog/RF systems. The computational rules for such MoCs are formally defined in the form of an equation solver or a circuit simulator.
- **Synchronous Data Flow (SDF) MoC:** is an asynchronous message-passing MoC, in which First-In, First-Out (FIFO) queues model the communication between processes. Processes encapsulate sequential behaviors and possibly states. This MoC can be either un-timed (i.e., only the presence of a input token or input data sample can trigger a process) or timed (i.e., process inputs and outputs are streams of sampled data). This MoC is suitable for modeling signal processing applications.

The Ptolemy II environment has been extensively used for research purposes and inspired the development of other system design frameworks such as *Metropolis* [Balarin 03] or *ForSyDe* [Sander 04]. These latter frameworks however deliberately limit their support to only one MoC, namely concurrent processes in Metropolis and purely synchronous/reactive models in *ForSyDe*. The main advantage over the Ptolemy approach is that they provide formal paths from abstract specifications to implementation through formal refinement, or synthesis, steps. The main drawback of the Ptolemy framework is its implementation that makes use of the Java structure, which is noticeably slow when complex broad systems have to be simulated.

MASCOT [Bjureus 01] has been developed as a modeling technique that integrates Specification and Description Language (SDL) models and Matlab [MATLAB 11] in a co-simulation environment. It also supports a modeling methodology, for which abstract specifications are de-

composed into a set of processes communicating through ideal, infinite-length, FIFOs. Processes are partitioned into control and data flow, thus associating the former to SDL computation and the latter to Matlab computation. The underlying MoC is a variant of the SDF MoC called Composite Signal Flow MoC.

Gielen [Gielen 05] provides an excellent review of methods and tools that have been developed over the past 10 years at Katholieke Universiteit Leuven (KUL) for the design of mixed-signal/RF integrated circuits and systems. It clearly shows the needs to have a rich library of behavioral models of basic building blocks that support a large spectrum of abstraction levels and MoCs. It also discusses the *current lack of systematic methods to generate appropriate behavioral models of analog/RF building blocks*.

Automated methods have been developed for extracting/abstracting behavioral models from circuit netlists [Nathke 04]. Such approaches fit well with bottom-up verification as the generated behavioral models have to more or less accurately represent real circuit behaviors. Although these models may also be used for top-down architectural exploration, other specific modeling approaches implementing functional behaviors with selected non-ideal effects such as phase noise or settling time can be used [De Smedt 99].

2.4.1. Application of modeling languages to hardware design

Parallel to the development of modeling representations (models of computation) and techniques to generate behavioral models, a lot of work has been done on the development of *modeling languages for hardware design*. In the 1980's, gate-level or transistor-level net lists were mostly used for logic/circuit simulation, timing and Layout-Vs-Schematic (LVS) verification. In the 1990's, Hardware Description Languages (HDLs) such as VHDL [IEEE 09c] and Verilog [IEEE 04] became the main design languages for digital hardware. In the same time, analog and mixed-signal extensions to these languages (i.e., the IEEE standard VHDL-AMS [IEEE 07] and Verilog-AMS [Accellera 09]) were developed. In the 2000's, C-based HDLs such as SystemC [Black 04, Grotker 02, IEEE 09a], SpecC [Gajski 00], Handel-C [Bowen 98] started to become the main design languages as they enable the development of more abstract models of complex systems including embedded software.

SystemC [Black 04, Grotker 02] is a C++ library of classes and methods that support the description and simulation of digital HardWare and SoftWare (HW/SW) systems from functional down to register transfer level by using the Discrete Event (DE) MoC. It has seen wide industry adoption over the past decade. Its development and standardization is coordinated by the Open SystemC Initiative (OSCI). Since 2006, it is an IEEE standard [IEEE 09a]. Its application domain is continuously broadening, as it is supporting powerful modeling and simulation capabilities at system level (e.g. transaction level models [Ghenassia 06, OSCI 09]) and it is currently being extended to better support (real-time) embedded software and AMS systems. The open source and object-oriented characteristics of SystemC together with the clear separation of computation and communication in it allow for adding support of various models of computation in a layered approach [Patel 05]. For digital HW/SW systems modeling, several competing efforts provide various untimed, synchronous, and timed MoCs such as Finite State Machine (FSM), Petri Net (PN), Kahn Process Network (KPN), Synchronous Data Flow (SDF), Synchronous Reactive (SR), and Clocked Synchronous (CS). Examples of such efforts are SystemC-H [Patel 04], and HetSC [Herrera 06], and UMoC++ [Mathaikutty 06].

All approaches impose additional rules on how to write modules to make use of one MoC. The implementation of the MoCs themselves can greatly differ. Some are exclusively implemented as channels that use regular SystemC events to coordinate their execution with the help of SystemC's discrete-event kernel. Some other use dedicated kernel extensions to speed up the execution of the MoCs.

In parallel to SystemC, SystemVerilog [IEEE 09b] has established itself as a popular hardware description, verification, and specification language. It is an object-oriented further development of the “classic” Verilog [IEEE 04] HDL to aid in the creation of complex system models on the RTL and on the architectural level¹ and facilitate their verification using simulation and formal assertion-based methods. It includes design specification methods, an embedded assertions language, a test bench language including coverage and an assertions Application Programming Interface (API) as well as a Direct Programming Interface (DPI) to interface with IP written in foreign programming languages (using C-bindings). SystemVerilog itself focusses on digital SoC design and does not yet offer AMS extensions comparable to Verilog-AMS². The Accellera Verilog Analog Mixed-Signal Group is currently managing to merge SystemVerilog and Verilog-AMS. In the meantime, vendor-specific co-simulation solutions are available to couple SystemVerilog with Verilog-AMS models. Compared to SystemC, SystemVerilog is a more closed environment, which hinders the integration of new MoCs into the language, which go beyond the semantics of SystemVerilog itself.

In [Pecheux 05], the modeling of an airbag system has been used to illustrate how the VHDL-AMS and the Verilog-AMS languages can be used to develop behavioral models with the necessary accuracy to validate the system's features while keeping the simulation time reasonably low (with respect to a full transistor level simulation). In addition, it showed the behavioral modeling of the chemical reaction, which inflates the airbag. This illustrated the capability of modeling and simulating physical behaviors other than purely electrical ones and, maybe most importantly, how a multidisciplinary or multiphysical system might be modeled and simulated as a whole.

In [Coytangiye 06], the capability of VHDL-AMS to describe compact semiconductor models is demonstrated. A first study focuses on the EKV v2.6 MOSFET model taking into account the thermoelectrical interactions and the extrinsic aspects. The EKV model uses linearization with respect to surface potential resulting in physically well based expressions for the whole model. A second study develops a simplified version of the MM11 Philips model, which takes into account the quantum mechanical effects. This compact MOSFET model is based on the formulation of the surface potential.


In [Mähne 06], the creation of virtual prototypes of complex micro-electro-mechanical transducers is presented. Creating these behavioral models can be partially automatized using a Reduced-Order Modeling (ROM) method. It uses modal decomposition to represent the movement of flexible structures. Shape functions model the energy conservation and full coupling between the different physical domains. Both modal shapes and shape functions for strain


¹Efficient system modeling on the architectural level with SystemVerilog requires the use of extension libraries like the ones developed as part of the Open Verification Methodology (OVM) [Systems 08]. These libraries provide building blocks for well structured and reusable verification components and test environments. To this end, they also provide support for Transaction-Level Modeling (TLM).


²The Verilog-AMS language [Accellera 09] is an extension of the digital Verilog language that has been standardized by Accellera. Verilog-AMS bears similar features with VHDL-AMS, but is mostly oriented towards circuits design.

energy and lumped capacitances of the structure can be derived in a highly automated way from a detailed 3-D FE model, available from earlier design stages. Separating the generation of the reduced-order models (ROM) from the same FE model but for different operation directions circumvents current limitations of the used ROM method. These sub models are integrated into a full model of the transducer. VHDL-AMS is used to describe additional strong coupling effects between the different operation directions, which are not considered by the used ROM method itself. The application of this methodology on a commercially-available yaw rate sensor, as an example for a complex transducer, demonstrates the practical suitability of this approach.

Figure 2.4 shows for which tasks popular modeling and verification languages are used in the SoC design process. We can say that first industrial usage of SystemC AMS is being done for analog/RF and further MoCs are planned for covering the other physical domains. In figure the SystemC AMS is represented to cover only the “transaction” level of abstraction but in most cases it is applicable/used even for the macro-architecture. However the borders of the abstraction levels are not unequivocally defined.

<div>physical domain</div> <div>level of abstraction</div>	software	digital	analog	radiofrequency	mechanical	optical	fluidic	chemical
service	CORBA							
transaction	SystemC TLM/ UML		SystemC AMS		SystemC-AMS MoC extensions			
macro-architecture	SystemC		Ptolemy			SystemC		
micro-architecture	SystemC / VHDL							
block		VHDL	RF Simulation / VHDL-AMS		VHDL-AMS	VHDL-AMS	VHDL-AMS	
circuit		Electrical Simulation		RF Simulation				
physical			Finite Element Methods				Finite Difference	FEMLab

 industrial solution

 laboratory solution


 no known solution

Figure 2.4.: Modeling languages for various physical domains at different abstraction levels, inspired by [O’Connor 07].

2.5. Objective of the thesis

Accepted that systems are nowadays more and more complex and multi-domain the use of high level languages for modeling such mixed-signal systems in a C++ environment has led the establishment of AMS extensions to SystemC. The objective of the thesis has been to explore the AMS extensions of the SystemC modeling and simulation framework for defining new modeling methodologies. These methodologies are related to physically heterogeneous analog and mixed-signal devices with the purpose to automatize the modeling flow for issuing high-level/reduced-order analytical models with simulation speed constraints. Analog to digital

interfacing procedures using the SystemC framework will be analyzed with the final aim to validate the overall platform in its DIGITAL/AMS/RF heterogeneity.

Chapter 3.

SystemC based environment for virtual prototyping

3.1. State of the art of AMS extensions to System C

In recent years there have been several parallel efforts to extend SystemC with analog and mixed-signal capabilities to describe heterogeneous systems.

In [Bonnerud 01], a SystemC simulation framework has been developed that defines analog signals as new SystemC objects. The simulation semantics is still event-driven, but so called virtual clocks allow to optimize the simulation of analog and mixed-signal modules. Also, a behavioral model library of analog and mixed-signal components allows building structural models of complex systems such as ADCs. However, this approach has not been generalized to model any kind of analog and mixed-signal behavior and the use of the event-driven formalism limits its application to signal flow behaviors and fixed time step integration.

In [Biagetti 04], a methodology for writing models of analog components using the SystemC standard library and simulation kernel is described. Analog modules are implemented as regular SystemC modules with a specific architecture to handle an adaptive time step simulation. The simulation of analog or mixed-signal models is event-driven, but each analog block is reactivated using its own time step. Ordinary differential equations have to be manually discretized with a proper time step. This approach primarily supports signal flow modeling.

The previous approach has been extended in [Vachoux 06] with the goal to support the modeling of conservative systems, e.g., by including wire load effects. Their work called *SystemC-WMS* allows the implementation of analog modules that communicate with each other by exchanging energy waves through wave channel interfaces. The wave channel interfaces are general analog interfaces that can support different physical domains. They allow the interconnection of modules that describe the component's physical behavior. As only the standard communication scheme of the SystemC kernel is used, no modification of the SystemC library itself is necessary. The wave channel interface simplifies also the interconnection of independently developed analog modules, because it avoids the interconnection problems commonly found in signal flow representations of conservative blocks, where the input/output role of the across (e.g., voltage) and through quantities (e.g., current) associated to the port have to be decided at implementation time of the module. This is too early because the direction of the information flow is determined from the interconnection of the modules, which is only known to the simulator at elaboration time. The wave channel methodology avoids this problem since incident waves always have the input role and reflected waves the output role. Parallel and

series connection of modules can be accounted through appropriate channels, which dispatch the waves to the modules they connect together. This is similar to the scattering junction of Wave Digital Filters (WDFs). The response of a module to the incident waves is described through a , b parameters, which are part of the WDF theory. The methodology can be extended to circuits with mildly nonlinear elements. A half-bridge inverter was modeled as an application example using SystemC-WMS and simulated using a fourth-order Adams-Bashforth Ordinary Differential Equation (ODE) solver. The simulation results showed good correspondence to the ones obtained from an equivalent model created using Matlab's Simulink Power toolbox and simulated using the `ode15s` stiff ODE solver. The simulation took about five times longer in SystemC-WMS. This can be partly attributed to the different ODE solvers used. Nevertheless the simulation performance of SystemC-WMS is limited to a good deal by the fact, that for each integration time step several discrete events are scheduled, which invoke the SystemC simulation kernel so that discrete and continuous parts cannot run independently from each other.

In [Al-Junaïd 05], SystemC is extended to SystemC-A that supports analog variables and analog components (e.g., SPICE-like primitives or user-defined components defining arbitrary Differential Algebraic Equations (DAEs)). Each analog component in a netlist contributes to the set up of a Modified Nodal Analysis (MNA) system matrix by specifying the contributions of each conservative terminal to the Jacobian and to the right hand side of the DAE system. The interconnection of analog and digital models is handled through specific interface models. Digital to analog interaction is realized by converting a digital signal into an analog signal using Backward Euler integration with very small time step. Analog to digital interaction is realized by detecting the crossing of thresholds. The SystemC simulation kernel is modified to include the execution of an analog solver. Mixed-signal timing synchronization is achieved using a lock-step mechanism to avoid backtracking. In [Al-Junaïd 06] SystemC-A is used to model an automotive seating vibration isolation system. The case study showed good correspondence between the simulation results of two equivalent models of the seating vibration isolation system, one written in SystemC-A and the other in VHDL-AMS. However, the presented solution has the drawback that it required modifications to the standard SystemC kernel itself to couple the analog solver with the discrete-event solver instead of providing an abstraction layer on top of SystemC to allow the parallel integration of various continuous time MoCs. Also, the way of defining contributions to the system matrix is very close to circuit level and thus may not be an appropriate approach for complex heterogeneous systems.

In [Vachoux 03], it is defined the context of the development of extensions to the SystemC modeling framework to support the description and the simulation of analog and mixed-signal systems, called SystemC AMS. In [Vachoux 05], the developed SystemC-AMS¹ prototype is presented in detail. Two formalisms, or MoCs, are implemented: a timed variant of the Synchronous Data Flow (SDF) MoC is used to model signal processing dominated behaviors as well as more general continuous-time behaviors using oversampled models and a linear network MoC provides a library of linear electrical primitives for describing linear macro models. Both MoCs are synchronized with the discrete event SystemC simulation kernel through a synchronization layer, thereby allowing mixed-signal and mixed-MoC simulation.

¹The official terminology states that "SystemC AMS" refers to the OSCI standard while "SystemC-AMS" refers to the simulator

This early SystemC-AMS prototype has been evaluated towards its applicability to the modeling of MEMS, which involve several physical domains under the rule of energy conservation laws. The prototype was examined on a heterogeneous inertial navigation system with micro-mechanical sensors. Two modeling approaches were reported. In the first one, the mechanical quantities were mapped on electrical ones, so that the linear electrical network models like resistor, capacitor, and inductor provided by SystemC AMS could be used to represent the mechanical dampers, springs, and masses. In the second approach [Markert 07], the mechanical system was modeled using a non-conservative block diagram with feedback, which was simulated using the SDF MoC and had simulation performance advantages over the first approach. The developed models showed good conformance to the results obtained with VHDL-AMS reference models of the sensor. However, the development effort for the SystemC AMS models was higher due to the lack of dedicated modeling capabilities for multiphysical systems. The work also showed a need to integrate such capabilities into SystemC AMS, because the considered system contained, besides the MEMS sensor and its analog front-end, a tightly coupled digital part with embedded software for the signal processing of the sensor signal and tuning of the sensor. For modeling this digital HW/SW part, SystemC AMS is better suited than VHDL-AMS yielding in a considerable simulation performance advantage.

Similar findings are reported in [Caluwaerts 08], where the modeling of an electromechanical energy harvester consisting of a resonator, a variable capacitor, a charge pump and a flyback circuit is presented. A combination of linear electrical primitives and user-defined SDF modules is used to describe the system. The nonlinear behavior of the involved diodes is modeled as a resistor, which resistance value is controlled through an SDF module based on the sensed voltage across the resistor. This very close feedback imposes very small time steps for the transient simulation due to the required unit delay in the SDF feedback loop to keep the resulting error small. This clearly shows SystemC AMS's current limitations regarding the modeling of physical systems due to the missing nonlinear solver and dynamic time step capabilities.

In [Herrera 07], it is shown how to couple the SystemC-AMS 0.15RC4 simulator [Vachoux 05] with HetSC [Herrera 06] to support in parallel a wide range of MoCs. This enables the use of SystemC for the complete specification of increasingly heterogeneous embedded systems, which include software control parts, digital hardware accelerators, analog front-ends, etc. Semantical and syntactical issues for the cooperation of both libraries on top of SystemC are discussed with a focus on the interfaces provided between the different MoCs for their interaction during simulation. One possibility for the synchronization of the MoCs from the two libraries is to use the DE MoC of SystemC as an intermediate layer, as both libraries already offer this synchronization capability. The second possibility allows a direct coupling of specific HetSC MoCs and SystemC-AMS MoCs using the concept of border channels in HetSC. The paper recognizes the need to standardize the synchronization semantics between different MoCs. In [Zaidi 10], it is shown how to couple the timed SDF MoC of SystemC-AMS via the Verilog Procedural Interface (VPI) with an AMS simulator to do mixed-level co-simulation of AMS systems, in which most parts of the system are modeled on the system level using SystemC(-AMS) and selected blocks are replaced through more detailed behavioral (Verilog-AMS or VHDL-AMS) models or circuit level (SPICE) models.

All two efforts [Herrera 07, Zaidi 10] show the advantages of the SystemC-AMS architecture, which facilitates the integration of very different modeling formalisms and tools to an efficient simulation platform supporting the specification, design, and verification of complex heterogeneous AMS SoCs.

In [Zhu 10], a formal heterogeneous model of computation framework for SystemC is introduced and called HetMoC. It complements the already presented approaches [Herrera 06, Patel 05, Vachoux 05]. It is based on a very formal definition of the semantics for Continuous-Time (CT), Discrete Event (DE), Synchronous Reactive (SR), untimed Data Flow (DF), and SDF model domains. A new modeling style for the CT MoC is presented, which has pure CT dynamics. Based on it, the other MoCs are derived by stepwise abstraction. In this framework, the target system is modeled as a process network. Blocks are processes that specify computation and edges are signals to connect processes. For each model domain the signals, domain interfaces, and processes are defined. The domain interfaces are polymorphic allowing to combine models using different MoCs. The implementation of the HetMoC framework in SystemC is inspired by a system level functional modeling style proposed for untimed dataflow models in [Grotker 02]. All models are communicating through standard SystemC `sc_core::sc_fifo<T>` channels. As an application example, an adaptive Amplitude-Shift Keying (ASK) transceiver system is modeled with HetMoC and its simulation results/performance is compared to a SystemC AMS reference model. The results are promising even though SystemC AMS showed a clear performance advantage paid with a higher memory consumption due to its more complex but optimized implementation. HetMoC's implementation seems to rely for the moment purely on SystemC's DE kernel to control its model execution without doing any kernel extension to optimize its model execution. This could be an explanation for the performance disadvantages over SystemC AMS. In the perspective of the standardization of AMS extensions to SystemC, the presented framework is interesting due to its sound formal base to integrate different modeling domains.

The development of the aforementioned SystemC-AMS prototype [Vachoux 05] was accompanied by the SystemC AMS study group [Study-Group 12] with the goal of generalizing and standardizing the concepts introduced with SystemC AMS. With support from the semiconductor industry (notably NXP Semiconductors, Infineon Technologies and STMicroelectronics), the study group promoted the creation of an official working group within the OSCI consortium that coordinates the development/standardization of SystemC and related libraries. Since its foundation in 2006, the charter of this OSCI AMS Working Group (AMSWG) has been the development and standardization of AMS extensions to SystemC to promote their industry acceptance. Based on the collected requirements and use cases [Vachoux 03], the AMSWG developed a Language Reference Manual (LRM), which became in March 2010 an official OSCI standard [OSCI 10]. In parallel to the standard release, the AMSWG published a user's guide [Barnasconi 10]. Fraunhofer IIS/EAS released also a conforming Proof of Concept (PoC) implementation of the SystemC AMS extensions 1.0 [IIS/EAS 10] as a further development of its former SystemC-AMS prototype. In their first version, the AMS extensions primarily address the needs for describing the continuous-time behavior of purely electrical AMS SoCs by proposing three MoCs, which allow their description on different levels of abstraction using Timed Data Flow (TDF), Linear Signal Flow (LSF), and Electrical Linear Network (ELN). This makes them well-suited for the design of communication systems, which analog front ends are tightly coupled to complex digital control, and for the design of Digital Signal Processing (DSP) applications. However, their modeling capabilities are not yet well suited to describe energy-conserving multi-physical system components with nonlinear behavior in a formal and consistent way at a high level of abstraction. The requirements specification for the AMS extensions shown [Vachoux 03] already mentions these needs to enable their usage, e.g., in the automotive sector. The OSCI SystemC AMS extensions will be introduced more in details later on in this Chapter.

First experiments were already done with the old SystemC-AMS prototype to extend it for the modeling of conservative elements with nonlinear dynamic algebraic equations. The results have been reported in [Einwich 06] using a micro relay as a multi-physical example. The newly proposed nonlinear MoC defines a new module class, which provides callbacks that can be overloaded to describe the module's energy-conserving behavior as contributions to the DAEs system of the nonlinear network formed by the interconnected modules of this type. To this end, each module defines its contribution to the through values (currents) of the nodes connected via the module's ports in dependency of the across values (voltages) and the derivation of the across values. Additional equations have to be described in the form $0 = F(t, v_{ports}, vars_{ports}, vars_{ports}, \dot{v}_{ports})$. The proposed syntax resembles Verilog-AMS, but in a way that it conforms to the syntax of C++. The advantage of the proposed approach is in its modularity. The nonlinear MoC integrates itself into the infrastructure of SystemC-AMS without requiring modifications to the latter. It uses the synchronization layer of SystemC-AMS to seamlessly interface with the SDF MoC of SystemC-AMS and the DE MoC of SystemC. Thus, large heterogeneous systems can be simulated, which components have been modeled using different MoCs in parallel.

In [Uhle 10], it is reported how the previous approach has been refined and generalized. Natures can now be declared like in VHDL-AMS to associate nodes, terminals, and branches to physical domains. Thus, model assembly mistakes can be detected upon compile time. The implemented syntax for describing the energy conserving behavioral has been improved to be more VHDL-AMS-like and thus user-friendly. The proposed new language constructs have been aligned with the syntax of the standardized OSCI SystemC AMS extensions. The implementation is founded on the SystemC-AMS PoC implementation of this standard. The nonlinear solver can generate events based on threshold crossing and is able to backtrack to react on events. Thus, if an Non-Linear Network (NLN) model is solely coupled to a DE model, the synchronization semantics are equivalent with the VHDL-AMS simulation cycle. If TDF models are additionally coupled to the NLN model, the synchronization becomes more complex, because synchronization can only happen at the end of a TDF cluster period. The capabilities of this NLN MoC are demonstrated on a complex electromechanical window lifter model coupling the electrical, mechanical, and magnetical domains. For a pure NLN model, both approaches [Einwich 06, Uhle 10] cannot achieve considerable runtime advantages over an equivalent VHDL-AMS/Verilog-AMS model, as the underlying DAE system and nonlinear solver algorithms are similar. However, in nowadays typical cases where the complexity of the digital HW/SW part dominates over the analog part, the system simulation clearly profits from the more abstract modeling capabilities offered by SystemC and its AMS extensions.

In [Hartmann 09], the modeling of physical control systems with SystemC AMS is described. A model of a crane with embedded control is used, which has been already previously proposed as a system modeling benchmark [Moser 99]. Due to numerical stability issues, an external 4th-order Runge-Kutta solver is integrated into the simulation replacing the linear State Space (SS) solver provided by SystemC AMS. The approach is interesting, because the external solver is encapsulated in a way that it provides the same interface as the original SS solver and thus requires only minimal modifications to the model itself.

An extensive bibliography of publications related to the initial SystemC-AMS prototype and its standardized successor in form of the OSCI SystemC AMS extensions can be found on the homepage of the SystemC AMS study group [Study-Group 12]. It documents its growing popularity in the research and industrial communities, who are applying both to a

variety of application domains, e.g., RF systems [Adhikari 10, Vasilevski 08, Xu 09], control systems [Hartmann 09], test development [Lu 09], automotive components [Arndt 10, Kreuter 09, Rafaila 10, Uhle 10], biological labs on chip [Pecheux 10]. The SystemC AMS extensions have already proved their capability as an integration platform able to support very different modeling formalisms, which can interact with each other. The openness of the C++-based SystemC simulation framework allows an extensibility with 3rd party libraries and tools, which cannot be matched by “classical” HDLs, as they do not give the user such a deep access to the internals of the simulation mechanism. For the AMS extensions, goals are to make the TDF MoC even more flexible by supporting features such as the dynamic modification of the TDF time step or to trigger the TDF cluster execution as a reaction to events. Long term goals are to define a standard Application Programming Interface (API) for plugging in external solvers and the formalization of the synchronization layer between the CT and DE MoCs to offer a standardized API for the integration of new MoCs.

3.2. SystemC AMS 1.0 OSCI standard model abstractions

The SystemC AMS extensions add new abstraction methods for system-level modeling and simulation of AMS systems to the existing SystemC framework. The model abstractions supported by the SystemC AMS extensions are based on well-known methods for abstracting analog and mixed-signal behavior. As shown in Figure 3.1, the abstraction levels distinguish discrete-time from continuous-time behavior and non-conservative from conservative descriptions.

Discrete-time vs. continuous-time descriptions.

On the one hand *discrete-time modeling* abstracts signals (e.g., audio or video streams) or physical quantities (e.g., voltages, currents, and forces) as sequences of values only defined at discrete time points. Values may be either real values or discrete values (e.g., integer or logic values). Values between time points are formally not defined, although it is common to consider them as constant. Behaviors are then abstracted as procedural assignments involving sampled signals. The description of static (algebraic) non-linear behaviors (e.g., using polynomials) is supported. Discrete-time modeling is particularly suited for describing signal-processing-dominated behaviors, for which signals are naturally (over)sampled. It can be also used for describing continuous-time behaviors, provided that the discrete abstraction produces reasonable approximations. On the other hand *continuous-time modeling* gets closer to the physical world, as signals and physical quantities are abstracted as real-valued functions of time. The time is now considered as a continuous value. Behaviors are then described using mathematical equations that can include time-domain derivatives of any order (so-called differential algebraic equations (DAEs) or ordinary differential equations (ODEs)). Equations must be solved by using a dedicated linear or non-linear solver, which usually requires complex numerical or symbolic algorithms. Continuous-time modeling is particularly suited for describing physical behaviors, as it can naturally account for dynamic effects.

Non-conservative vs. conservative descriptions.

Continuous-time models can be divided into two classes: non-conservative and conservative models. On the one hand *non-conservative models* express behaviors as directed flows of continuous-time signals or quantities, on which processing functions such as filtering or

integration are applied. Non-linear dynamic effects can be properly described, but mutual effects and interactions between AMS blocks, such as impedances or loads, are not naturally supported. On the other hand *conservative models* are the most detailed continuous-time models at system level and circuit level, as energy conservation laws (Kirchhoff's laws) must be satisfied. As a result, the set of equations to be solved is larger and possibly more complex than the ones inferred by non-conservative models.

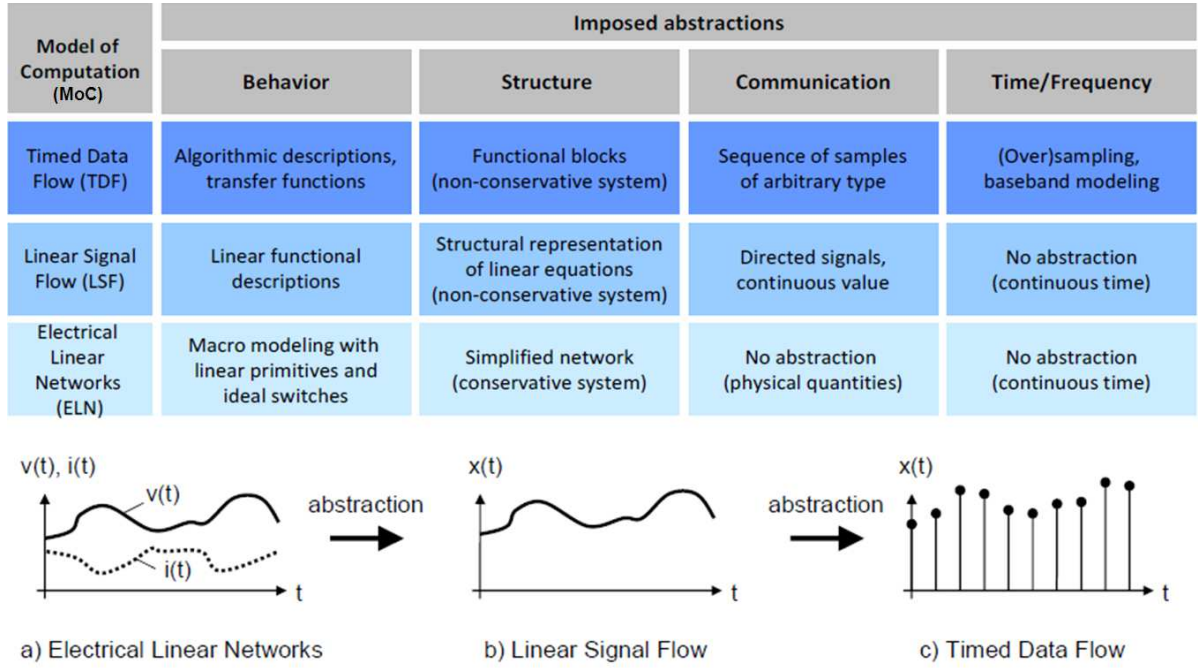


Figure 3.1.: Abstractions in relation to the SystemC AMS models of computation.

3.3. SystemC AMS 1.0 OSCI standard models of computation

The SystemC AMS extensions also define the essential modeling formalisms required to support AMS behavioral modeling at different levels of abstraction (Figure 3.1). These modeling formalisms are implemented by using different models of computation (MoCs): Timed Data Flow (TDF), Linear Signal Flow (LSF), and Electrical Linear Networks (ELN). The three MoCs are here briefly introduced, in the following the modeling fundamentals for each MoC will be shown by means of images and explanations extract from the user's guide [Barnasconi 10] wherein further details can be found:

- **Timed Data Flow (TDF):** the execution semantics based on TDF introduce discrete-time modeling and simulation without the overhead of the dynamic scheduling imposed by the discrete-event kernel of SystemC. Simulation is accelerated by defining a static schedule, which is computed before simulation starts, and which executes the processing functions of the scheduled TDF modules according to the stream direction of the dataflow. The sampled, discrete-time signals, which propagate through the TDF modules may represent any C++ type. If, e.g., a real-valued type such as double is used, the TDF signal can

represent a voltage or current at a given point in time. Complex values can be used to represent an equivalent baseband signal.

- **Linear Signal Flow (LSF):** the Linear Signal Flow formalism supports the modeling of continuous-time behavior by offering a consistent set of primitive modules such as addition, multiplication, integration, or delay. An LSF model is made up from a connection of such primitives through real-valued time-domain signals, representing any kind of continuous-time quantity. An LSF model defines a system of linear equations that is solved by a linear DAE solver.
- **Electrical Linear Networks (ELN):** modeling of electrical networks is supported by instantiating predefined linear network primitives such as resistors or capacitors, which are used as macro models for describing the continuous-time relations between voltages and currents. A restricted set of linear primitives and switches is available to model the electrical energy conserving behavior.

3.3.1. TDF modeling fundamentals

The Timed Data Flow (TDF) model of computation is based on the well-known Synchronous Data Flow (SDF) modeling formalism. Unlike the untimed SDF model of computation, TDF is a discrete-time modeling style, which considers data as signals sampled in time. These signals are tagged at discrete points in time and carry discrete or continuous values like amplitudes. Figure 3.2 shows the basic principle of the Timed Data Flow modeling. In this figure, there are three communicating *TDF modules* called A, B, and C. A *TDF model* is composed of a set of connected TDF modules, which form a directed graph called *TDF cluster*. TDF modules are the vertices of the graph, and *TDF signals* correspond to its edges. A TDF module may have several input and output *TDF ports*. A TDF module containing only output ports is also called a producer (source), while a TDF module with only input ports is a consumer (sink). TDF signals are used to connect ports of different modules together. Each TDF module contains a C++ method that computes a mathematical function f (i.e., f_A , f_B , and f_C), which depends on its direct inputs and possible internal states. The overall behavior of the cluster is therefore defined as the mathematical composition of the functions of the involved TDF modules in the appropriate order, $f_C (f_B (f_A (...)))$, indicated with $A \rightarrow B \rightarrow C$ in Figure 3.2.

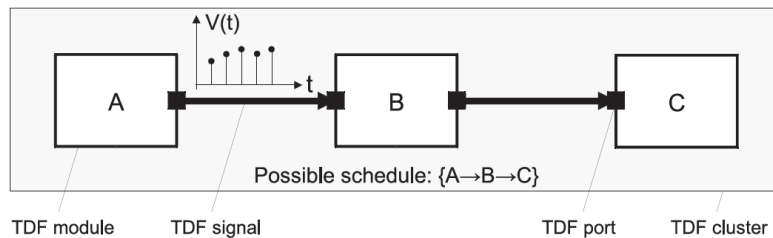


Figure 3.2.: A basic TDF model with 3 TDF modules and 2 TDF signals [Barnasconi 10].

A given function is *processed* (or “fired” according to the SDF formalism) if and only if there are enough samples available at the input ports. In this case, the input samples are read by the TDF module, where the function uses these values to compute one or more resultants, which are written to the appropriate output ports. In TDF, the number of samples read from or written to the module ports is fixed during simulation, but the numbers of read and written

samples by a TDF module are not necessarily equal. A time stamp is associated to each sample using the local TDF module time. The fixed interval between two samples is called *time step*.

A TDF module is the basic structural building block allowing to describe both discrete-time and continuous-time behaviors. It is a class that implements a TDF behavioral description, and may not instantiate other modules. TDF modules act as primitive modules.

- **Discrete-time modeling:** discrete-time behavior can be defined in the member function **processing**. In this member function, a pure algorithmic or procedural description in C++ can be given, which is executed at each module activation. The module activation is defined by the module time step, which can be either user-specified with the member function `set_timestep` or derived by time step propagation.
- **Continuous-time modeling:** a TDF module can be used to embed linear dynamic equations in the form of linear transfer functions in the Laplace domain or state-space equations. Although the TDF model of computation processes the samples at discrete time steps, the equations of these embedded functions will be solved by considering the input samples as continuous-time signals. The result of the embedded linear dynamic equations system, which is also continuous in time and value, is sampled into a signal using a time step which corresponds to the time step of the port, in which the samples are written.

3.3.2. ELN modeling fundamentals

The Electrical Linear Networks model of computation introduces the use of electrical primitives and their interconnections to model conservative, continuous-time behavior. The ELN modeling style allows the instantiation of electrical primitives, which can be connected together using *electrical nodes*, to form an *electrical network*. The mathematical relations between the electrical primitives are defined at each node in the network, where both the potential (voltage) and flow (current) quantities are used according to Kirchhoff's voltage law (KVL) and Kirchhoff's current law (KCL). As such, the electrical network is represented by a set of differential algebraic equations, which will be resolved during simulation to determine the actual circuit behavior.

Figure 3.3 shows an example of an electrical network, with two resistors, a capacitor, and a current source. Such a network is called an *ELN model* and is composed of a set of connected *ELN primitive modules*, which will form together an *ELN equation system* or *cluster*. Each ELN primitive module can have one or more *ELN terminals*. The ELN primitive modules are interconnected via their terminals using *ELN nodes*. The reference or ground node, which always has a voltage of zero, is called *ELN reference node*. ELN terminals are also used as an interface to connect the ELN model with other ELN models.

3.3.3. LSF modeling fundamentals

The Linear Signal Flow model of computation allows the modeling of AMS behavior defined as relations between variables of a set of linear algebraic equations. LSF is a continuous-time modeling style using directed real-valued signals, resulting in a non-conservative system description. There is no dependency between flow and potential quantities; instead only one

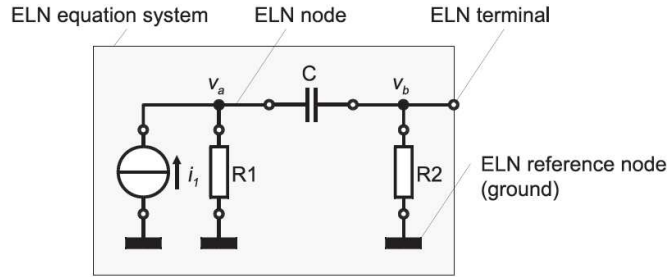


Figure 3.3.: Example of a basic ELN model representing an electrical network [Barnasconi 10].

real-value quantity is used to represent each signal. Signal flow models can be described in a block diagram notation. The elementary parts or functions are represented by blocks. Signals are used to interconnect these blocks. The resulting relations between the blocks define equivalent mathematical equations. Figure 3.4 shows an example of such a signal flow block diagram, composed of four *LSF modules*, which are interconnected using *LSF signals*. Note that the addition “operator”, although having a different graphical representation, is also an LSF module. An *LSF model* is composed of a set of connected LSF modules, which will form together an *LSF equation system* or *LSF cluster*. The resulting LSF model has input and output LSF ports to connect it with other modules.

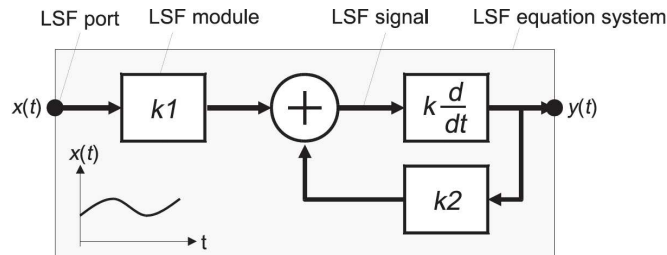


Figure 3.4.: Example of a basic LSF model composed of 4 LSF modules [Barnasconi 10].

3.4. Interaction among SystemC AMS models of computation

In order to understand how the MoCs interact among them it is wise to introduce the **layered structure of the SystemC-AMS simulator architecture** to the SystemC framework. Figure 3.5 shows how these three formalisms are integrated with the discrete event modeling formalism of SystemC by implementing the AMS extensions in a layered architecture on top of the standard SystemC kernel. Each formalism requires its own execution layer that implements the formalism’s underlying MoC. For LSF and ELN MoCs, a linear DAE solver is required and for the multi-rate TDF MoC, a scheduler. A synchronization layer coordinates the parallel execution of the different continuous-time MoCs and offers to them a common interface to interact with each other and with the DE simulation kernel of SystemC. Thus, the simulation of heterogeneous SystemC models is possible, which employ in parallel different MoCs. The LRM of the AMS extensions [OSCI 10] just defines the user interfaces to the three offered MoCs in form of the proposed modeling primitives and their semantics. It also specifies the exact synchronization semantics between the different MoCs. The interfaces to the model

execution and synchronization layer are not standardized and can thus vary between different implementations of the standard. At the time of the writing of this thesis, one Proof of Concept (PoC) implementation of the OSCI SystemC AMS extensions is available from Fraunhofer IIS/EAS as open source under the Apache license and is called SystemC-AMS [IIS/EAS 10].

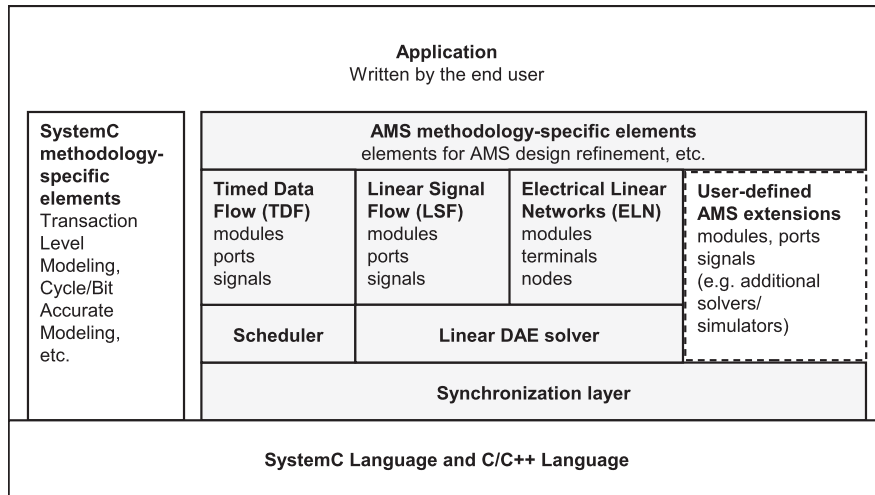


Figure 3.5.: Architecture of the OSCI SystemC AMS extensions 1.0 standard.

The design of embedded analog/digital systems requires the combination of different models of computation and of different levels of abstraction. This requires the conversion of communication/synchronization at the border between different models of computation. The SystemC AMS extensions provide a basic set of language primitives that enable conversion between SystemC (discrete-event), TDF, ELN, and LSF. In ELN and LSF, converter modules are provided; in TDF, converter ports are available. It is recommended to model the general signal flow of a system using the TDF model of computation, if possible. This has the following advantages:

- The TDF model of computation provides conversion to all other models of computation.
- The TDF model of computation is needed to provide time steps to connected ELN and LSF components.

3.4.1. Continuous-time (ELN or LSF) to/from discrete-time (TDF) or discrete-event (DE) domains

Both the ELN and the LSF MoCs will set up and solve an equation system to simulate the modeled continuous time behavior, based on the basic set of ELN/LSF primitive modules. Any “external” input value, e.g., from a discrete-event signal or TDF sample, need to be contributed to the equation system via one of these ELN/LSF primitive modules. Therefore, specialized primitive modules with ports to the discrete-event domain and TDF MoC are available, which are called converter modules (*sources* for writing to ELN/LSF, *sinks* for writing to TDF/DE). These modules establish an interface to convert and transfer data from one MoC to the other.

- **Reading from DE or TDF:** a module time step has to be assigned to the ELN/LSF converter module. The ELN/LSF model continuously reads values from the input at the time points for DE (the samples for the TDF), which are calculated from the time steps assigned to the ELN/LSF converter module. In the case of reading from DE, the input value is assumed constant until the next value is read. The input values are then interpreted to form a continuous-time signal, which is made available at the output of the converter module.
- **Writing to DE or TDF:** the samples (for TDF) or the values (for DE) at the output port are written at the calculated time points, which correspond to the time step assigned to the converter module.

3.4.2. Continuous-time conservative to/from non-conservative domain (ELN/LSF)

It is necessary to convert to the TDF and back. Note that ELN and LSF can communicate with discrete-event and TDF, but not with each other in a direct way, the conversion from LSF (or ELN) to TDF and then to ELN (or LSF) introduces a delay of one time step.

3.4.3. Discrete-time domain to/from discrete event SystemC domain (TDF/DE)

TDF MoC has its own mechanisms for time annotation, which could result in time differences between the local time of each TDF module and the time in the discrete-event domain (SystemC kernel time). Therefore, special care should be taken in synchronizing TDF signals with the discrete-event domain of SystemC in both directions (i.e., reading from and writing to discrete event signals). To maintain a high simulation efficiency despite the presence of TDF and discrete-event domain interactions, a loosely-coupled synchronization mechanism is used, which is called data synchronization. For TDF modeling this means that discrete events will not influence the activation and execution of TDF modules.

- **Reading from the discrete-event domain:** unlike the regular TDF input ports of class `sca_tdf::sca_in<T>`, the availability of a discrete event signal at the TDF input converter ports will not activate module execution. Instead, the TDF module activation order (schedule) is determined independently at its individual port time step in accordance with the converter port rate and the TDF module time step. Precondition for correct data synchronization is that the value read from the converter port should be available at the first delta cycle of the corresponding time point in the discrete-event domain. As the TDF cluster runs independently from the discrete-event domain, it could happen that the previous discrete-event value is read, indicating that a discrete-event process did not write the value to the channel before the first delta cycle. This would result in a delay in the signal. To overcome this, a small time offset could be introduced using the port member function `set_timeoffset`.
- **Writing to the discrete-event domain:** the time offset and time step assigned to the output converter port define, at which time point and time interval a value is written to the discrete-event domain. Precondition for correct data synchronization is that the sample written to the converter port can be written to the associated channel at the

first delta cycle of the corresponding discrete-event time point. In case a channel of class `sc_core::sc_signal<T>` is connected to the converter port, discrete-events are generated only in case of a signal change. In case a channel of class `sc_core::sc_buffer<T>` is connected to the converter port, all samples written to the port will generate an event.

3.5. SystemC TLM

In order to interface models of AMS components with the surrounding digital system at a very high abstraction level in a unified SystemC based framework it is necessary to interface the SystemC AMS MoCs with the Transaction Level Modeling support for SystemC.

Transaction-level modeling (TLM) is a transaction-based modeling approach founded on high-level programming languages such as SystemC. It highlights the concept of separating communication from computation within a system. In TLM notion, components are modeled as modules with a set of concurrent processes that calculate and represent their behavior. These modules exchange communication in the form of transactions through an abstract channel. TLM interfaces are implemented within channels to encapsulate communication protocols. To establish communication, a process needs to access these interfaces through module ports. Essentially, the interface is the very part separating communication from computation within a TLM system. TLM defines a transaction as the data transfer (i.e. communication) or synchronization between two modules at an instant (i.e. SoC event) determined by the hardware/software system specification. Using SystemC as a vehicle to provide the Transaction Level Modeling (TLM) abstraction proved to be the key to the fairly fast deployment of this methodology. One of the final aims of the TLM paradigm is indeed to bridge the gap between the embedded software developer and the hardware architect by enabling hardware/software co-development based on virtual prototypes. TLM-based SoC platforms actually allow early application software development by the end customer before the actual hardware architecture is even frozen.

The OSCI released the official OSCI TLM standard and a first proof of concept library called SystemC-TLM 1.0 in 2005 providing two modeling styles PV (Programmer View) and PVT (Programmer View Timed). Later on, in 2008 the TLM 2.0 standard defining the interfacing of memory-mapped bus has been released by OSCI. SystemC TLM models for processors, RAM and peripheral are represented by SystemC processes that communicate (i.e. exchange packet data) by means of Interface Methods Calls (IMC) instead of cycle-accurate signals. The TLM 2.0 C++ methods involved in the transfer of data between components can either be blocking or non-blocking according to two coding styles, loosely-timed (TLM-LT) and approximately-timed (TLM-AT). These two coding styles provide a good trade-off between simulation speed and accuracy. The key point idea in TLM 2.0 is that exchanged packet data correspond to the payloads of transactions, sent along a communication channel with a delay relative to the global simulation time. This delay can be used to make some components run ahead of the global simulation time, thus providing a useful mechanism known as temporal decoupling. This way, context switches between processes are reduced as necessary, and simulation speed is greatly increased.

In the TLM 2.0 standard the more accurate, **approximately-timed** (AT) coding style is typically achieved by using the *non-blocking transport interface* and the *payload event*

queues. While a fast, **loosely-timed** (LT) coding style is typically expected to use the *blocking transport interface*, the *direct memory interface*, and *temporal decoupling* (refer to [OSCI 08] for more details). The *blocking transport interface* is appropriate where an initiator wishes to complete a transaction with a target during the course of a single function call, the only timing points of interest being those that mark the start and the end of the transaction. The *non-blocking transport interface* is appropriate whether it is desired to model the detailed sequence of interactions between initiator and target during the course of each transaction. In other words, to break down a transaction into multiple phases, where each phase transition marks an explicit timing point.

The two modeling styles offered by the SystemC-TLM are equivalent to two levels of abstraction of modeling of the timing, this is analogous to the SystemC AMS different MoCs. Other developments of TLM coding styles and interfaces using SystemC have been studied by other research works. Notably, an extension of the approximately-timed coding style (TLM-AT), called TLM-DT, for Transaction Level Modeling with *Distributed Time* has been studied by *Université Pierre et Marie Curie*, Paris [Mello 10]. The main idea is to rely on *Parallel Discrete Event Simulation* (PDES) principles to simulate a complete More-than-Moore system. In this approach, there is neither a global scheduler nor a global clock. The temporal decoupling mechanism of TLM 2.0 is generalized. To each process (TLM) involved in the simulation is associated a local clock, defining a local time, and the processes synchronize themselves by embedding timing information in the data packets carried through the communication channels. Two kinds of PDES exist, the *pessimistic* and the *optimistic* PDES. The pessimistic PDES is regarded as more suitable for the interaction with the SystemC AMS TDF MoC. In the pessimistic PDES, a process is allowed to increase its local time if and only if it has the guaranty that it can not receive, on any of its input channels, a message with a timestamp smaller than its local time, and the simulation relies on the temporal filtering of the incoming messages.

Finally, companies that started using the OSCI SystemC-TLM 1.0 standard at its early stage in 2005 had first adapted the coding styles PV and PVT to their needs and preferences. The case of STMicroelectronics is a bit different since the company has not only created their own transaction-based protocols by extending the PV and PVT coding styles (see TLM_TAC, TLM_STBUS and TLM_SYNCHRO protocols in [Ghenassia 06]), but has actually driven the development of the OSCI standard for memory-mapped bus communications. Once the 2.0 standard has been released companies had to check the compliancy of their previously developed extended protocols with the new standard and to correct where needed for keeping the compliancy and take advantage of the infrastructure offered by the OSCI 2.0 release. Since TLM 2.0 only defines the memory-mapped bus interface, in the case of STMicroelectronics the philosophy is to keep the compliancy to the TLM 2.0 standard and to develop their own protocols for the communication not covered by the standard by keeping a timing philosophy close to the TLM-LT. The concept to bear in mind is that the main purpose of virtual prototyping is to early develop/debug the embedded software (relaxed timing) instead of performing studies on the hardware architecture (accurate timing required). Concerning to the coding style of the STMicroelectronics' models' inside, the timing aspect is specifically handled depending on the application.

3.6. Interaction between SystemC AMS models of computation and SystemC-TLM 2.0

The current scenario of available tools for the interaction between AMS systems described at system level and their surrounding digital world does not offer an efficient, high simulation-speed interaction. A promising solution certainly is the interaction between SystemC AMS and SystemC-TLM. Mixed signal simulations would benefit from this unified modeling and simulation environment thanks to the availability of different levels of abstraction. For instance, if an AMS intellectual property (IP) described in SystemC AMS has to be simulated together with its digital control/consuming units, the level of accuracy can be increased by either refining the timing aspect SystemC-TLM 2.0 AT/LT or by changing the SystemC AMS MoC.

Different interfacing cases can be wise to consider, however it is recommendable to take into account interfacing between similar levels of abstraction, in this case the highest level offered by SystemC AMS 1.0, that is the TDF MoC, with the SystemC-TLM 2.0 LT coding style for relaxed applications (formalized as *communication* applications) and with the SystemC-TLM 2.0 AT coding style for more severe and crucial applications (formalized as *automotive* applications).

It is our opinion that generic converters are required in order to automatize the interfacing between, some works can be found in the literature about the interfacing and *ad-hoc* solutions are mostly used depending on the application.

In [Beserra 11] an `sc_export` was created for converting from SystemC AMS TDF to TLM 1.0 PV coding style. In addition, an `sc_port` was added and connected into the processor to generate the interrupts. No other details on the interfacing strategy are provided.

In [Schulz 10], it is shown how analog transmission effects can be reproduced by means of TLM 2.0 interfaces. This work is addressing the likelihood of TLM transactions in order for them to reflect as much as possible the analog effects of the line. Non-blocking transport interfaces are used and the analog line model is described as a SystemC AMS model whose parameters come from layout extraction. Lookup tables are built since the idea is to do the time consuming simulations only once, subsequently the lookup tables are used in the SystemC TLM/AMS simulations.

In [Rafaila 09], an Electronic Control Unit (ECU) for an automotive application is simulated by interfacing the SystemC AMS TDF to SystemC TLM 1.0 following a master/slave principle. The ECU is composed by an MCU with AMS sensors/actuators and other peripherals. The interface between the SystemC model of the microcontroller (MCU) and SystemC AMS models of the sensors is done through a TLM interface. The MCU acts as the master and the AMS modules are enhanced with slave interfaces for read/write accesses.

Since the nature of the TLM loosely timed and approximately timed applications is not unique three interaction cases have been identified. A case-by-case study needs to be performed and the possible interactions are TDF/TLM-AT, TDF/TLM-LT and TDF/TLM-DT. The latter will not be discussed in this thesis. Once the type of interaction is defined the SystemC-AMS simulator is synchronized with the SystemC kernel by means of application-specific and user-defined converter ports.

In the industrial case-study of Chapter 5 a CMOS Image Sensor (CIS) is modeled using SystemC AMS and it is interfaced to its surrounding virtual SystemC TLM platform. First, it will be shown how the interfacing to the TLM OSCI coding style takes place by using point-to-point non-blocking transport interfaces.

Subsequently, the STMicroelectronics' proprietary protocols are used for interfacing the CIS model. The TDF MoC is used for the modeling of CIS and the acquired image is sent as transaction using the proprietary *streaming* protocol compliant with the OSCI TLM-2.0 coding style. Which is, as mentioned above, substantially in-line with the LT modeling approach. On the other side, the sensor control interface is implemented through an *ad-hoc* interfacing between the STMicroelectronics' control protocol and the TDF-based sensor model.

3.6.1. Interfacing TDF/TLM2.0 AT

In the automotive use case, the focus is on intermediate interactions between hardware (analog and digital) and software. A simulation model mixing SystemC AMS TDF with Bit-True Cycle-Accurate SystemC may be required. When high simulation speed is necessary and some time accuracy can be relaxed, the digital part can be modeled with SystemC TLM-AT. Yet, this type of simulation needs a rather tight synchronization schema that will slow down the simulation performance significantly versus the TLM-LT. One possible issue is to switch automatically from a lower abstraction formulation with SystemC TLM-AT coupled with AMS to a higher abstraction with SystemC TLM-LT coupled with AMS. Unfortunately the requirements are too different to allow a generic solution.

3.6.2. Interfacing TDF/TLM2.0 LT

In order to have an efficient synchronization mechanism between SystemC AMS and SystemC TLM the interfacing should become available at the highest possible abstractions, which is TDF for the analog functions and the LT coding style for TLM 2.0. It is proposed to introduce the concept of "loosely coupled analog and digital processes" allowing to avoid unnecessary kernel context switching. Specialized converter channels defining the communication and interface between TDF and TLM-LT have not been standardized yet. It is expected that these channels are implemented as TDF modules, which can be instantiated and configured by the user. The usage of these modules should be defined as being part of design refinement methodology. The TDF/TLM-LT interface should make use of the temporal decoupling concept as defined in the TLM 2.0 standard. Preferably, the global time quantum (e.g. using class `tlm_global_quantum`) and the quantum keeper should be used to manage time for these specialized converter channels.

In [Damm 08], a solution is proposed for coupling SystemC AMS TDF MoC models with loosely-timed TLM 2.0 models using a temporal decoupling approach by means of payload event queues (PEQs) and FIFO buffers.

Part III.

The contribution

Chapter 4.

Specification and implementation of SystemC AMS extension libraries

4.1. Introduction to high-level modeling of AMS multi-physics systems

A survey of the circuit modeling techniques that abstract reduced-order, higher level models from low-level descriptions is given in this section.

The approach we use is related to the modeling scenarios we want to address. The scenarios are listed in the following and sorted by levels of **starting knowledge** of the system. Two categories of systems are firstly identified:

- The physical laws defining the behavior of the device are known (see “modeling of knowledge” blue regroupment in Figure 4.1).
 - **Modeling of knowledge.** The structure of the device is known and the physical laws that govern the behavior of the device are mastered. The transistor level representation of an analog system is considered to be part of this case, since the laws ruling the devices have been exhaustively characterized by the silicon manufacturer from the technology point of view (violet branch in Figure 4.1).
 - **Structural knowledge.** If the device cannot be decomposed in elementary blocks connected as to form a netlist or a transistor level schematic, the analysis of the target device is typically performed through tools allowing to regard the device structure itself in its entirety. This is particularly true for micro electro-mechanical systems (MEMS) where a structural analysis of the system can be performed through *ad-hoc* tools based on the finite elements analysis (FEA). The analysis via these type of tools typically provide input-output response curves in the frequency domain when the analysis is carried out and a linearization is considered around a biasing point. Beside frequential response curves, non-linear responses can be obtained too such as the saturated attenuation of the magnitude of a cantilever vibration. As consequence, an accurate frequency-domain description of the device is obtained from a structural analysis performed by *ad-hoc* domain-specific simulators/tools (FEM for instance), this flow is represented by the blue branch in Figure 4.1.
- The physical laws that define the behavior of the device are not *a priori* known.

- **Experiment/Simulation-based modeling.** The device is physically available and experimental/simulation data only are available. The device is either available in the form of a hardware prototype IP thus experiments can be done, or in the form of a pre-compiled simulateable model, in both cases, it is considered as a black box with accessible inputs and outputs (see “Experiment/Simulation-based modeling” blue regroupment in Figure 4.1).

The three cases just introduced have been sorted by the type of source and starting knowledge of the user/designer. A second phase consists in analyzing the devices and take advantage of the available description in order to build a behavioral model. For this purpose the three flows of Figure 4.1 should be differently regrouped. In all cases (hereunder re-mentioned and re-sorted) a **mathematical model** needs to be built for describing the behavior of the device in order to perform simulations.

- A description of the system representing its netlist or schematic is available (see “From low level” orange regroupment in Figure 4.1).
 - **Modeling of knowledge.** The full knowledge of the system allows to calculate an analytical time-domain input/output equation, possibly involving state variables. Nevertheless, the possibility of having an exhaustive knowledge of the system is nowadays less and less probable because of the increasing complexity/coupling of the devices. Furthermore, if the analytical model is too complex in term of computational effort the order of the model can be reduced by means of Model Order Reduction (MOR) techniques known in the literature. A survey is available in [Mantooth 03] showing different techniques of behavioral modeling for analog circuits for Linear Time-Invariant (LTI), Linear Time-Varying (LTV)/Linear Periodically Time-Varying (LPTV) and Non-Linear (NL) systems.
- A data file making the relation between input and output is available (see “From a data set” orange regroupment in Figure 4.1).
 - **Structural knowledge.** A structural analysis can lead to an input-output response curve in the domain of the frequency composed of a set of points, without an analytical form. This set of data could be the only information available to model the system or a low-complexity model could be required. In both the cases, an approximation of the set of data could be done using a fitting by means of an equation with the desired order of complexity and frequency range of interest.
 - **Experiment/Simulation-based modeling.** The controllable input(s) is a degree of freedom that allows to identify the system and associate a mathematical equation. System identification techniques use statistical methods to build mathematical models of dynamic systems from measured data. System identification is particularly used in the field of automatic control and signal processing engineering and the basis principle is to stimulate both the model and the real system and to minimize the error between the outputs by identifying the parameters of a given starting equation (model structure).

Our research contributions addressed all of the three axes with different approaches. All the approach methodologies have the purpose to automatically individuate the available starting data and to define a methodology for easily creating behavioral models

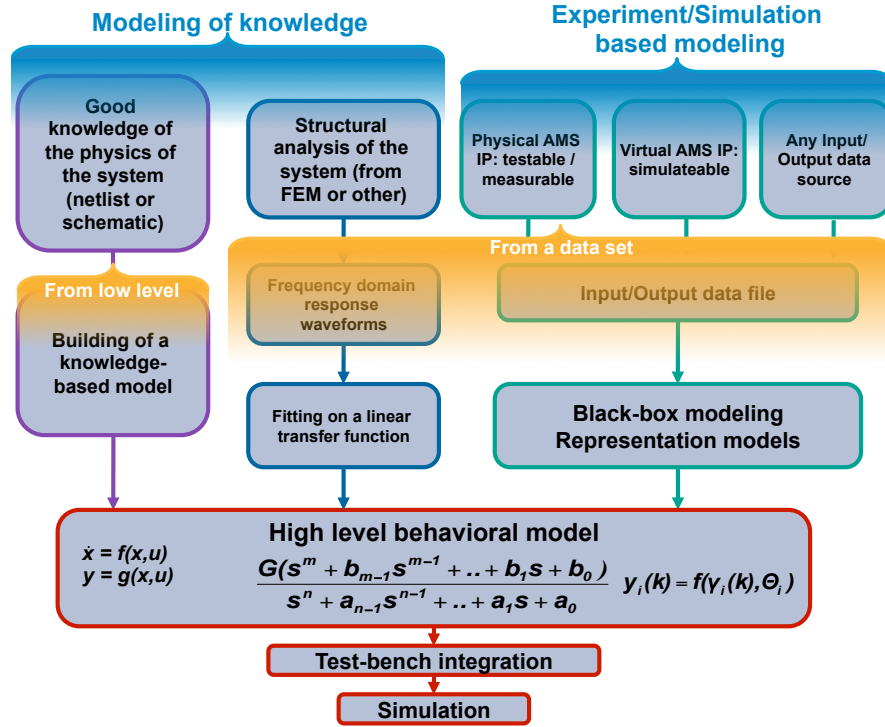


Figure 4.1.: Methodologies for analog behavioral modeling from starting knowledge.

written in an AHDL to be integrated in the *virtual test bench* at different levels of detail and launching the overall simulation. Depending on the level of abstraction required for the model of the analog system the three axes flows can be tuned for the desired level, a trade-off between model accuracy and model rapidity/simplicity must be accepted and found. More precisely and generally speaking, the analog device functioning is conditioned by the operational environment wherein it is inserted, it is mandatory that the modeling methodology focuses on two vital issues, that are: accuracy of the model in the Region of Operation (RoO) and the peculiar device behavior that the model is intended to capture and to reproduce.

- **Accuracy of the model in the Region of Operation (RoO).** The analog device functioning is conditioned by the operational environment wherein it is inserted. Most of the cases, the full-scale range behavior of the device is not completely stimulated/excited, the conditions of operation of the device itself are determined by the environment. In other terms, usually many internal modes of the device are not excited in normal operation. As a first remark, since the modeling effort has to be kept as low as required, a model must firstly capture/reproduce the device behavior inside the range defined by the operational conditions, the so called Region of Operation (RoO).
- **Desired behavior to be captured and reproduced by the model.** Generally speaking, the information/peculiarities of the behavior of the device that we desire to be captured by the model depend on the usage that is intended to be done with the model itself. The type of simulation, transient simulation or a frequency domain simulation, is certainly one of the criteria. Another criterion is the simulation purpose, as opposed to simple systems modeling when dealing with complex heterogeneous systems, simulations are carried out in order to validate only a part of the functioning of the overall system.

Such a part may consist of certain performances of the system that has to be checked for the compliancy to the specifications. Resuming, it is necessary to choose what aspects of the behavior must be captured by the model depending on what system performance the simulation is intended to validate.

In the following sections the three axes flows will be addressed separately and the works performed for each flow will be detailed as well as the corresponding study cases that validate the flows/concepts.

4.2. Contribution to the elaboration of knowledge-based high-level models from a netlist descriptive view

A descriptive starting point for some type of system may typically be a netlist of basic electrical components. Typical electrical circuit simulators allow to instantiate the symbolic view of electrical active or passive components from a library of predefined components and to compose them as to form a network, a selection of the circuit node voltages or branch currents to be plotted is then done and the transient or frequency domain simulations are launched. In the frame of our work these simulations are considered low-level simulations with quite complex electrical models of the primary components. The concerns addressed by our contribution are:

- **Extraction of a behavioral model from a netlist.** Electrical simulators first elaborate the instantiated network and build a system of differential equations then they solve the system keeping trace of the important nodes of the circuit defined by the user. When such networks has to be simulated in a higher-level environmental context, some simulators provide higher-level models only accessible/simulateable by means of their tool suite. There is typically neither simple nor standardized interface or API to their simulator in order to extract a behavioral model from the netlist information.
- **Customizable function of circuit voltages and currents for back-annotating noteworthy information.** Common electrical simulators typically do not provide a seamless and tool-independent way to extract user-defined noteworthy information (voltages or currents) of the circuit and to perform calculations from such data in order to obtain other desired information. More precisely, the reason that moved our researches to provide such a custom function is to monitor/exploit the power consumption information when simulating complex AMS systems.

Our research contribution was aimed at overcoming the two issues here explained. The extraction of a state space model was introduced by [Simeu 00] and further developed for the tool that we have proposed in [Bousquet 11b] and [Bousquet 11a]. The tool allows accessing to the power consumption information for models described at the behavioral level. For the moment the flow is limited to linear analog components and aims at a SystemC AMS-based modeling. A first possibility is to simulate the implementation-level description by means of the electrical linear network model of computation. This MoC gives the possibility to model and simulate linear analog sub-systems at the conservative level. It is possible to access the instantaneous supply current value and to raise it up to the time data flow MoC higher level of abstraction by means of specific ELN to TDF signal converters. This solution would reach a high level of details but the simulation would be computationally heavy, especially in the

case of very complex systems. For these reasons a level of abstraction higher than the one offered by the ELN MoC is needed. On the other hand, AMS modeling and simulation tools do not provide any information on the consumption of equipment simulated at high-level of abstractions (SystemC AMS Timed Data Flow (TDF) MoC). This information is available only when an implementation choice is made, and a lower level description model is available (ELN-based models). The idea proposed is to automatically extract both behavioral model and relevant power consumption information from a low-level description (netlist) of a linear circuit, and to reassemble them in the high level model so that the power information are propagated during the simulation. Thus, each simulation cycle, it will be possible to extract the supply electric currents associated with each component of the system and to use the result in order to estimate the power consumption of the overall system.

4.2.1. From the netlist to the state space equations with SystemC AMS

The type of systems we are focusing on are dynamic linear systems. These systems may be presented under two forms. The first one is the typical electrical circuit schematic well understood by electrical engineers, it is composed by connecting linear components such as resistors, capacitors and inductors. The second one is a mathematical representation of the dynamic linear system by means of, either a Laplace transfer function, or the state space equations. The state space model equations are shown below, Equation 4.1 is the state equation and Equation 4.2 is the output equation. u is the input vector, y is the output vector, x is the state vector and A , B , C and D are the time invariant matrices of the system in a higher level model.

$$\dot{x} = A \cdot x(t) + B \cdot u(t) \quad (4.1)$$

$$y(t) = C \cdot x(t) + D \cdot u(t) \quad (4.2)$$

The state space model can be regarded as a block diagram where the vectors are represented as signals and the matrices are the blocks. Figure 4.2 shows the state space model block diagram of the system.

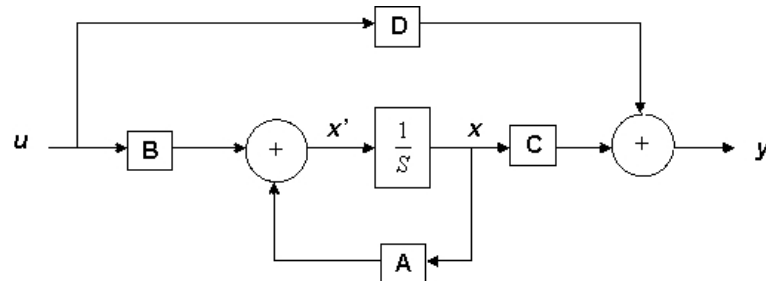


Figure 4.2.: State space-based block diagram of a dynamic linear system.

The state space model and the transfer function (Equation 4.3) are the two equivalent ways of modeling a continuous time linear system. The relationship between these two representations is given in Equation 4.4. Where $L(s)$ is $q \times p$ transfer function matrix between $u(t)$ and $y(t)$, meaning that a transfer function is needed for each output variable of the Y array.

$$L(s) = \frac{Y(s)}{U(s)} = K \frac{b_m s^m + b_{m-1} s^{m-1} + \dots + b_1 s + b_0}{a_n s^n + a_{n-1} s^{n-1} + \dots + a_1 s + a_0}, n \geq m \quad (4.3)$$

$$L(s) = C(sI - A)^{-1}B + D \quad (4.4)$$

With respect to the SystemC AMS modeling formalisms, a dynamic linear system described by an electrical network is directly instantiatable by connecting ELN primitives. With respect to the mathematical representation of a dynamic linear system, SystemC AMS allows to use Laplace or state space functions by means of the LSF or TDF MoCs. In the case of the LSF MoC the code in listing 4.1 shows the instantiation of a state space represented model according to the terminology of Figure 4.2.

Listing 4.1: State space model using the LSF MoC

```
1 sca_lsf::sca_ss eqs("eqs", A, B, C, D);
2   eqs.u(u);
3   eqs.y(y);
```

The `sca_lsf::sca_ss` class has two interfacing LSF signals (input `u` and output `y`) and receives the four matrices (**A**, **B**, **C** and **D** of the state space model) as input parameters. The idea is to analyze the electrical circuit topology in order to extract an extended state space model that contains the power consumption information. In the following, it will be explained how the matrices **A**, **B**, **C** and **D** of Equations 4.1 and 4.2 are generated from a SystemC AMS circuit description or SPICE netlist. Three main steps are identified: the generation of a matrix representing the circuit (the so called circuitual matrix), the arrangement of the circuitual matrix and the extraction of the state space matrices.

In the following, the three steps of the flow are illustrated. A case study circuit is used for better explaining the flow, the circuit is shown in Figure 4.3(a). The *first step* consists of the **extraction of the circuitual matrix**. The matrix is obtained in the form of Figure 4.3(b) by analyzing the circuit topology, no choices of the desired information to be extracted are done at this step. The circuitual matrix represents a set of equation where the variable on the left (one for each row) is expressed as a linear function of the column variables. Among the independent row variables only the inputs, the state variables and their derivatives are needed to build the state space model. All the other variables are undesirable (currents through capacitors, voltages across inductors).

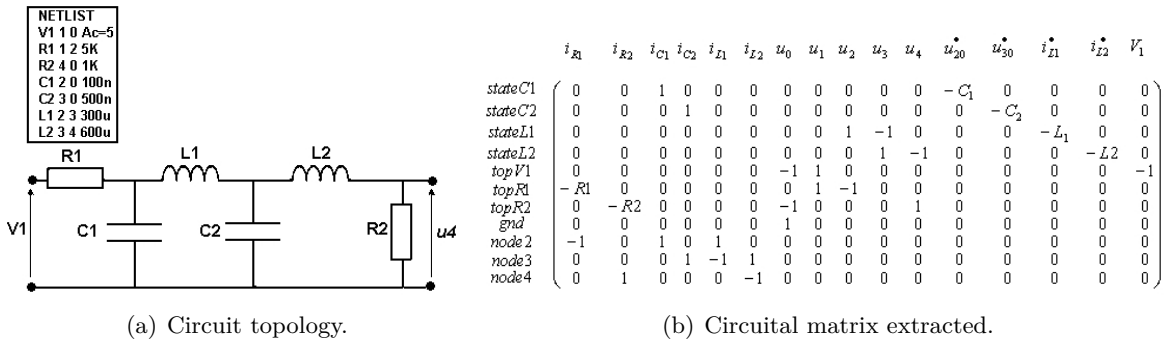


Figure 4.3.: Case study.

The *second step* consists of a **rearrangement of the circuital matrix** resulting in the structure of Figure 4.4(a). For the case study the result of the rearrangement step is shown in Figure 4.4(b).

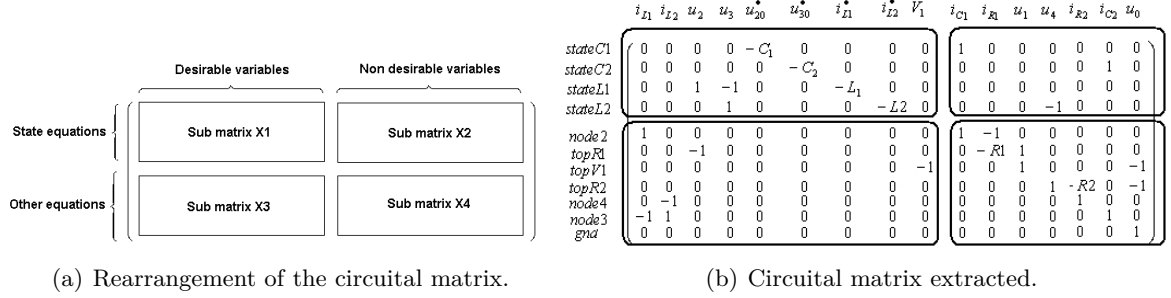


Figure 4.4.: Rearrangement step result.

The non-null terms of sub-matrix X2 must be nulled by expressing the dependency of the state equations as function of the desirable variables only (column variables of sub-matrix X1). An intermediate matrix called **relation matrix** is built (Figure 4.5(a)) and the matrices **A** and **B** of the state space model are directly obtained from sub-matrix X1. The **A** and **B** matrices are shown in Figure 4.5(b).

Now a choice must be done on which node voltage or branch current has to be regarded as output of the state space model. We choose u_2 and u_4 as output variables. We also want the model to output the information on the absorbed input current too, the input current of this passive case study circuit include the information on the power consumption. Therefore we also choose i_{R1} as output of the system. For building the matrices **C** and **D** of the state space model the dependency of the chosen outputs to the state variables and inputs. In the case of u_2 it is trivial since it equals to u_{20} . For u_4 and i_{R1} their dependencies are directly obtained from the relation matrix of Figure 4.5(a).

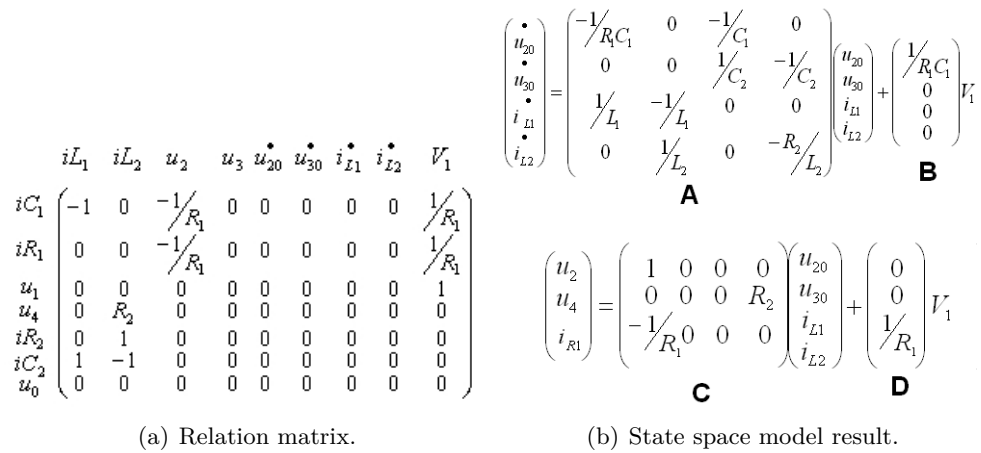


Figure 4.5.: State space model.

The result is a state space model, thus a model described at a higher level of abstraction, that does not lose any information or make approximations. Variables that define containing

the power consumption information are extracted as outputs of the model and can be used to perform an additional processing such as a multiplication to node voltages or a time integration. The key concept is that it is possible to regard any of the variables of a circuit as an output of the state space model, and this requires only the matrices **C** and **D** to be modified. The matrices **A** and **B** are unchanged since they are not directly in relation with the output vector. Thanks to this flow a SystemC AMS component implementing the state space model in the LSF MoC is directly obtained from the netlist of the circuit.

More technically speaking, at the moment of the construction of the SystemC AMS module the automatized flow allows reading an external netlist file and specifying the desired information to be monitored. During the construction the state space is calculated and during the simulation it is evaluated/solved. Additionally to the normal input/outputs of the state space model a further LSF output port is instantiated. The user will decide how to exploit such an information. The LSF signal could be treated by using LSF primitives such as derivators or integrators. Otherwise, a conversion to a TDF signal could be done in order to multiply the signal by another TDF signal, this could be useful for example for calculating the instantaneous power consumption (multiplication between voltage and current values).

Further details on the methodology can be found in [Bousquet 11b] and [Bousquet 11a]. As aforementioned, using Equation 4.4 it is possible to obtain different transfer functions with the same denominator for each defined output of the state space model.

4.3. Contribution to the elaboration of knowledge-based models from simulation data

The title of this section is quite general. This section concerns the techniques for building models from simulation data obtained from other knowledge based low-level models, as opposed to the black box system identification based modeling from simulation data obtained from empirical data (section 4.4). Sub-section 4.3.1 will describe the adopted technique for modeling from frequency domain response curves for dynamic linear components. Sub-section 4.3.2 will deal with the adopted modeling technique from lower-level model simulation results for static non-linear components.

The methodology will be shown together with the tools used for providing an automated flow to a reduced order model. Such a model is directly instantiable in a transistor level or system level test bench for the simulation.

4.3.1. Modeling from frequency domain response curves for dynamic components

As mentioned in section 4.1, domain-specific tools based on FEA are typically used for analyzing MEMSs during the design phase. A number of analytical and numerical techniques to handle MEMS macromodel have appeared over the last decade. The purpose is to accurately describe MEMS behaviors and to take into account changes of design parameters. In our work we intend to keep a system level view. The models must be rapid and capture the strictly necessary information. The choice of having such a coarse granularity in modeling is driven by the matter of fact that impressively big amounts of post-processing are now performed by SW

algorithms mapped into HW architectures, the target is to enable a system level simulation of the overall platform. We do not intend to deal with FEAs, the concept is to collect and elaborate FEA simulation results for obtaining a functional model which is rapidly integratable in a transistor-level or system-level environment.

A structural analysis can lead to an input-output response curve in the domain of the frequency composed of a set of points, without an analytical form. This set of data could be the only information available to model the system or a low-complexity model could be required. In both the cases, a linear approximation of the set of data could be done using a fitting by means of an equation with the desired order of complexity and frequency range of interest. Keeping in mind the purpose to obtain a Laplace transfer function several frequency fitting techniques exist in the literature. The advantage of the approximation of the input-output behavior is that the accuracy of the model can be tuned by varying the order of the transfer function and the region of higher accuracy can be tuned too typically using a weight function, that becomes a weight array when discretized in the frequency domain.

Generation of dynamic macro-models

In practice, for most dynamic blocks, we do not have an analytical representation of their behavior. The behavior is most often obtained via a low level simulation of the analog block (typically transistor-level simulation). For obtaining a macro-model of this block, we can approximate the frequency response of the device using a fitted function, with high accuracy in the region of frequency operation of the block. This fitted function reduces the complexity of the actual transfer function, providing a computationally lighter function with a lower number of poles and zeros. This approach can also be used in the case that the analytical transfer function is known. A fitted function in this case can help to provide a simpler model for simulation in the region of frequency operation.

This fitting is performed as follows. We start from the frequency response in the form of an array of complex values. The Matlab Rational FittingTM function is used to obtain a fitted version of the input frequency response. An array of weights must be provided to the fitting function in order to specify the frequency region with the highest accuracy. The result of the fitting is a Laplace transform obtained in the form of Equation 4.5.

$$F(s) = \left(\sum_{i=1}^{n/2} \left(\frac{C_i}{s - A_i} + \frac{\overline{C_i}}{s - \overline{A_i}} \right) + D \right) \cdot e^{-s \cdot Delay} \quad (4.5)$$

The level of accuracy of the fitting, thus the number of poles of the Laplace transfer function, depends on the value of “n” and the range of the frequency region of interest. A typical call to the fitting function is as follows:

$$rational_object = rationalfit(f, H(f), \varepsilon, w, delayfactor, diszero, n)$$

where “f” is the array of frequency values and “H(f)” is the array of the complex values related to the frequency array. The “ ε ” parameter is the tolerance that must be reached by the fitting and “w” is the array of weights for the frequency range. The “delayfactor” parameter is a

scaling factor between 0 and 1 that controls the amount of the “Delay” value used to fit the data. The “diszero” value is a boolean value that specifies whether the constant term “D” in Equation 4.5 is zero or nonzero. Finally “n” is an even integer number as it appears in Equation 4.5. The result of the function is a “rational object” belonging to the Matlab RF Toolbox™ it contains each parameter of Equation 4.5 thus “n/2” values of A_i and “n/2” values of C_i .

Macro-modeling of dynamic components for SystemC AMS simulation

For dynamic blocks, the values of the outputs depend on the values of the inputs at the current time but also at previous time instants. This type of behavior can be modeled in SystemC AMS using the TDF or the ELN MoCs. In both cases, with the current version of the language, only linear systems can be modeled. The modeling of non-linear dynamic blocks is only possible by using a linearization around an operating point. The ELN MoC is in addition limited to the use of basic building blocks available in the libraries. On the other hand, the TDF MoC can be used to represent any linear behavior in terms of its Laplace transfer function or state space model. The approach presented is shown in Figure 4.6 and uses the TDF MoC. If the Laplace transfer function of the device is known, this can be readily instantiated in SystemC AMS using a pole/zero or polynomial representation.

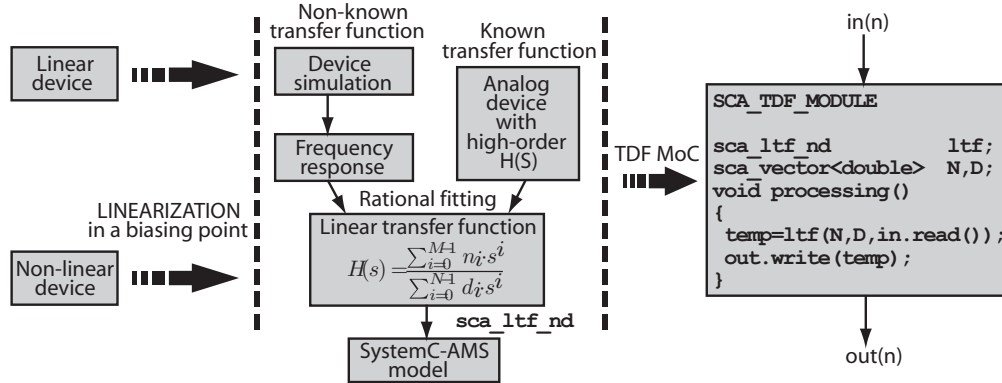


Figure 4.6.: SystemC AMS macro-modeling of dynamic components by means of a linear approximation.

The listing 4.2 below illustrates how the instantiation is done for a transfer function using the polynomial representation of Equation 4.6. Where *num* and *den* are vectors containing respectively the n_i and d_i values. The instantiation is performed by exploiting the “sca_tdf::sca_ltf_nd” class (stands for Linear Transfer Function in the form of Numerator-Denominator) that is provided among the tool libraries of SystemC AMS.

Listing 4.2: Modeling of a dynamic behavior by using the “sca_ltf_nd” construct for Laplace transfer functions.

```

1 SCA_TDF_MODULE(transfer_function) {
2   sca_tdf::sca_ltf_nd      ltf_1;
3   sca_tdf::sca_in<double>  in;
4   sca_tdf::sca_out<double> out;
5   sca_util::sca_vector<double> num, den;
6   double                  k;
7

```

```

8  void initialize ()
9  {
10 num(0) = ...;
11 den(0) = ...;
12 den(1) = ...;
13 }
14
15 void processing() {
16     double tmp = ltf_1(num, den, in.read(), k);
17     out.write(tmp);
18 }
19 ...
20 };

```

$$H(s) = k \cdot \frac{\sum_{i=0}^{M-1} num_i \cdot s^i}{\sum_{i=0}^{N-1} den_i \cdot s^i} \cdot e^{-s \cdot delay} \quad (4.6)$$

A MatlabTM script that converts the fitting results, in terms of parameters of Equation 4.5, to the SystemC AMS code has been developed, it is shown in Annex A.3. Therefore the starting point for the SystemC AMS model generation are the A_i and C_i complex numbers while the “D” and “Delay” values are not taken into account in this work as if they were both at zero. In order to handle a Laplace transfer function within the SystemC AMS environment the “sca_ltf_nd” class is exploited. The latter is provided among the tool libraries of SystemC AMS and it implements a transfer function in form of Equation 4.6.

For the conversion of the Laplace transform of Equation 4.5 to the SystemC AMS compatible form of Equation 4.6, it must be considered that the “sca_ltf_nd” construct cannot accept complex numbers at its input, therefore it is not allowed to pass the A_i and C_i values as its arguments. Thus, the couples of elements of Equation 4.5 are added up in order to obtain the transfer function of Equation 4.7.

$$F(s) = \sum_{i=1}^{n/2} \left(\frac{N_{0,i} + N_{1,i} \cdot s}{D_{0,i} + D_{1,i} \cdot s + s^2} \right) \quad (4.7)$$

with:

$$\frac{N_{0,i} + N_{1,i} \cdot s}{D_{0,i} + D_{1,i} \cdot s + s^2} = \frac{C_i}{s - A_i} + \frac{\overline{C_i}}{s - \overline{A_i}} \quad (4.8)$$

Here, the $N_{k,i}$ and $D_{k,i}$ values are real hence allowing the use of the SystemC AMS “sca_ltf_nd” construct. Subsequently, the “ $\delta=n/2$ ” instances of the 2-pole transfer function module referring to the elements of Equation 4.8 are instantiated. These 2-pole modules are called “fract_i” and the corresponding SystemC AMS module is denominated “2poles_tf”. These instances are arranged in parallel, that is, they have the same overall input and the outputs are added up as shown in Figure 4.7.

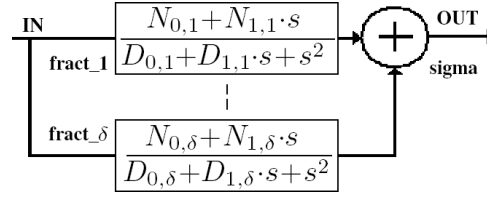


Figure 4.7.: Structure of the n-poles model.

The two values of N_k and the two values of D_k needed for each one of the “n/2” instances are obtained from the A_i and C_i values generated from the fitting, as shown in Equation 4.8. Listing 4.3 shows the code required for the instantiation of the “fract_i” modules and the sum of their outputs.

Listing 4.3: SystemC AMS code of the n-poles model.

```

1 SC_MODULE(npoles_model) {
2     2poles_tf*   fract_1;
3     2poles_tf*   fract_2;
4     ...
5     adder*       sigma;
6     SC_CTOR(npoles_model) {
7         fract_1 = new 2poles_tf("fract_1");
8         fract_1->in(input);
9         fract_1->out(output1);
10        fract_1->N0 = ...;
11        fract_1->N1 = ...;
12        fract_1->D0 = ...;
13        fract_1->D1 = ...;
14        fract_1->D2 = 1.0;
15        fract_2 = new 2poles_tf("fract_2");
16        ...
17        sigma = new adder("sigma");
18        sigma->in1(output1);
19        sigma->in2(output2);
20        ...
21    }
22 };

```

In summary, the generation of a dynamic macro-model in SystemC AMS from its frequency response or a complex transfer function is sketched in Figure 4.8. The fitting provides a transfer function decomposed in elements of first order. The converter script receives at its input the result of the fitting. It provides a transfer function decomposed in elements of second order. Subsequently it generates the SystemC AMS file by first defining the “2poles_tf” module, then defining the “adder” module and finally instantiating them in the “npoles_model” module. The latter is the SC_MODULE that has to be instantiated in the SystemC AMS environment in order to be simulated together with the overall system. The code of the Matlab generation script is shown in Annex A.3 while the code of the SystemC AMS generated model is shown in Annex A.4.

In section 4.6, the methodology is validated with an advanced case study of the design of a Phase Locked Loop (PLL) based microelectronics frontend interface for a Surface Acoustic Wave (SAW) based chemical sensor.

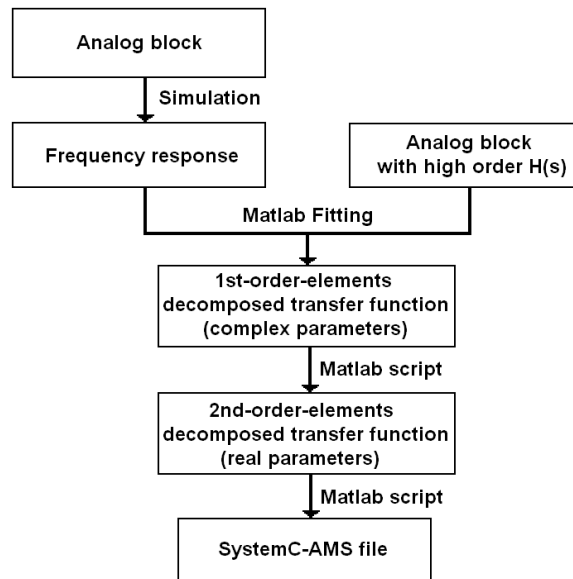


Figure 4.8.: SystemC AMS model generation flow.

4.3.2. Macro-modeling of static components for SystemC AMS simulations

With respect to static blocks, the output at time “t” depends only on the value of the inputs at the same time “t” as in function $Y(t) = f(X(t))$ where “f” has an arbitrary form, often non-linear. Therefore there is no memory involved. The function “f” could have severe non-linearities and in some case an analytical expression of “f” is not known, thus only a set of points or graphical representation is available. In these cases the function “f” can be approximated by a simple analytical equation obtained by different methods as interpolation, fitting approximation, Taylor expansion, etc. Figure 4.9 shows the flow used for static non-linear analog components, which could also be applied to the trivial case of static linear behaviors.

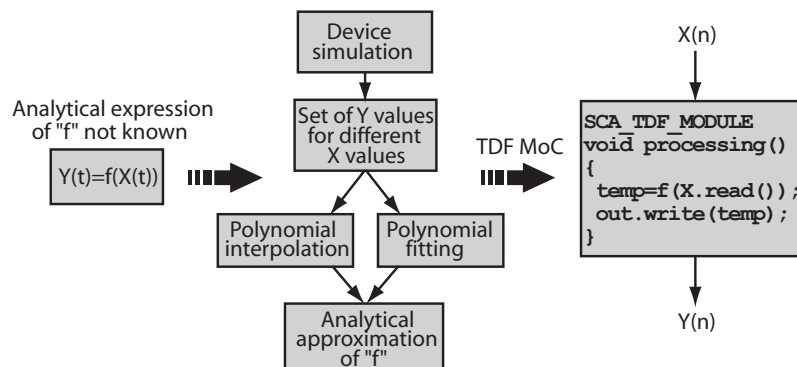


Figure 4.9.: SystemC AMS macro-modeling of static non-linear components.

This type of behavior, described by an analytical equation, can be directly modeled with SystemC AMS using the TDF MoC. The listing 4.4 illustrates the code required for implementing this behavior. When the actual model is called, a signal processing function of the TDF module

reads the input values, next calculates the output values from the current inputs, and finally writes out the output values.

Listing 4.4: Modeling of static behaviors using SCA_TDF_MODULES.

```

1 SCA_TDF_MODULE(...) {
2 ...
3 void processing() {
4     double input = in.read();
5     double tmp = function(input);
6     out.write(tmp);
7 }
8 ...
9 };

```

In section 4.6, the methodology is used for the modeling of the Voltage Controlled Oscillator (VCO) of the Phase Locked Loop (PLL) frontend interface for a Surface Acoustic Wave (SAW) based chemical sensor.

4.4. Contribution to the elaboration of black-box models from empirical data

The key concept of the black box modeling is that the device itself or a model of it is available with no knowledge about the internal structure. Therefore experimental/simulation data only are available. The device is either available in the form of a prototype IP hence experiments can be done, or in the form of a pre-compiled low-level simulateable model (for example a transistor level simulation of an analog circuit pre or post layout). In both cases the model/prototype is considered as a black box with accessible inputs and outputs and has to be stimulated at its input ports for obtaining the output values. The “Experiment/Simulation-based modeling” blue regroupment in Figure 4.1 shows the achievement of a behavioral model by means of data issued from measures or simulation data. It is claimed that starting from sets of uniformly sampled input/output waveform values it is possible to obtain a behavioral model, also called representation model, described with the generic equation $y(k) = f(\gamma(k), \Theta)$, where y is the output, γ is the regression vector and Θ are the model parameters.

Choice of the input stimuli and sampling frequency. The controllable input(s) is the only degree of freedom. The choice of the well suited stimuli must be carefully done. For instance, the performance estimation of a microelectronics device under test, essentially depends on the choice of the input stimuli (stimulation signal). The stimuli must cover a wide band of frequency in order to stimulate and excite the different functioning modes of the circuit. If the system identification technique is aimed at testing the functioning of the device, a good choice allows to detect a variety of faults during the test, thus leading to a good fault diagnosis. Typically, circuit faults are not all detectable at the same operating frequency. This condition is mostly verified by using a gaussian or uniformly distributed random signal. In the control field, a commonly used stimuli are pseudo-random binary sequences (PRBSs) with low amplitude that are sequences of rectangular pulses modulated in their width, they approximate a discrete white noise. A condition that has to be fulfilled by the input stimuli is the so called persistent excitation, that means that the regression vector (internal states with memory) has to perform

a periodic sweep of the space. For respecting this, it is sufficient that the input power is distributed on a number of frequencies at least equal to the dimension of the regression vector.

Together with an input signal trend a sampling frequency has to be specified too. The Shannon theorem must be respected and the sampling frequency must be at least two times greater than the presumed maximum frequency of the system bandwidth. Higher sampling frequency ensures a better identification but 10 times the presumed bandwidth is typically sufficient, no advantages are carried out from going beyond $10 \cdot f_{MAX}$.

Data file availability. As already mentioned we assume that we are able to inject a continuous time input signal (PRBS for instance) in the simulatable model (from now on called virtual prototype) or in the real prototype (called physical prototype) and to collect the output signal. These sequences are saved in a data file and those data are the starting point for identifying the system.

4.4.1. System identification, model structures, parameters and criteria.

Citing Professor Ljung in [Ljung 08]: “*System identification is the art and science of building mathematical models of dynamic systems from observed input-output data. It can be seen as the interface between the real world of applications and the mathematical world of control theory and model abstractions. As such, it is an ubiquitous necessity for successful applications. System identification is a very large topic, with different techniques that depend on the character of the models to be estimated: linear, nonlinear, hybrid, nonparametric etc.*”

The process of constructing models from experimental data is called system identification. These experimentally derived models are not intended to explain the physical system in totality or in any meaningful way. In order to control a system we must have a model of its behavior, understanding the details of that behavior is useful but unnecessary. We need a model adequate to develop a controller for the real system which provides stability and the desired performance.

System identification techniques use statistical methods to build mathematical models of dynamic systems from measured data. System identification is particularly used in the field of automatic control and signal processing engineering and the basis principle is to stimulate both the model and the real system and to minimize the error between the outputs by identifying the parameters of a given starting equation (model structure). The terms “system identification” do not include the identification of the model structure but only the parameters of a given model architecture. A parametric model is firstly introduced, second the error between model and the system has to be defined, finally, a criterion to be optimized in order to find the best parameters for the parametrical model.

The system identification principle is shown in Figure 4.10, $u(t)$ is the system input at the instant t . $\hat{\varepsilon}(t)$ is the error committed by approximating the system output $y(t)$ with $\hat{y}(t)$. g is the unknown function representing the real system and \hat{g} is the mathematical estimated model. θ is the vector of unknown parameters of the system and $\hat{\theta}$ is the vector containing their estimators. The system identification aims at determining the g function of Equation 4.9 where $\gamma(t)$ is the regression vector. Each element of the regression vector is a function of u or y or a combination of them at different temporal instants. The regression vector can be written as shown in Equation 4.10.

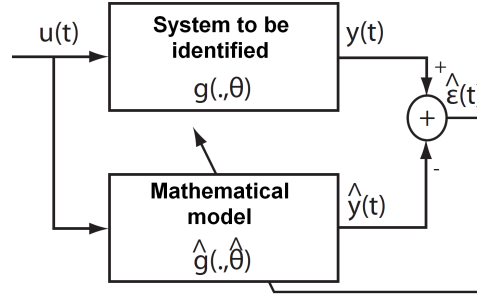


Figure 4.10.: System identification principle.

$$y(t) = g(\gamma(t), \theta) \quad (4.9)$$

$$\gamma(t) = \{f(y(t - d_y), u(t - d_u))\}, d_y > 0, d_u \geq 0 \quad (4.10)$$

$$\hat{y}(t) = g(\hat{\gamma}(t), \hat{\theta}) \quad (4.11)$$

$$y(t) - \hat{y}(t) = \hat{\varepsilon}(t) \quad (4.12)$$

The prediction of the output value $y(t)$ is done by estimating both the structure of the function $g(\gamma(t), \theta)$ and its parameters θ as shown in Equation 4.11. The prediction error is given by Equation 4.12. In order to identify linear systems a large number of well known methods are available, these techniques give an approximation of the reality since in most cases systems are non linear. Non linear phenomena are not easily treatable, the distortions observed at the output can have different natures. In the case of weak non linearities it is possible to neglect them by approximating the system with a linear model. When distortions are strong they must be identified and modeled. The best model structure that approximates the real system has to be found and the parameters of the model estimated.

The main steps typically are:

- the choice of the model structure \hat{g} that includes the choice of the regression vector γ and the estimation of the model complexity
- definition of the algorithm for parameter estimation

Model structure. Different model structures appear in the literature, they can be linear, non-linear, static or dynamic, continuous or discrete in time, determinist or stochastic [Walter 94]. This thesis will deal with models that are discrete in time (numerical simulations make these models simple and fast, they are particularly suited for real time applications), dynamic (in most cases systems have dynamic behaviors), stochastic (they also consider possible perturbations or noise not dependent on the input values).

If we consider the dynamic system of Figure 4.11 we denote with n_u the memory effect on the input and respectively n_y on the output. In the case of a single-input single-output (SISO) system. The input/output relationship can be written as in Equation 4.13, the models with this formulation are called Auto-Regressive with eXogenous variable (ARX) models. The output value $y(t)$ can also be expressed as a function of the regression vector as shown in Equation 4.14.

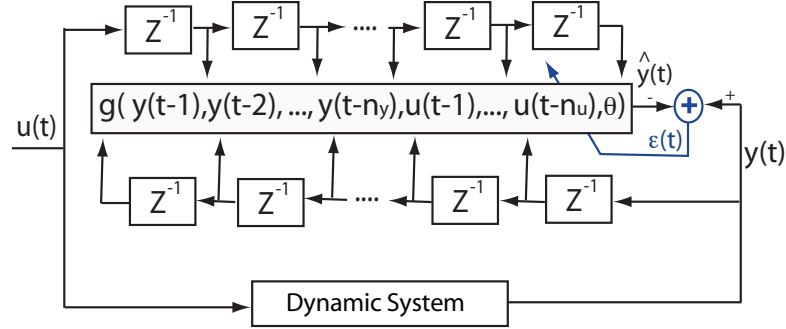


Figure 4.11.: Modeling of a dynamic system.

$$y(t) = \sum_{i=1}^{n_y} a_i y(t-i) + \sum_{j=1}^{n_u} b_j u(t-j) + \varepsilon(t) \quad (4.13)$$

$$\{a_i, i = 1, \dots, n_y\} \{b_j, j = 1, \dots, n_u\} \quad (4.14)$$

$$y(t) = \gamma^T(t) \theta + \varepsilon(t)$$

Optimization criterion. Once the parametric model structure has been defined, the set of parameters that best fit the system behavior has to be found. For this purpose a criterion has to be defined and optimized. The criterion $J(\theta)$ is a scalar function of the parameters, it has to be optimized (minimized or maximized) and the optimal value of the criterion is given by the best model $\hat{J}(\hat{\theta})$. Many criteria are used in the literature such as quadratic, absolute value, maximum of likelihood, Bayesian, etc... We will focus on the **quadratic criteria** since they are the most used thanks to their intuitiveness and to the fact that they are well suited for the demanded optimization calculations. The criterion can be written as shown in Equation 4.15 where $\varepsilon(\theta)$ is the error vector depending on the parameters set and \mathbf{Q} is an averaging matrix defined non negative. Very often \mathbf{Q} is chosen as diagonal with the w_i elements on its diagonal, the criterion can then be expressed as in Equation 4.15 where i is the index corresponding at the sampling time. The optimal criterion value (Equation 4.16) is given by the best parameter set $\hat{\theta}$ (called *estimator*), then reversely the optimal parameter set is given by Equation 4.17.

$$J_{squares} = \varepsilon^T(\theta) \mathbf{Q} \varepsilon(\theta) = \frac{1}{N} \sum_{i=0}^{N-1} w_i [y(t_i) - \hat{y}(t_i)]^2 \quad (4.15)$$

$$\hat{J} = J(\hat{\theta}) = \text{Min}_{\theta} J(\theta) \quad (4.16)$$

$$\hat{\theta} = \text{Arg}[\text{Min}_{\theta} J(\theta)] \quad (4.17)$$

The most typical estimator is the one that minimizes the mean squares. The **Least Mean Squares** (LMS) estimator is obtained by replacing the \mathbf{Q} matrix of Equation 4.15 with the identity matrix so that the terms w_i are ones. The algorithm used for solving the LMS problem is well known in the literature and it is depicted hereunder in Equations 4.20 to 4.22. The general structure for representing a linear system is shown in Equation 4.18.

$$Y = X\theta + \varepsilon \quad (4.18)$$

Where:

$$Y = \begin{pmatrix} y(t) \\ \vdots \\ y(1) \end{pmatrix}, X = \begin{pmatrix} \gamma(t)^T \\ \vdots \\ \gamma(1)^T \end{pmatrix}, \theta = \begin{pmatrix} \theta_1 \\ \vdots \\ \theta_p \end{pmatrix}, \varepsilon = \begin{pmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_p \end{pmatrix} \quad (4.19)$$

The regression vector is defined in Equation 4.10. The parameters of the model are estimated using the simple ordinary LMS algorithm, its principle is to minimize the sum of the error squares (see Equation 4.15). The criterion function to be minimize is given by Equation 4.21. The value of θ that minimizes J , on the condition that the matrix is invertible, is given by Equation 4.22. $\hat{\theta}$ is the unbiased estimator of θ meaning that $E(\hat{\theta}) = \theta$ where E is the expected value.

$$\varepsilon = Y - X\theta \quad (4.20)$$

$$J = \varepsilon^T \varepsilon = (Y - X\theta)^T (Y - X\theta) \quad (4.21)$$

$$\hat{\theta} = (X^T X)^{-1} X^T Y \quad (4.22)$$

One of the main concerns when identifying systems is the speed of the algorithm since a typical application is the on-line control of the system itself. The ordinary LMS algorithm shown above needs the input and output values to be available before performing the solving of the system and the matrix $(X^T X)^{-1} X^T$ changes at every sampling step.

The **Recursive Least Mean Square** (RLMS) estimation of the model parameters can be performed using the equations shown below.

Initialization:

$$t = 0;$$

$$\hat{\theta}_0 = 0;$$

$$P_0 = \sigma^{-1} I; (\sigma \text{ is a positive constant, } I \text{ is the identity matrix})$$

For $t = 1, 2, \dots$:

$$K_t = \frac{P_{t-1} \cdot \gamma(t)}{1 + \gamma^T(t) \cdot P_{t-1} \cdot \gamma(t)} \quad (4.23)$$

$$P_t = P_{t-1} - \frac{P_{t-1} \cdot \gamma(t) \cdot \gamma^T(t) \cdot P_{t-1}}{1 + \gamma^T(t) \cdot P_{t-1} \cdot \gamma(t)} \quad (4.24)$$

$$\hat{\varepsilon}(t) = y(t) - \gamma^T(t) \cdot \hat{\theta}_{t-1} \quad (4.25)$$

$$\hat{\theta}_t = \hat{\theta}_{t-1} + K_t \cdot \hat{\varepsilon}(t) \quad (4.26)$$

If the stop condition is fulfilled the algorithm stops otherwise it is necessary to perform other iterations. In the case of the adaptive control of on-line systems the stop condition is not valid since the identification is performed as long as the control procedure is active.

Penalization of the model complexity. It is often the case where the complexity of the model has to be kept as low as possible, there may be two main motivations of this limit. The first is the limited resources onto which the model identification/simulation has to run, in terms of memory or processing units. In particular this may be the case for embedded system identification. The second reason is the main purpose of the thesis: to keep the model complexity as low as possible all the while ensuring to represent its functioning with enough details, the purpose remains to be able to simulate an entire AMS SoC at a high level of abstraction.

Resuming, one key point when identifying systems is to prevent the explosion of the complexity of the model by penalizing the dimension of the regression vector. This penalization is typically implemented by adding a term in the criterion to be optimized as shown in Equation 4.27. The new overall criterion is called J_{TOT} , it is given by two contributions where $\alpha \in [0, 1]$ is a parameter that allows to make a trade-off between the accuracy of the model and its complexity S . S is defined as shown in Equation 4.28, $Weight(i)$ is an array of weights, one for each element of the regression vector (*regressor*). Typically, $Weight(i)$ is higher for i corresponding to high delays in order to penalize high memory models, but for simplifying $Weight(i)$ can be considered at 1 resulting in $S = dim(\theta)$.

$$J_{TOT}(\theta) = \alpha(J(\theta)) + (1 - \alpha)S \quad (4.27)$$

$$S = \sum_{i=0}^{dim(\theta)-1} (Weight(i)) \quad (4.28)$$

In subsection 4.4.2 it is proposed an automated tool for the identification of a given system and extraction of a model suitable for SystemC AMS simulations. The tool takes into account a system identification based on the LMS criterion and implements a trade-off between accuracy and complexity. A simple case study will be used as illustrative example.

Our contribution to the elaboration of black-box models from empirical data addressed a Low Noise Amplifier (LNA) case study. Section 4.5 will show how a system identification based methodology allows to estimate the performances of the identified analog system, an LNA in our case-study, for test and control purposes.

4.4.2. Proposed extension of SystemC AMS libraries for building identification models

An extension to the SystemC AMS libraries allows to automate the flow of our modeling scenario, from a data file of input/output sampled waveforms to the construction and instantiation of a SystemC AMS suited model of the identified system. If we consider a system with input U and output Y , we aim at identifying a model in the form of Equation 4.29 where the error is denoted by Equation 4.30.

$$\hat{y}(t) = a_1 \cdot y(t-1) + \dots + a_{na} \cdot y(t-na) + b_1 \cdot u(t-nk) + \dots + b_{nb} \cdot u(t-nk-nb) \quad (4.29)$$

$$\hat{y}(t) - y(t) = \varepsilon(t) \quad (4.30)$$

$$J_{TOT} = \alpha \left(\sum_i \varepsilon^2(t_i) \right) + (1 - \alpha)(na + nb) \quad (4.31)$$

Figure 4.12 shows in three axis the methodology at different levels of details. On the left is the concept, we start from the simulation of a detailed model of the device and obtain an ARX model (Equation 4.29). On the center is a possible scenario wherein the waveform results of a transistor level transient simulation are used to obtain the vector of the parameters Θ . The model structure is given by Equation 4.29, na and nb are unknown and the minimization of the criterion in Equation 4.31 is obtained by searching the space of the architecture (na, nb) for the lowest value of J_{TOT} . The LMS algorithm is launched for each combination of $na \in [1, na_{MAX}]$ and $nb \in [1, nb_{MAX}]$ and the model details ($\Theta = [a_1, \dots, a_{na}, b_1, \dots, b_{nb}]$ and na, nb) corresponding to the lowest J_{TOT} are retained and a SystemC AMS model using the Timed Data Flow (TDF) Model of Computation (MoC) is instantiated.

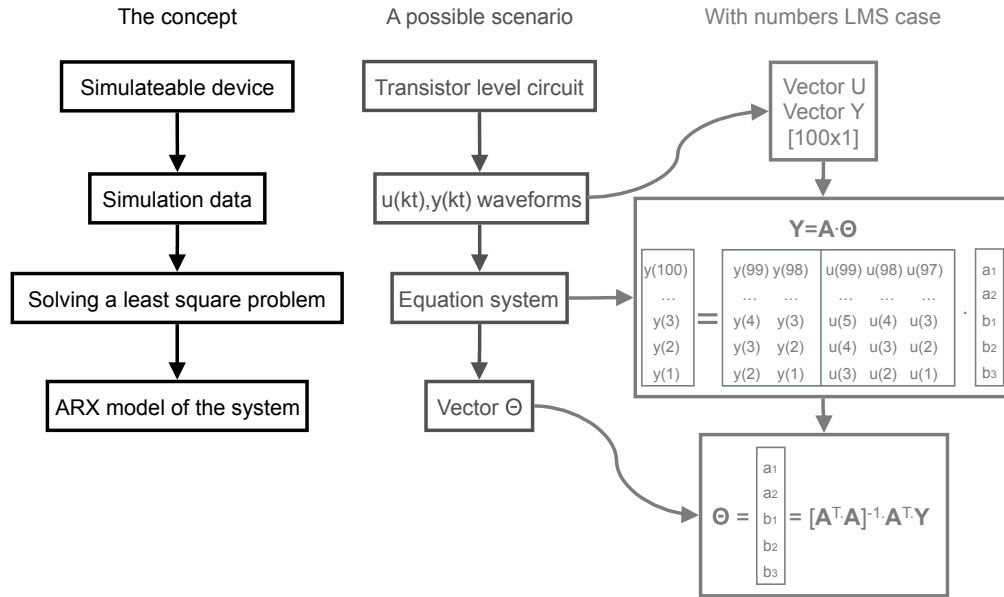


Figure 4.12.: The concept of system identification extensions to SystemC AMS.

Figure 4.13 shows the test bench and how the methodology is implemented in SystemC AMS. The following steps are performed:

1. The ARX TDF predefined module is instantiated with different argument sets (see the possibilities in Listing 4.5).
2. During the construction of the ARX TDF module the waveforms data file is loaded.
3. The loaded waveform data are used to calculate the system identification and the resulting J_{TOT} for the starting combination of na, nb, nk .

4. The system identification and the corresponding J_{TOT} is then calculated for all the combinations of na, nb, nk . The maximum number of parameters specified as arguments in the constructor (see line 3 in Listing 4.5) sets the limits of the search.
5. The model that gives the lowest value of J_{TOT} is retained and, from this point on, the parameter vector Θ is used for the ARX TDF module to behave as the discovered best ARX model with one TDF input port and one TDF output port (u and y in Figure 4.13). The ARX class was originally thought for supporting more than one input and/or output (see `_nb_in, _nb_out`) but the proof-of-concept ARX class only implements the SISO case.
6. The simulation starts and two modes can be chosen: the test mode (solid connection of Figure 4.13) or the operating mode (dashed connection).

Test mode: it is used to test if the ARX model that has been built behaves correctly. In order to prove its correctness, when the simulation starts, it will be stimulated with exactly the same input waveform used for the identification and the model output \hat{Y} is compared to the output values used for the identification Y . The “Golden Model” module provides the golden input and output by reading them from the same CSV¹ file used for the identification. The golden output and the model output are then subtracted and the error ε can be displayed.

Operating mode: it is used for allowing the ARX TDF module to operate in normal conditions, the stimuli received by the model are driven by the source (illustrated as “Custom Source” in Figure 4.13).

Listing 4.5: Constructor overloading.

```

1 ARX(sc_core::sc_module_name, std::string _filename, double alpha, int _nb_in, int _nb_out)
2 ARX(sc_core::sc_module_name, std::string _filename, double alpha, int _nb_in, int _nb_out, int
  _na_max, int _nb_max, int _nk_max)
3 ARX(sc_core::sc_module_name, std::string _filename, double alpha, int _nb_in, int _nb_out, int
  _na_min, int _na_max, int _nb_min, int _nb_max, int _nk_max)
4 ARX(sc_core::sc_module_name, int _nb_in, int _nb_out, std::vector<param_type>& P, int _na, int
  _nb, int _nk)
5 ARX(sc_core::sc_module_name, int _nb_in, int _nb_out, std::vector<param_type>& Pa, std::vector<
  param_type>& Pb, int _nk)

```

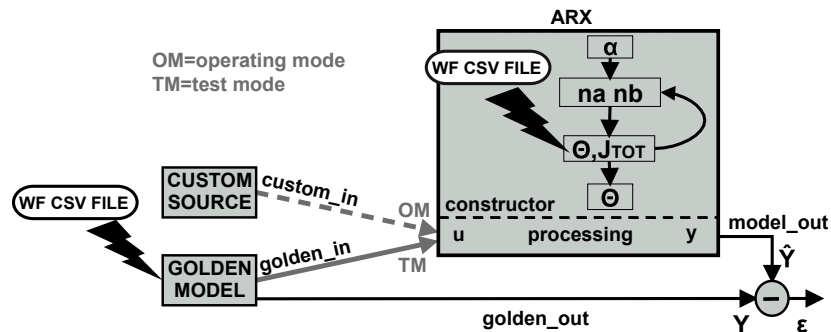


Figure 4.13.: SystemC AMS-based system identification test bench with both operating and test modes.

¹Comma-Separated-Values file format

The code is in Annex where Listing A.1 shows the code of the SytemC AMS module called *ARX* that, at its construction, performs the system identification by searching the parameter space for the best model and, during simulation, performs as the model of the identified device. Listing A.2 shows the SystemC test bench file wherein the ARX model operates.

A simple case study is used here as illustrative example. The system to be modeled and simulated is a hair dryer, the input u is the on/off control and the output y is the temperature of the blown air. The input/output data file is obtained from a Matlab benchmark for system identification. The output vector, contains 1000 measurements of temperature in the outlet airstream. The input vector contains 1000 data points, consisting of the voltage applied to the heater. The input was generated as a binary random sequence that switches from one level to the other with probability 0.2. The sampling interval is 0.08 seconds.

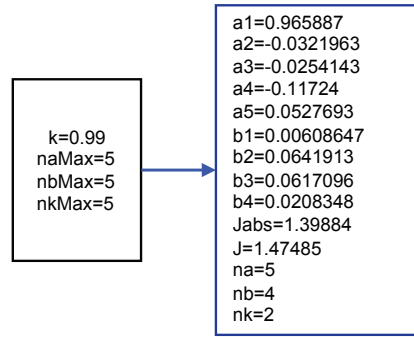


Figure 4.14.: Hair dryer ARX model constraints and resulting identified model.

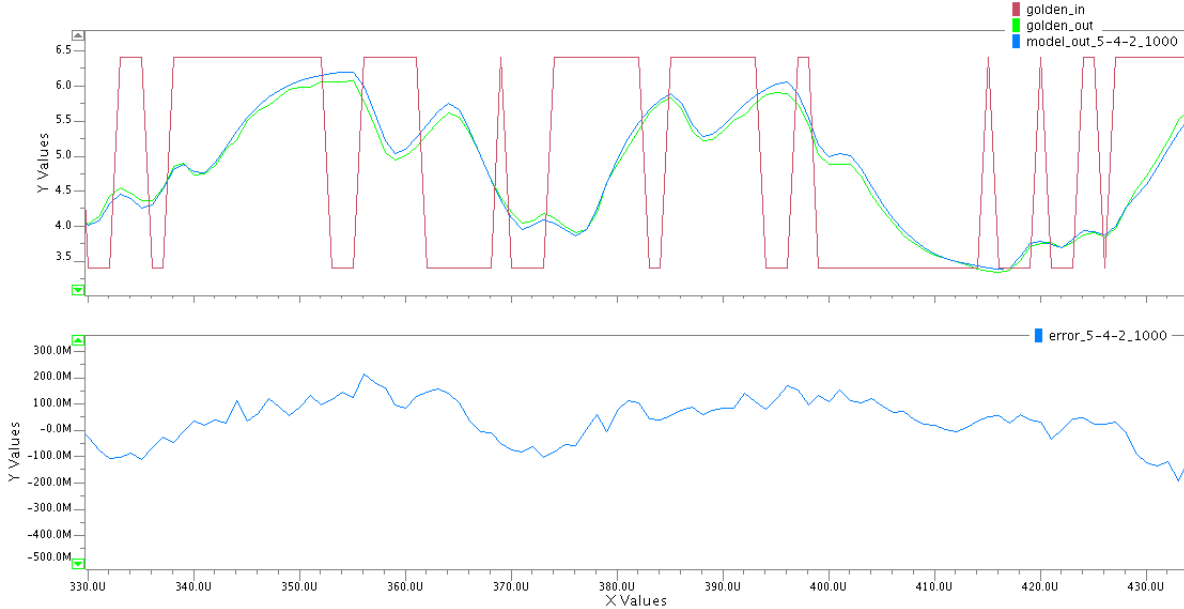


Figure 4.15.: Waveforms resulting from the hair dryer ARX model identification.

Figure 4.14 shows the input constraints given for ensuring a constricted complexity, the parameter α is 0.99 and the maximum values for na , nb and nk are set at 5. The system identification provides the best model with the specifications shown on the right of Figure 4.14. The input/output waveforms contain 1000 samples. Figure 4.15 shows the results of the

simulation configure in test mode. The red curve is the golden input, the green is the golden output, the blue curve on top is the output of the model ($na = 5, nb = 4, nk = 2$) that has been identified using the entirety of the 1000 temporal points. Finally, the blue curve on the bottom is the error ε between golden and model output.

Resuming, a class for the automated extraction/instantiation of ARX models of an AMS component from the input/output sampled waveforms has been developed, the minimization of the optimality criterion is done by taking into account the complexity of the model.

4.5. Application of system identification techniques to the closed-loop control for power consumption optimization

A new approach for controlling power consumption in RF devices is presented in this section. The approach is based on the definition of application-dependent performance modes for power hungry RF circuits and a logical control strategy that adjusts the power supply of each circuit to the mode required by the application. The control strategy uses embedded sensors, a recursive parameter identification approach and regression models for performance prediction, while demanding minimum embedded resources for computation.

The system identification techniques are here used with a slightly different aim with respect to the “behavioral modeling for overall simulation scope”. Instead of identifying the system behavior with a mathematical model in order to simulate its functioning in a wider complex system, the concept here is to perform online successive identifications for controlling the system performances. The control strategy is robust with respect to circuit parametric deviations due to the manufacturing process or aging mechanisms.

In the first subsections the methodology is described in a generic way for the control of a Circuit Under Control (CUC). The final subsections of this section will show how the methodology has been applied to an RF LNA case-study, a first system identification for defining the model structure is performed at design phase then the online identification is recursively performed thanks to envelop detectors as embedded sensors.

During the design phase two steps are required for applying the technique:

- construction of a behavioral input/output model of the CUC through system identification
- definition of a nonlinear model that links the set of parameters in the behavioral model to the performances of the CUC.

We will mainly discuss the first step, since the scope of this manuscript is to deal with behavioral modeling for different application purposes. Nevertheless, the second step will also be briefly depicted in order to show the simulation results allowing to validate the approach.

Figure 4.16 illustrates the approach for the construction of the two models. Monte Carlo simulation of N samples of the CUC (which includes the embedded sensors) is considered. For each sample i , the set of performances P_i are calculated by simulation. In addition, a transient simulation of the CUC is also carried with a persistently exciting input sequence $u(k)$ that covers the frequency range of the CUC. This is typically a Gaussian stimulus up converted to the CUC central frequency. The output sequence $y_i(k)$ is obtained via the embedded sensor,

typically an envelop detector. For the set of N CUC samples, we obtain the set of performances $\mathbf{P} = \{P_1, \dots, P_N\}$ and the set of output transient sequences $\mathbf{Y} = \{y_1(k), \dots, y_N(k)\}$. For a given model structure with m parameters, an identification algorithm uses the input sequence $u(k)$ and the resulting set of output transient sequences \mathbf{Y} to estimate the set of behavioral model parameters $\Theta = \{\Theta_1, \dots, \Theta_N\}$, where $\Theta_i = \{\theta_i^1, \dots, \theta_i^m\}$ corresponds to the set of m behavioral parameters for the i -th sample. Finally, from the set of performances \mathbf{P} and the set of behavioral model parameters Θ , nonlinear regression is used to compute a performance prediction model.

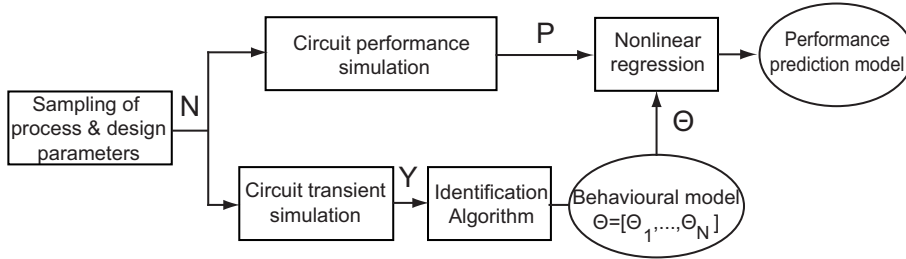


Figure 4.16.: Model building during the design phase.

In Figure 4.16, the structure of the behavioral model for the identification algorithm is known *a priori*. In practice, the structure of this model may be obtained after several iterations until the most accurate prediction models are obtained. Next section will describe how the behavioral input/output model structure has been built.

4.5.1. Behavioral input/output model

Behavioral modeling aims at finding a mathematical relationship between the input/output transient sequences of the CUC. For this work autoregressive models are used, so that the input/output relationship of the i^{th} sample is expressed as

$$y_i(k) = f(\gamma_i(k), \Theta_i), \quad (4.32)$$

$$\gamma_i(k) = [y_i(k-1), \dots, y_i(k-n_y), u(k-1), \dots, u(k-n_u)]. \quad (4.33)$$

where $\gamma_i(k)$ defines first order regressors that consider the memory of the model. It contains n_u previous values of $u(k)$ and n_y previous values of $y_i(k)$. Θ_i is the parameter vector of the model. In most practical cases, the behavioral model is nonlinear with respect to the parameters. In this work, *a priori* knowledge of the CUC (e.g. an LNA) allows us to restrict the study to dynamic models that are linear with respect to the parameters and function $f(\cdot)$ has a polynomial form.

To find the model structure, we apply the identification algorithm indicated in Figure 4.17. Some algorithm constants are fixed by the user according to his *a priori* knowledge of the CUC. These include the maximum memory allowed for the input sequence $u(k)$ and the output sequence $y(k)$, respectively, n_{u-max} and n_{y-max} . Also, since the model can be nonlinear with respect to the input, the maximum powers that can be applied for the input values $u(k-j)$ and output values $y(k-j)$ in the expression of the polynomial function $f(\cdot)$, respectively, p_u and p_y , are also given as constants. The model structure of the i^{th} CUC sample will now contain

higher order regressors given by Equation 4.34:

$$\gamma_i(k) = [u(k-1), \dots, u(k-n_{u-max}), \dots, u(k-1)^{p_u}, \dots, u(k-n_{u-max})^{p_u}, y_i(k-1), \dots, y_i(k-n_{y-max})^{p_y}]. \quad (4.34)$$

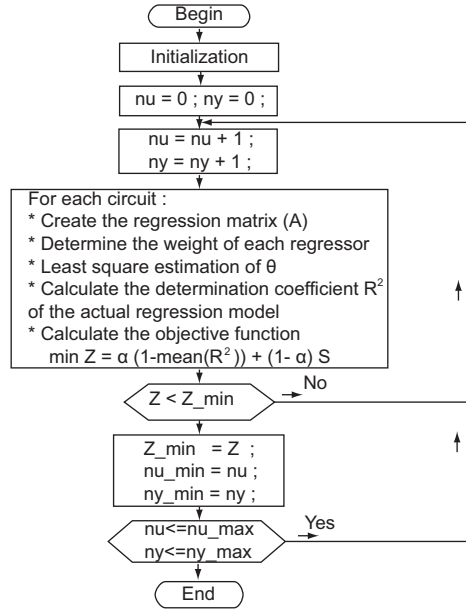


Figure 4.17.: Identification algorithm for deriving an input/output model structure.

The algorithm searches a model structure that contains regressors constructed from these variables. Each regressor has a weight w_j associated with it. These weights are used in the objective function to penalize or encourage the corresponding regressor. They are proportional to the memory and the exponential power of the variables in the regressor.

The identification algorithm is based on LMS estimation of the parameter vector Θ_i of the behavioral model, that is, the value of Θ_i that minimizes the regression error ε_i in

$$Y_i = A_i \Theta_i + \varepsilon_i, \quad (4.35)$$

where

$$Y_i = \begin{pmatrix} y_i(k-1) \\ \vdots \\ y_i(k-n_{max}) \end{pmatrix} \quad (4.36)$$

is the output sequence of length n_{max} of the i^{th} sample of the CUC, $n_{max} = \max\{n_{u-max}, n_{y-max}\}$ and

$$A_i = \begin{pmatrix} \gamma_i^T(k) \\ \vdots \\ \gamma_i^T(n_{max} + 1) \end{pmatrix} \quad (4.37)$$

is the regression matrix composed of vectors of the form of Equation 4.34, which are monomial terms of polynomial $f(\cdot)$ in Equation 4.33.

The LMS estimation of Θ_i is given by

$$\hat{\Theta}_i = (A_i^T A_i)^{-1} A_i^T Y_i. \quad (4.38)$$

The quality of the regression is quantified by the determination coefficient R_i^2 , given by

$$R_i^2 = 1 - \frac{\varepsilon_i^T \varepsilon_i}{\sum_k (y_i(k) - \bar{y}_i)^2}. \quad (4.39)$$

Finally, a multi-objective cost function is used in the identification algorithm to select the most suitable model structure. This function is given by

$$Z = \alpha(1 - \frac{1}{m} \sum_{i=1}^m R_i^2) + (1 - \alpha)S, \quad (4.40)$$

where α is a weighting factor for the two criteria of the objective function. S is a criterion used to penalize the complexity of the behavioral model structure as follows

$$S = \frac{\sum_{j \in \text{selected-model}} w_j}{\sum_{l \in \text{full-model}} w_l}, \quad (4.41)$$

where the numerator corresponds to the sum of the weighting factors of the regressors in the selected model, and the denominator to the sum of the weighting factors in a full model that contains all possible regressors.

4.5.2. LNA : Low Noise amplifier

Our case-study CUC is a Low Noise Amplifier (LNA) used in the 802.11g standard receivers that work in the 2.4 GHz ISM BAND. The LNA topology is presented in Figure 4.18(a). This inductive degenerated cascade structure is compatible with narrow band applications and offers WiFi performances. The biasing stage of the circuit is formed by resistors R1, R2 and transistor M3. With the use of gate and source inductances, a real part of the input impedance can be generated without the need of actual resistances. Thus, inductors Lg and Ls provide appropriate input matching at 50 Ω . Using this topology we can match the circuit without adding noise which implies a lower noise figure of the LNA. The gain stage is composed by M1 and M2. M1 provides the high gain, whereas M2 isolates the input from the output, reducing the Miller capacitor and eliminating the dependency between the gate-drain capacitance and the drain inductance. Increasing the reverse isolation is important for: (1) lowering the effect of the Local oscillator leakage produced by the following mixer, and (2) minimizing the feedback from the output to the input. At the output of the circuit, the parallel Ld-Cd tank resonates at 2.4GHz and the resistor Rd controls the gain at this frequency. The LNA is designed using the 0.25 μm BiCMOS7RF technology provided by STMicroelectronics. The principle performances of the LNA at 2.4 GHz are: Gain > 12 dB, NF < 1.6 dB and IIP3 > 5.9 dBm.

4.5.3. Envelope Detector

Different sensors that extract RF power and convert it into a low-frequency signal for BIST purposes are available in the literature [Abdallah 10, Valdes-Garcia 08, Hsieh 06]. The envelop detector that has been used is detailed in [Abdallah 10]. It has been conceived with a very simple architecture, based on the following design constraints: minimum silicon area overhead, high input impedance in the frequency range of interest to avoid undesired loading of the CUC, high dynamic range suitable for testing different on-chip CUCs, and wide band of operation to monitor CUCs that work at different frequencies involved in the system. This circuit consists of two stages as shown in Figure 4.18(b).

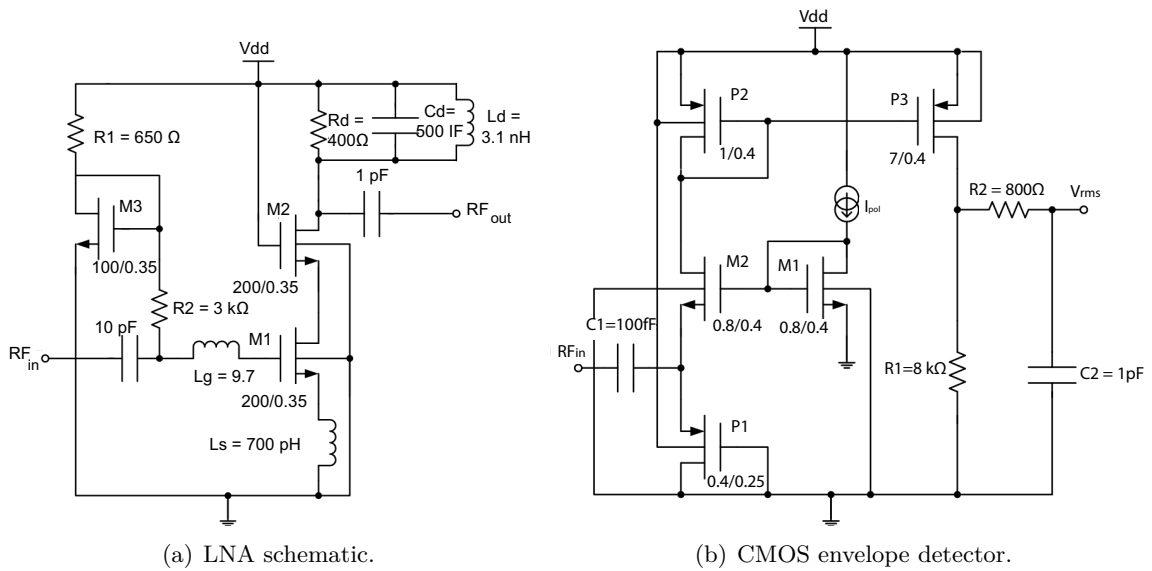


Figure 4.18.: Transistor level circuits.

The first stage is a rectifier that performs half-wave rectification on the current delivered at the source of the transistor M2. The half-wave rectifier works as follows. The operating point of the transistor M2 is controlled by the bias current I_{pol} which flows through the diode-connected transistor M1. The difference between the fixed gate voltage of M2 and its source voltage is very close to its threshold voltage, such that M2 is at the verge of conduction. When the current passing through the source of M2 is positive, transistor M2 is off and the current passes entirely through transistor P1 to the ground. During the negative half-cycle, the source voltage of M2 decreases which activates M2. During this half-cycle, the current flowing through M2 is copied and amplified through the current mirror formed by transistors P2 and P3. It is important to note that the sensitivity of the detector is mainly controlled by I_{pol} . In particular, as this current is reduced, the rectifier is sensitive to smaller signal amplitudes and this characteristic is critical in an on-line monitoring scenario when the envelope detector is related to the input of the LNA. On the other hand, a main challenge exists between the high dynamic range of the envelope detector and its sensitivity. In the second stage, the amplified current is converted to voltage through R_{out} which is equivalent to the output resistor r_{ds} of the transistor P3 in parallel with the resistor R1. Finally, in the low pass filter R2-C2 a compromise exists between the time constant of the settling response and the ripple in the output voltage. The envelope detector has the following characteristics: an input impedance equal 1.5-11 kΩ in the band

of operation 500 MHz -10 GHz, an input dynamic range of 35 dB and an area equal to $2170 \mu m^2$, which corresponds to 0.543 % of the area of the LNA. Figure 4.19 plots the input-output characteristic of the envelope detector for the two edges of the frequency band. Furthermore, the study of the impact of the envelope detector on the LNA specifications is achieved by simulating the LNA performances with and without the envelope detector. The analysis shows just a low degradation thanks to the high impedance at the input of the envelope detector.

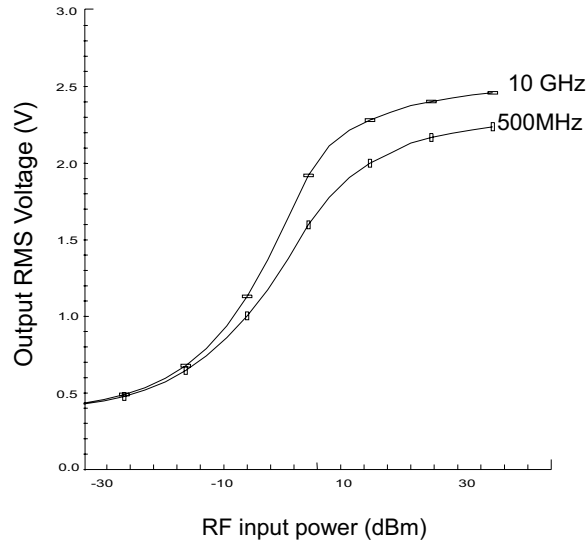


Figure 4.19.: Input-output characteristic of the envelope detector.

4.5.4. Nonlinear performance prediction model

Nonlinear regression is performed to obtain a relationship between each performance in the set \mathbf{P} of CUC performances, and the set Θ of estimated parameter vectors as shown in Figure 4.16. Simple functions are required in order to minimize the computation resources required on-chip. For this, we use a kind of Branch and Bound algorithm to explore a predefined space of regressors. These regressors use as variables the coefficients Θ of the model structure. The predefined space is limited to second order polynomial regressors. The steps of this algorithm for obtaining the regression functions for each performance are as follows:

- Circuit data (\mathbf{P}, Θ) are randomly separated into training and validation sets.
- The complete space of regressors is considered to form a reference regression matrix. Each column of this matrix corresponds to a regressor that is weighted according to its complexity given by the sum of the power of the involved variables. An initial value of predicted performances is obtained using the best correlated column of this matrix with the performance we want to predict.
- The algorithm keeps track of two subsets of columns: a subset of columns currently accepted in the model and a subset of columns in the reference matrix that have not yet been considered.

- In each iteration of the algorithm, the column of the reference matrix that is best correlated with the regression error obtained in the previous iteration is added to the model and LMS parameter estimation is performed.
- An objective cost function is calculated from the determination coefficient R^2 and the complexity of the model as in Equation 4.40.
- If a considerable improvement of the objective function is found, a new iteration is considered with the current model, otherwise we return to the previous model and another column is tried.
- The algorithm stops once there are no further columns to try (the space of regressors has been explored).

In the LNA case-study the performances tone controlled are here listed but further details can be found in [Khereddine 12]:

- Input matching and input reflection parameter (S11): it reflects the matching at the input of the LNA.
- Gain (S21): the LNA represents the first gain stage in a receiver system.
- Noise Figure (NF).
- Non linearity effects (1dB compression, IIP3).
- Isolation parameters (S12, S22).

4.5.5. Adaptive logical control

The control of the CUC can be done either concurrently with the system normal operation, or during idle times using the same test sequence considered in the design phase (in this last case, the Gaussian-like persistently exciting input sequence will be generated by the controller, which can allow the extraction of a more precise behavioral model). As shown in Figure 4.20, the input/output sequences obtained via the embedded sensors are used by the controller to estimate the parameters of the CUC behavioral model from which the performances are predicted.

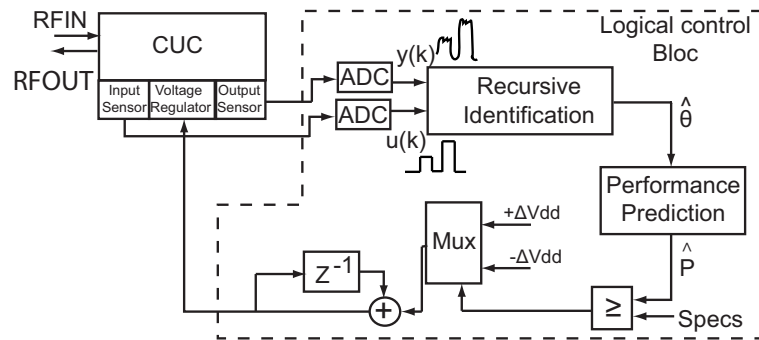


Figure 4.20.: Adaptive logical control strategy.

Recursive parameter identification

For online and offline estimation of the parameter vector Θ of the behavioral model, we use a recursive LMS algorithm that process data on the fly, thus saving memory resources, and which requires only additions and multiplications by a constant. The algorithm is initialized as

$$\Theta^{(0)} = 0, Q^{(0)} = \rho I, \quad (4.42)$$

where $Q^{(0)}$ is an initial variance-covariance matrix formed by multiplying the identity matrix I by a positive constant ρ (as discussed in section 4.5.7). The classical recursive LMS algorithm is as follows

$$\begin{aligned} K^{(k)} &= \frac{\lambda^{-1} Q^{(k-1)} \gamma'(k)}{1 + \lambda^{-1} \gamma'^T(k) Q^{(k-1)} \gamma'(k)}, 0 \ll \lambda \leq 1 \\ \varepsilon^{(k)} &= y^{(k)} - \gamma'^T(k) \hat{\Theta}^{(k-1)} \\ \hat{\Theta}^{(k)} &= \hat{\Theta}^{(k-1)} + K^{(k)} \varepsilon^{(k)} \\ Q^{(k)} &= \lambda^{-1} Q^{(k-1)} - \lambda^{-1} K^{(k)} \gamma'^T(k) Q^{(k-1)}, \end{aligned} \quad (4.43)$$

where $\gamma'^T(k)$ is a subset of the structure given by Equation 4.34, according to the selected model structure. The recursive parameter estimation stops once convergence is reached ($\varepsilon^{(k)}$ is smaller than a given constant) or a pre-defined number of iterations is attained.

Logical control strategy

The logical control is not intended to be permanently on. It is activated when the application sets a new performance mode for the CUC which requires a different CUC power supply. The control follows an iterative algorithm that starts with the CUC power supply set at the maximum value. During each iteration, the behavioral parameters $\hat{\Theta}$ are estimated by the LMS recursive algorithm and used by the regression equations to predict the CUC performances \hat{P} . These are in turn compared with the specifications required by the new performance mode. If the specifications are met, the power supply of the CUC is reduced by a pre-defined value ΔVdd and a new iteration is considered. Otherwise, if the specifications are not met, the power supply is incremented by a value ΔVdd and the control stops.

4.5.6. LNA performance modes

An analytical study on the variation of the LNA performances with respect to changes to the power supply voltage has been done. For each one of the six afore-defined performances of the LNA the necessary condition of monotony in the range of interest has been analytically proved [Khereddine 12]. These theoretical analysis have then been verified by simulation for the case-study LNA. Figure 4.21 shows transient-level simulations of the performance variation when the power supply is varied, for all performances except S22 (which does not vary significantly). These variations are in all cases monotonic.

For a typical application, gain and noise figure are the most important LNA performances to be controlled. S11 and S12 typically have maximum values that cannot be exceeded (e.g.

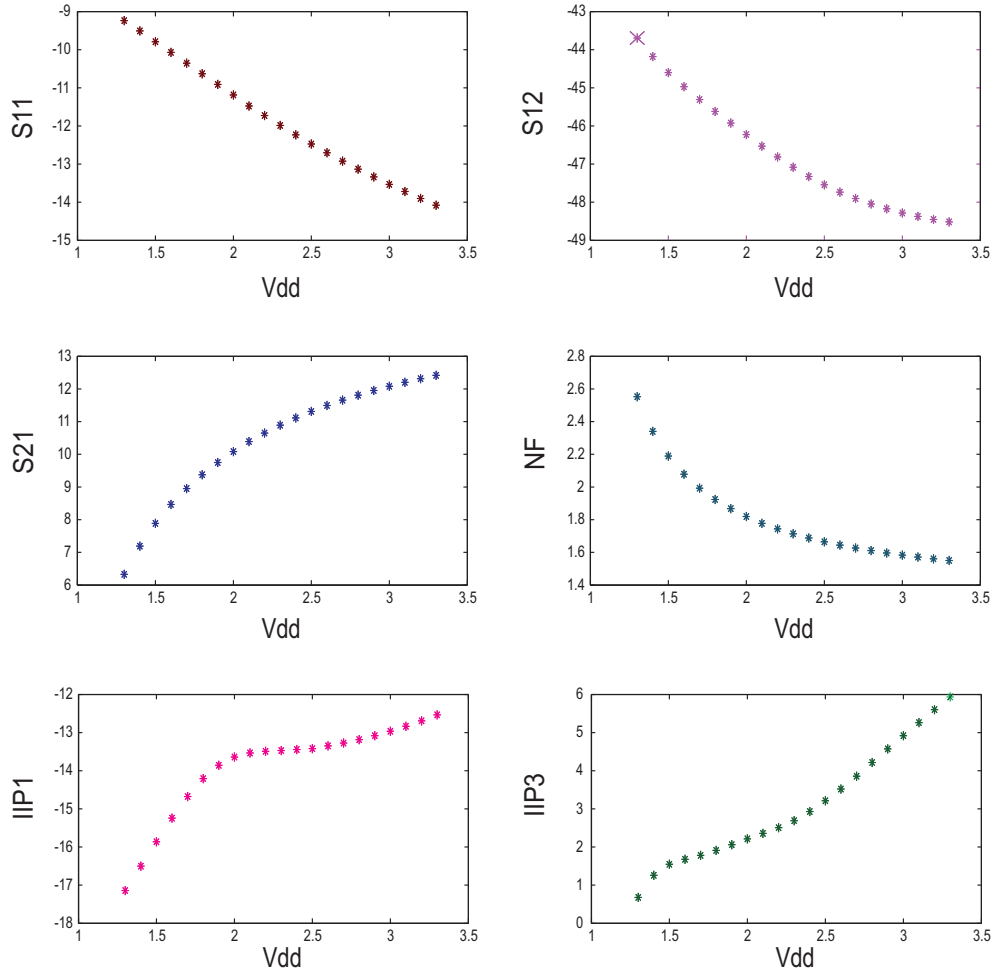


Figure 4.21.: LNA performances versus power supply.

$S_{11} < -10dB$ and $S_{12} < -40dB$). Similarly, $IIP1$ and $IIP3$ have minimum values that cannot be exceeded (e.g. $IIP1 > -20dBm$, $IIP3 > -10dBm$).

As an example, we define three different performances modes: a MAX mode of maximum power supply, a MIN mode of minimum power supply and an INT mode of intermediate power supply. In all modes, the above conditions for S_{11} , S_{12} , $IIP1$ and $IIP3$ must be respected. In MAX mode, $Gain > 11.5dB$ and $NF < 1.65dB$. In INT mode, $Gain > 10dB$ and $NF < 2dB$. Finally, in MIN mode, $Gain > 8dB$ and $NF < 2.2dB$.

The level of power supply required by each performance mode will be set by the controller. For later reference, Figure 4.22 shows the power consumption of the CUC as a function of the power supply voltage obtained by transistor-level simulation.

4.5.7. Simulation results

For building the behavioral and predictive models for the CUC of Figure 4.18(a), the CUC input stimulus is obtained by mixing a Gaussian signal with an average amplitude of 150 mV,

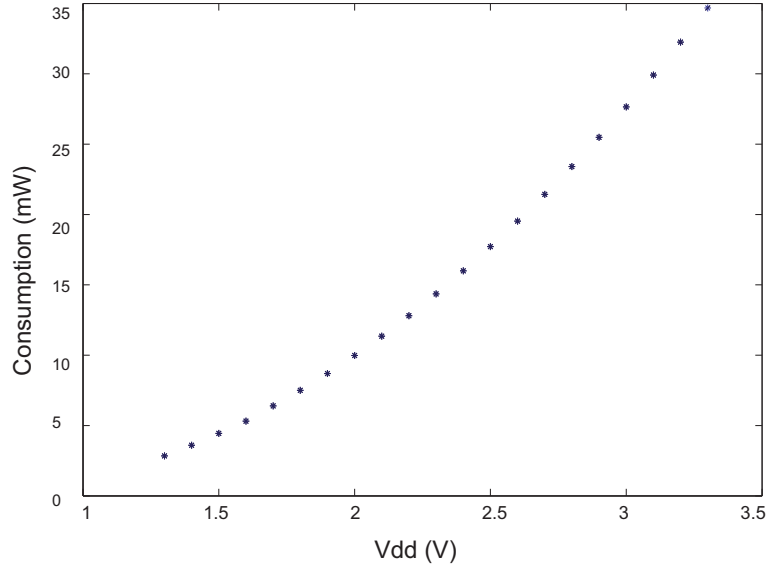


Figure 4.22.: LNA consumption versus power supply.

sampled at 10 MHz, with a carrier signal at 2.4 GHz. As shown in Figure 4.23, 200 values of $u(k)$ and $y(k)$ are obtained for a simulation time of 10 us (the mixer and the ADC/DAC converters are considered ideal).

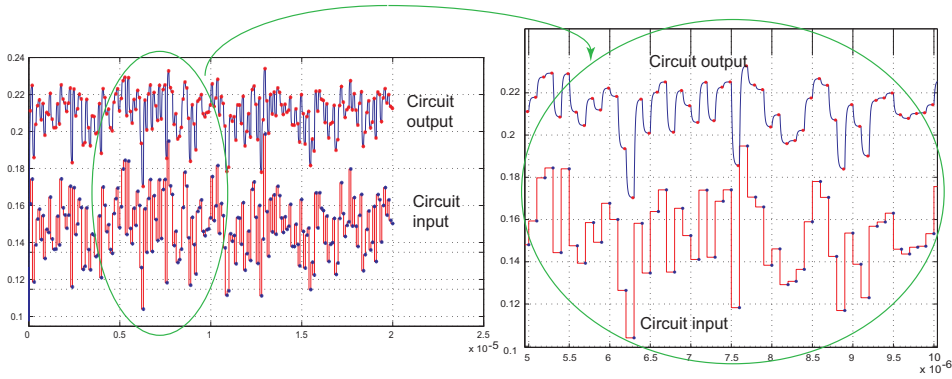


Figure 4.23.: Sampled input and output signal.

A Monte Carlo transistor-level simulation of the CUC, with $N = 1000$, has been performed for three different levels of power supply: 3.3 V (maximum power supply voltage), 2.3 V and 1.3 V. For each level of power supply, the 200 values of the input/output sequences of each circuit sample are used by the algorithm of Figure 4.17 to identify the model structure. In the three cases, the following model structure has been retained:

$$y(k) = \theta^0 + \theta^1 u(k) + \theta^2 u(k)^2 + \theta^3 u(k)^3. \quad (4.44)$$

This behavioral model gives a determination coefficient R^2 higher than 98% for all 3000 circuit samples. For example, the model identified for the 5-th circuit sample is given by

$$y_5(k) = -0.03 + 0.78 u(k) - 3.37 u(k)^2 + 6.76 u(k)^3. \quad (4.45)$$

Figure 4.24 compares the value predicted by the model and obtained by simulation of the circuit for 100 different time points. The predicted and the actual values are very close.

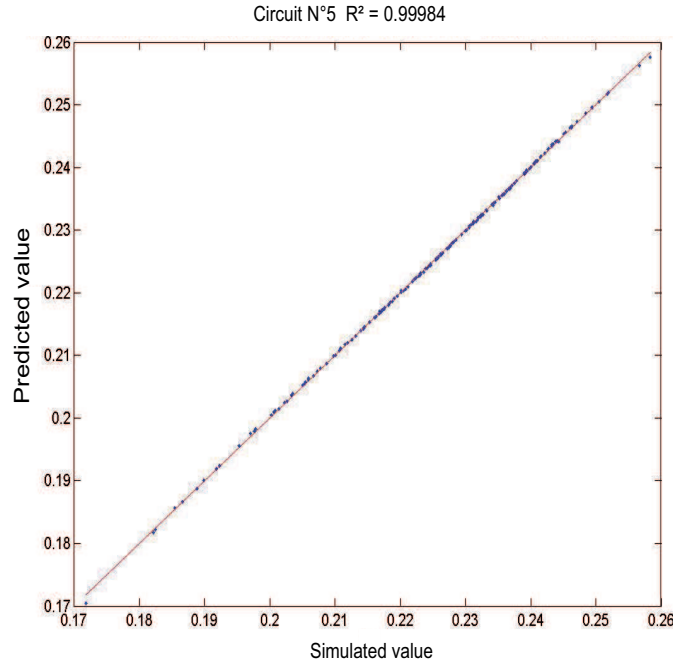


Figure 4.24.: Predicted and simulated output values of the CUC.

A prediction model is built using Branch and Bound algorithm for each performance. The prediction equations use only the parameters $\theta^0, \dots, \theta^3$ of the behavioral model to predict the values of all CUC performances, as shown in Figure 4.25.

The simulation of the adaptive logical control strategy has been performed by considering the CUC at transistor-level and the controller modeled in Verilog-A, the tasks of the latter are to recursively identify the model parameter set and to run the regression equations to predict the CUC performances. The CUC is stimulated by the same stimulus described above. Initially, the CUC is set at the MAX mode, where the maximum power supply of 3.3 V is required. Next, we simulate the transition to a MIN mode, for which the performances are specified as indicated in section 4.5.6.

Figure 4.26 shows the different iterations of the algorithm. Each iteration lasts 30 us, with the convergence time for the recursive parameter identification algorithm being somewhat smaller. This Figure also illustrates the convergence of the behavioral parameter θ^0 , for different iterations. The choice of values for the variables in the recursive LMS algorithm is important, in particular θ^0 , ρ and λ in Equations (4.42) and (4.43). The value of ρ must be as large as possible. The closer the value of λ to 1 the slower the convergence of the algorithm, but the

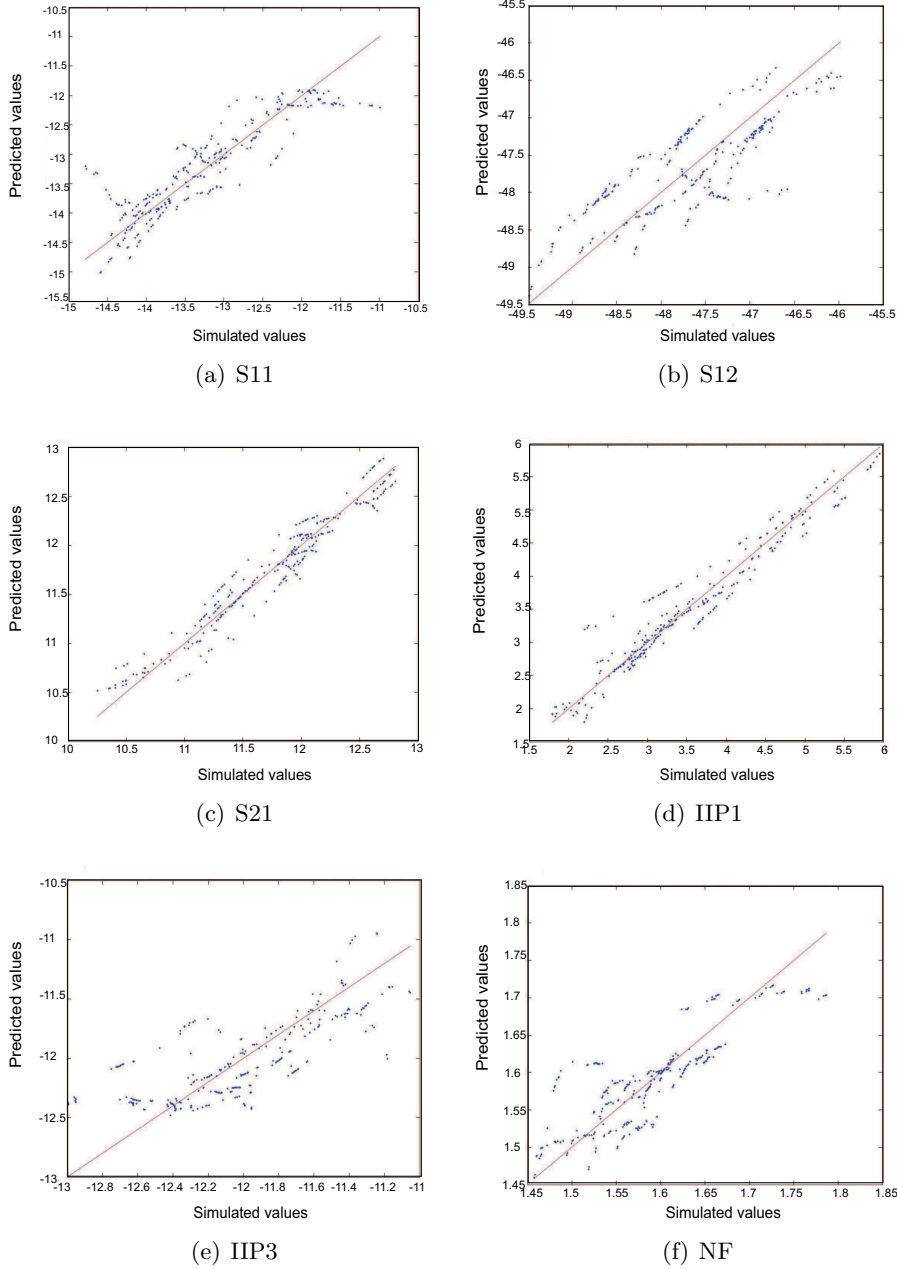


Figure 4.25.: LNA performances prediction.

variance of $\hat{\theta}$ after convergence is smaller, oscillating around the optimum value. If λ is closer to 0, the opposite behavior is observed.

The circuit performances are estimated by the end of each iteration, once convergence has been achieved. Figure 4.26 shows that the control algorithm needs six iterations to reach a power supply voltage level of 2.3 V for which one of the specifications of the MIN mode is no longer respected. In the next iteration, the control stops with a power supply voltage level of 2.5 V, since the controller uses power supply voltage steps of 200 mV (this large value is used here

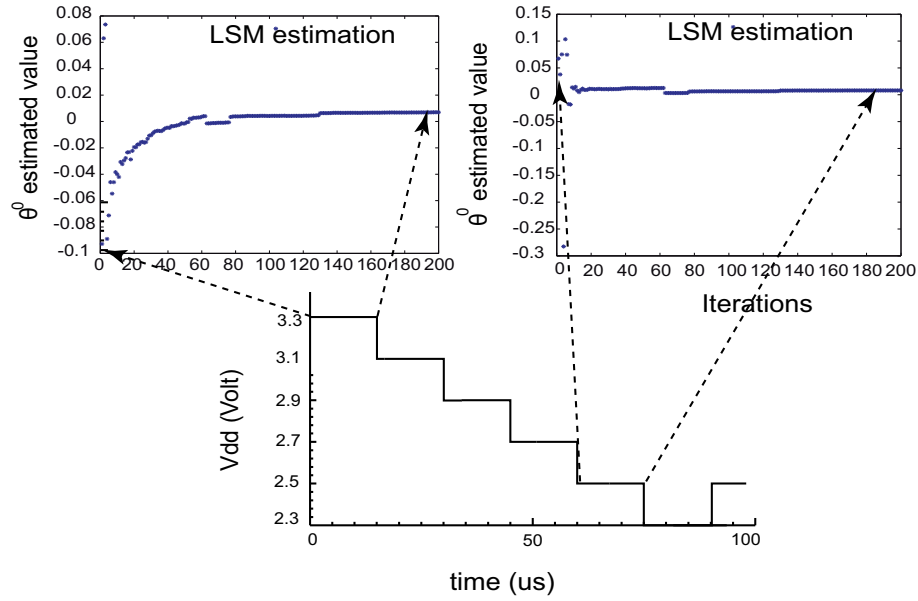


Figure 4.26.: Transistor-level simulation of the control strategy.

Table 4.1.: Performance prediction during logical control.

Vdd (V)	Saved power(%)	Predicted/simulated values	NF	S11	S12	Gain	IIP1	IIP3
			≤ 2.32	≤ -10	≤ -40	≥ 6.2	≥ -20	≥ -3
3.1V	-14	Prediction	1.70	-12.46	-43.65	10.41	-16.29	1.92
		Simulated value	1.57	-13.72	-48.37	12.20	-11.61	5.26
2.9V	-26	Prediction	1.74	-12.30	-43.23	10.06	-17.09	1.06
		Simulated value	1.59	-13.33	-48.17	11.95	-11.83	4.57
2.7V	-38	Prediction	1.99	-12.23	-42.99	8.97	-18.58	0.66
		Simulated value	1.62	-12.91	-47.90	11.65	-12.03	3.85
2.5V	-49	Prediction	2.25	-11.96	-42.88	7.86	-19.70	0.39
		Simulated value	1.66	-12.47	-47.54	11.31	-12.19	3.21
2.3V	-54	Prediction	2.39	-11.35	-42.90	7.16	-19.73	0.22
		Simulated value	1.71	-11.98	-47.08	10.89	-12.26	2.6902

to reduce simulation time). Table 4.1 illustrates the evolution of the performances predicted, and the reduction of power consumption at each level of power supply voltage (according to Figure 4.22). For example, the MIN mode with the power supply controlled at 2.5 V has a power consumption reduction of 54%.

4.5.8. Conclusions

A new approach for reducing power consumption in RF devices based on adapting the power supply voltage by means of a logical control strategy has been described. This strategy relies on embedded sensors, **real-time parameter identification** and performance prediction, and

makes use of simple on-chip resources. Significant power savings have been demonstrated at the transistor-level for an RF LNA case-study with different performance modes. The control algorithm can guarantee the required specifications for each performance mode, despite circuit parametric deviations due to the manufacturing process or aging mechanisms.

The Verilog-A/Cadence-Spectre simulations allowed to validate the power saving methodology through a performance estimation from a low-level description. From a more abstracted viewpoint, if the hardware description of the system has to be simulated, a promising solution would be to describe the analog CUC by means of the available SystemC AMS MoCs while the control/recursive identification algorithm would be loaded on a SystemC-based model of an Instruction Set Simulator (ISS). Such a solution would allow to accurately describe the algorithm execution time according to the amount of hardware resources such as memory size or processor clock frequency. This would enable a system-level architecture exploration/validation of the digital control/identification part.

Further details can be found in our works [Khereddine 10, Khereddine 12]. Future perspectives aim at validating this approach in hardware.

4.6. SAW based chemical sensor case study

In this section as a proof of the proposed behavioral modeling-based design methodology it will be shown how a behavioral model of a chemical sensor based on Surface Acoustic Waves (SAWs) has been obtained and used for the electrical characterization of the microelectronic frontend [Cenni 08, Cenni 09a, Cenni 10].

A SAW device is used in this work as the sensing element for a chemical sensor that will be embedded in a wireless sensor node. The SAW device is intended to detect the concentration of gaseous mercury in the environment. The microelectronics frontend architecture has been designed at transistor level in a $0.35\mu\text{m}$ CMOS technology. The SAW device is embedded in a phase-locked loop (PLL) that converts the change of concentration of gaseous mercury into a shift of the loop frequency.

The section will first give an overview of the SAW device and its functioning, then the Matlab based model reproducing the structural analysis for the SAW sensor will be shown. In order to perform simulations of the overall sensor including the digital control and read-out blocks it has been calculated a reduced-order fitting Laplace transfer function in the region-of-interest. The Laplace transfer function has been implemented both in an AHDL (in our case Verilog-A) for coupled simulations with the transistor level (such as Cadence-Spectre) and in a higher level modeling language (such as SystemC AMS) for coupled simulations within the system-level overall surrounding environment (e.g. SystemC / TLM platform).

In section 4.6.4 it will be shown how the read-out interface has been designed at transistor level and how a model of the SAW device described in the Hardware Description Language Verilog-A has been used to carry out post-layout simulations of the overall sensor including the digital control and read-out blocks. Analog tests on the fabricated circuit have also been performed demonstrating the functionality of this microelectronics frontend interface.

In section 4.6.5 it will be shown how the transistor level implementation functionality has been captured and used to model the overall system using SystemC / SystemC AMS.

4.6.1. Overview of the SAW device

The structure of the SAW device as shown in Figure 4.27 was described in our work [Cenni 09a]. The SAW device receives an input electrical signal which is transformed into a surface acoustic wave by means of an input InterDigitated Transducer (IDT). The acoustic wave propagates on a piezoelectric substrate that is covered by a chemically sensitive Au layer between the two IDTs. The velocity of the SAW depends on the amount of target gas adsorbed in the sensitive layer, that is, the SAW propagates and undergoes an amount of delay proportional to the concentration of the adsorbed target gas. The acoustic wave is reconverted into an electrical signal at the output IDT. From a temporal point of view, the SAW device behaves as a delay line controlled by the concentration of gas adsorbed in the sensitive layer. From a frequential point of view, the SAW device has a pass-band behavior with the form of the sinc function.

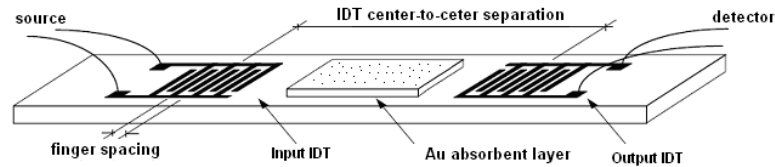


Figure 4.27.: Structure of the SAW device.

4.6.2. Behavioral modeling of SAW sensors

Two behavioral models of the SAW sensor have been studied but only one of them is presented. They are both based on the Mason's model that gives an analytical description of the propagation of acoustic waves in piezoelectric materials. A comparison of both the models will be done before presenting a behavioral model suitable for simulation together with the microelectronics interface.

The first model, the Delta-Mason's model [Urbanczyc 02], is made of the widely known Mason's model (describing piezoelectric transducers with volume elastic waves) and the δ -function model that consists of a mathematical description of the IDT behavior. This model requires a quite complex lumped components network for the modeling of each interdigitated finger. The equivalent circuit of the overall SAW sensor has been built by replicating a high number of interdigitated fingers for each IDT. It has been tested and considered not suitable for the simulation of the overall frontend because of the too high computational effort demanded.

The second model is the SAW crossed-field model. It is derived from the Mason's equivalent circuit as in the previous case. In this model, the distribution of the electric field under the electrodes of an excited IDT is approximated as being normal to the piezoelectric surface, as sketched in Figure 4.28.

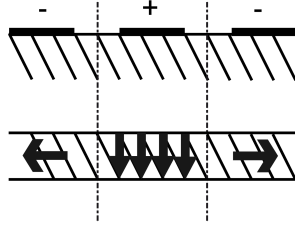


Figure 4.28.: Instantaneous E-field direction in the SAW crossed-field model.

In the building of the crossed-field model, each finger pair is represented by the original Mason's equivalent 3-port network block shown in Figure 4.29 [Grobelyny 06]. In the figure the terms F_1, v_1 and F_2, v_2 describe the acoustic ports and U, I describes the electrical port.

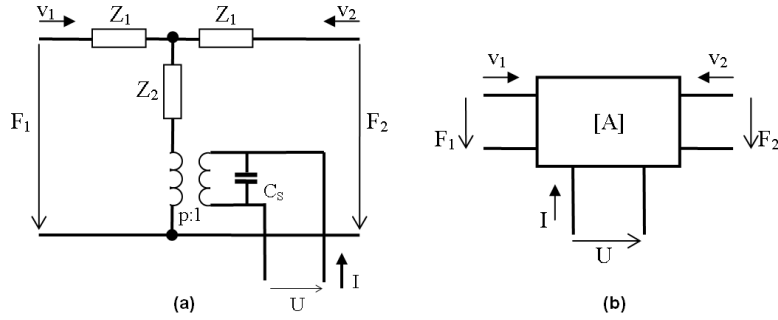


Figure 4.29.: Equivalent circuit for an IDT electrode and its associated gap between metalizations, based on the SAW crossed-field Mason's model (a) and the corresponding three-port building block (b).

The IDT model is constructed by linking the acoustic ports in cascade and the electrical ports in parallel, thus resulting in a 3x3 matrix for each IDT. The overall model of the SAW device (Figure 4.30) is then obtained by instantiating two IDT models (A_I, A_O), both terminated with corresponding acoustic impedances, and a T-network (A_S) which models the propagation path of the SAW in the sensor chemical layer.

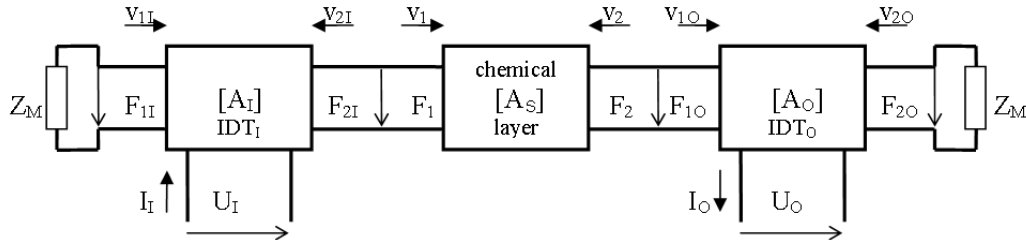


Figure 4.30.: Equivalent network of both IDTs and the delay path of the SAW chemical sensor.

However, this equivalent circuit of the SAW sensor cannot be simulated using an electrical simulator. This is because the impedances of Figure 4.29(a) are frequency-dependent in a way that cannot be represented with typical components such as inductors and capacitors. In addition, this model leads also to an excessively large number of components. Thus, a mathematical representation of this equivalent circuit has been described using Matlab, where the different cascaded blocks have been combined using matrix computations. The resulting

mathematical model can be easily simulated, but only in the frequency domain [Grobelyny 06]. The obtained model is called structural model since the frequency response is obtained by analyzing the attenuation provided by the model at linearly distributed frequency values in a large enough range of frequencies (see blue branch in Figure 4.1).

4.6.3. Laplace transfer function model of the SAW sensor

The crossed-field Mason's model described in Matlab provides the frequency response of the SAW sensor. In order to obtain a model that can be used with an electrical simulator, we have used the fact that the SAW device has a pass band behavior. Thus, we have performed a rational fitting of the computed frequency response in order to obtain a rational function in the Laplace domain which approximates the crossed-field Mason's model in the required frequency range. This fitting function is calculated for a given SAW velocity in the piezoelectric material. Once the Laplace function is obtained, the Verilog-A library primitives are used to describe it. The resulting Verilog-A model can then be simulated in both the frequency and the time domains. Figure 4.31 shows the frequency response of the SAW sensor in magnitude and phase. The continuous lines refer to the Matlab model and the dashed lines to the rational fitting function.

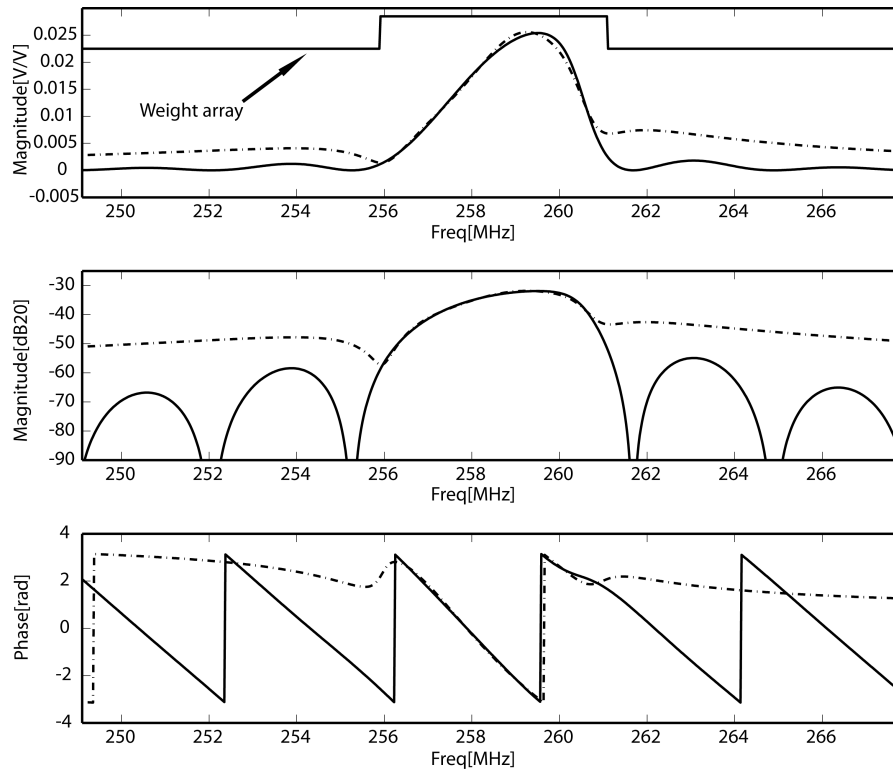


Figure 4.31.: Frequency response of the SAW sensor using the Matlab models and the rational fitting function.

The fitting function $H(s)$ is obtained in the form of Equation 4.46 where n is the number of poles ($n=10$ in our case). The band of highest accuracy for the fitting is delimited by the piece-wise linear curve on the top of Figure 4.31.

$$H(s) = \sum_{k=0}^{n-1} \frac{C_k}{s - A_k} \quad (4.46)$$

The obtained Laplace fitting function model does not provide an access for emulating the effect of the gas concentration. In fact, a detailed analysis of the crossed-field Mason's model in Matlab (Figure 4.32) has shown that varying the SAW velocity does not affect the amplitude of the transfer function but only results in a phase shift, if considered in a narrow frequency range. Thus, the effect of a change in gas concentration, resulting in the SAW velocity variation, has been modeled with an additional delay model that is placed in cascade with the fitted SAW model. Figure 4.32 shows the phase of the frequency response of the crossed-field model for different values of SAW velocity.

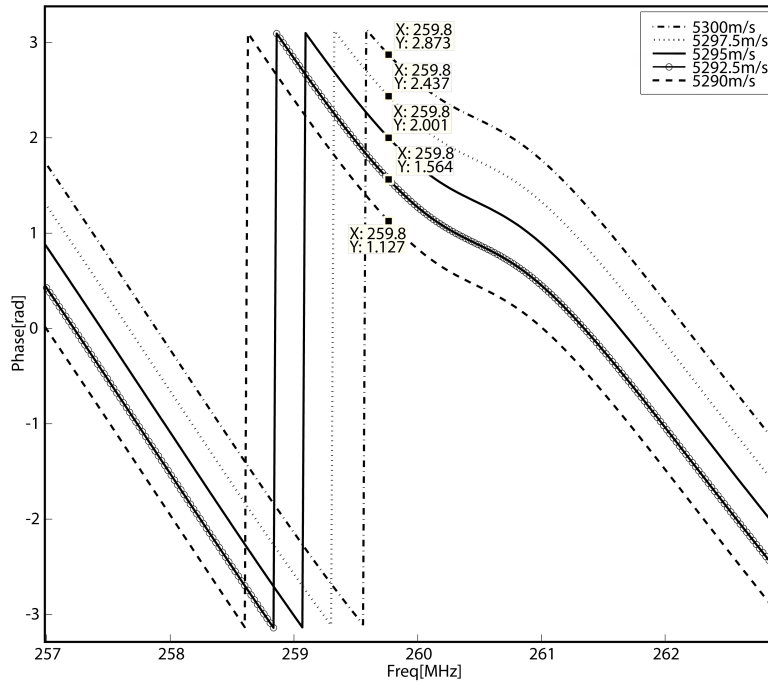


Figure 4.32.: Effect of SAW velocity changes in the crossed-field model.

Our target specification is to obtain a gas concentration measurement up to 10 *ppm* with linearity better than 1%. We assume that a gas concentration change of 10 *ppm* results in a SAW velocity change of 10 m/s. This is however dependent on the experimental conditions. Unfortunately, this data is not available at this early stage since this requires chemical tests performed with a fabricated sensor interface. For a SAW velocity change of 10 m/s, the corresponding delay value D is readily obtained through Equation 4.47, where Φ_{v1} and Φ_{v2} are, respectively, the phases in radians corresponding to the SAW velocities of $v1 = 5300$ m/s and $v2 = 5290$ m/s according to Figure 4.32, and ω_0 is the nominal pulsation $2\pi 259.8$ MHz. The SAW sensor has then a conversion factor of about 107 *ps/ppm*. Thus, for example, if a measurement resolution of ± 1 *ppm* is required, the microelectronics interface must have a time resolution of about ± 100 ps.

$$D = \frac{\Phi_{v1}(\omega_0) - \Phi_{v2}(\omega_0)}{\omega_0} = \frac{2.873 - 1.127}{2\pi \cdot 259.8 \cdot 10^6} = 1.07ns \quad (4.47)$$

4.6.4. Low-level Verilog-A / Cadence-Spectre based modeling

The Laplace transfer function model of section 4.6.3 has then been implemented using Verilog-A. Two blocks make the SAW device model: the pass-band rational fitting function and the controlled delay block.

Microelectronics interface architecture

In order to measure the input-output delay provided by the SAW sensor, different interface architectures can be considered [Sternhagen 02]. These include oscillators, phase detectors and phase-locked loops (PLL).

SAW oscillators are based on a loop configuration composed by a single amplifier and the SAW sensor. A variation in the SAW velocity results in a shift of the oscillation frequency in order to maintain the total loop phase shift as a multiple of 360° . However, there may be more than one frequency within the passband that can permit a loop phase shift of 360° , giving rise to an undesirable phenomenon called mode hopping.

Phase detectors compare the phase between the input and the output of the SAW sensor. These architectures are composed by a local oscillator, whose output is split into two signals. One signal is directly sent to one input of the phase detector. The other signal is sent to the SAW sensor input and the output of the SAW sensor is sent to the other input of the phase detector. Thus, the phase detector measures the phase difference that corresponds to the SAW sensor input-output delay. This architecture is more robust than the oscillator configuration but lacks of high resolution because of its voltage output.

The PLL architecture places the SAW sensor in a loop that includes a Voltage Controlled Oscillator (VCO), a Phase Detector (PD) and a frequency counter. As shown in Figure 4.33 the PD output is used in a closed loop to maintain a constant phase shift across the SAW sensor by controlling the voltage value V_{TUNE} , thus the loop frequency. Changes of the loop frequency are proportional to changes of the SAW velocity. The PLL-based architecture offers both robustness and high resolution thanks to the frequency output.

The architecture of our chemical sensor is shown in Figure 4.34. It consists of two SAW sensors with their respective PLL-based circuits and a digital control unit. Since SAW devices are sensitive to temperature changes, two identical measurement units are used in order to cancel temperature effects. Both sensors are fabricated on the same substrate. While one is coated with a sensitive layer (indicated as “active” in Figure 4.34), the other (“reference”) is not. Frequency measurements are performed by the digital control unit which is composed by the read-out circuitry and a Finite State Machine (FSM) that schedules the sensor phases.

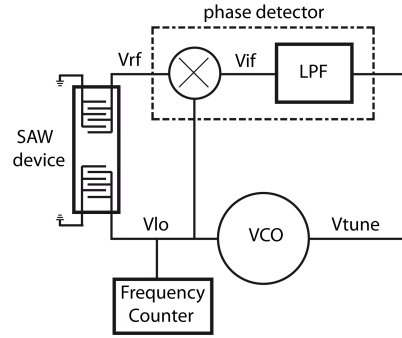


Figure 4.33.: SAW PLL-based measurement architecture.

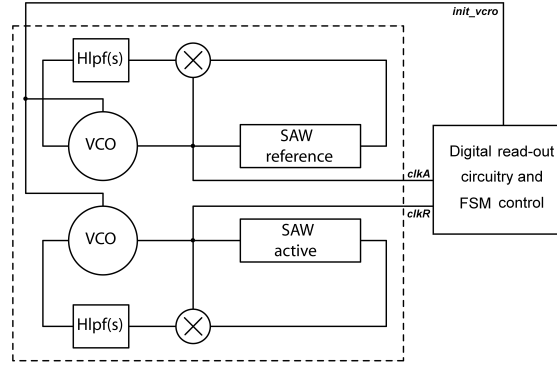


Figure 4.34.: Block diagram of the overall measurement system.

Design of the microelectronics interface

The PLL-based measurement architecture of Figure 4.33 has been implemented using fully differential signals. The differential signal (D) is obtained as the difference of the + and - signals while the common-mode signal (CM) is their average value. A block diagram closer to the transistor level implementation of a single PLL is shown in Figure 4.35.

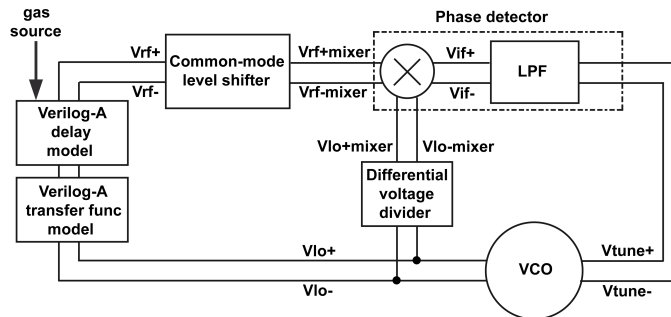


Figure 4.35.: Detailed block diagram of the PLL structure.

In this section the design of each component of the PLL is presented, that is the mixer, the Low-Pass Filter (LPF), the VCO, the common-mode level shifter and differential voltage divider. Furthermore the digital control unit composed of a 4-state FSM and the read-out circuitry will be presented as well as the layout of the overall chip.

VCO design: The VCO is a Voltage Controlled Ring Oscillator (VCRO) modified from the design given in [Lu 05]. In order to have low values of phase noise a differential structure has been considered in a $0.35\ \mu\text{m}$ technology and it has been tuned for a central output frequency of 259 MHz. The structure consists of a loop containing two differential delay cells. The instability, so the oscillation, is ensured by inverting the connections in the second delay cell. Each delay cell is made of four inverters as shown in Figure 4.36. The pairs of transistors M1 and M4 are used as input inverting stages supplied by the control current $I_{TUNE D}$. The pairs of transistors M2 and M3 are typical cross-coupled CMOS inverters that perform as a latch. The latch outputs correspond to the output of the delay cell. This configuration allows the outputs to be stabilized with the maximum power supply, thus reducing the phase noise. The oscillation frequency is set by means of the ratio between latch and input inverter sizes.

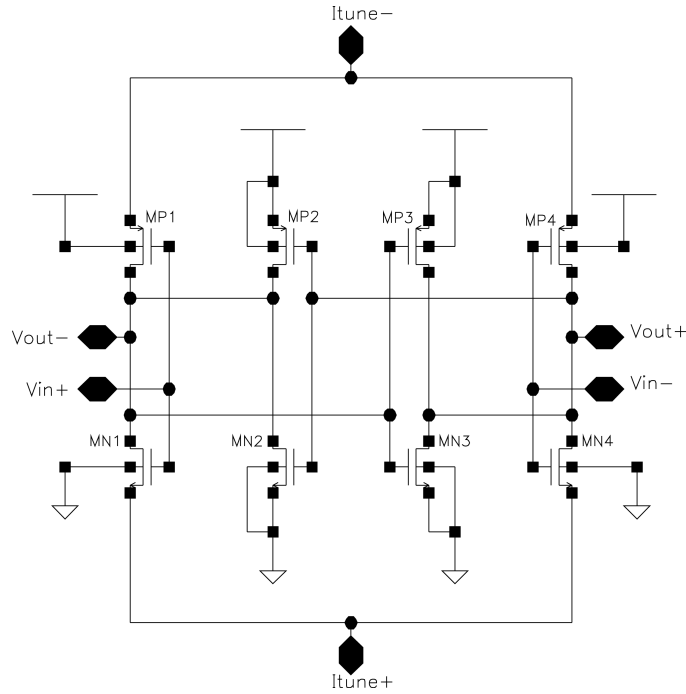


Figure 4.36.: Low phase noise differential delay cell of the VCRO.

Figure 4.37 shows the overall architecture of the VCRO. It consists of two delay cells, an enabling circuit and an output current buffer. The latter is used to drive the input impedance of the SAW sensor without affecting the VCRO internal functioning. The differential input $V_{TUNE D}$ controls the oscillation frequency by setting the current flowing through M6. Similarly the differential input $V_{CTR D}$ allows an external tuning (c.f. M5) needed to correct the effect of the manufacturing process. This adjustment has to be performed finely at the startup in order to ensure a good lock frequency of the PLL.

Figure 4.38 shows the output frequency versus the differential input. The VCRO has been designed in order to have an output frequency for $V_{TUNE D}=0\text{V}$ around the resonance frequency of the SAW sensor. The considered range of the $V_{TUNE D}$ is from -100 mV to $+100\text{ mV}$.

For the VCRO non-linear behavior, the derivative of the frequency over the input voltage has been simulated. Figure 4.39 illustrates the range of the input $V_{TUNE D}$ wherein the deformation versus an ideal linear curve is less than 1%. The dashed curves represent the minus and plus 1%

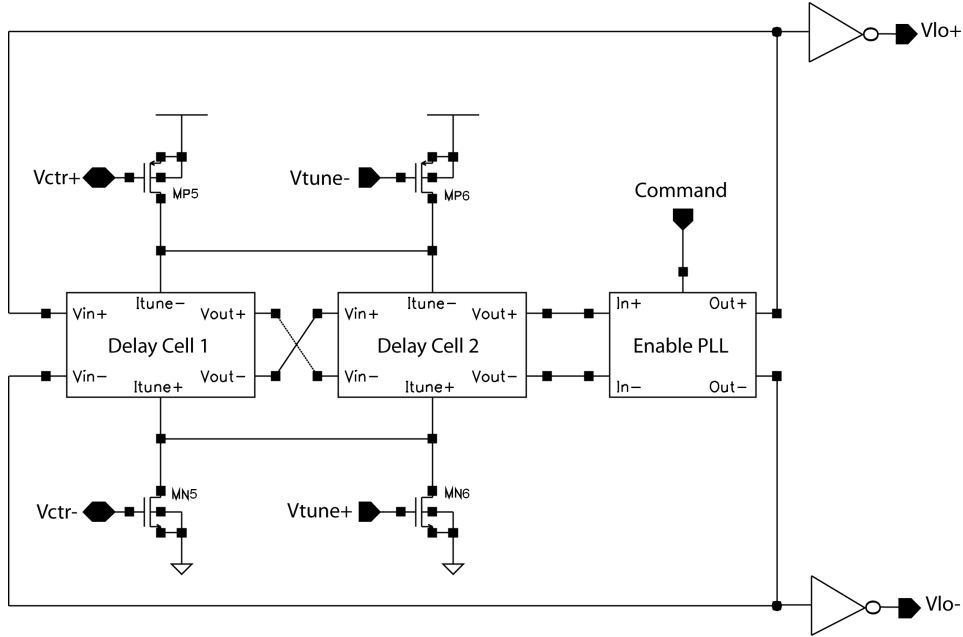


Figure 4.37.: VCRO with differential control and differential external tuning.

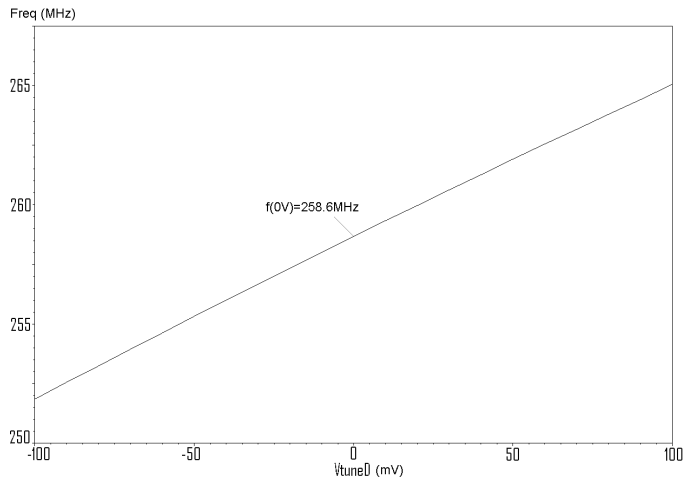


Figure 4.38.: VCRO output frequency versus differential voltage input.

variation. As a result from the figure ΔV is always less than 15 mV, that leads to the assertion that a variation of V_{TUNED} of ± 15 mV still ensures a linearity better than 1%. The gain of the VCRO is 66 MHz/V at the central frequency.

Mixer design: The mixer employed in the frontend electronics is a typical double-balanced Gilbert cell shown in Figure 4.40. The radio frequency (RF) signal coming from the SAW sensor is applied to the transistors MN0 and MN1 that perform a voltage to current conversion. Transistors MN2-MN5 perform a multiplication of the input RF signal and the local oscillator (LO) signal in the current domain. The two load resistors form a current to voltage conversion providing a differential intermediate frequency (IF) signal. The gain of the mixer and its common-mode IF level (V_{IFCM}) are related to the choice of the bias current provided by the

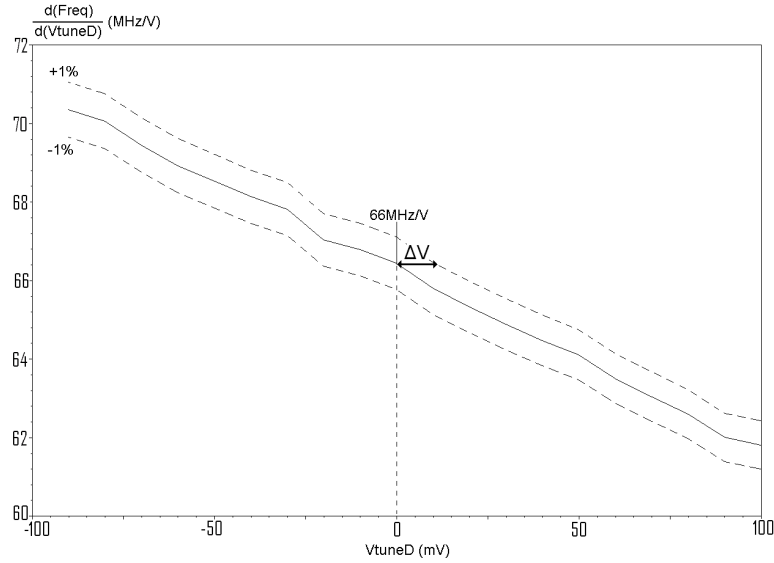


Figure 4.39.: VCRO non-linear behavior.

current mirror and the size of the resistors. In order to obtain $V_{IFCM} = V_{DD}/2 = 1.65$ V as the VCRO requires, the needed values are a bias current of $330 \mu A$ and a resistor value of $10 k\Omega$.

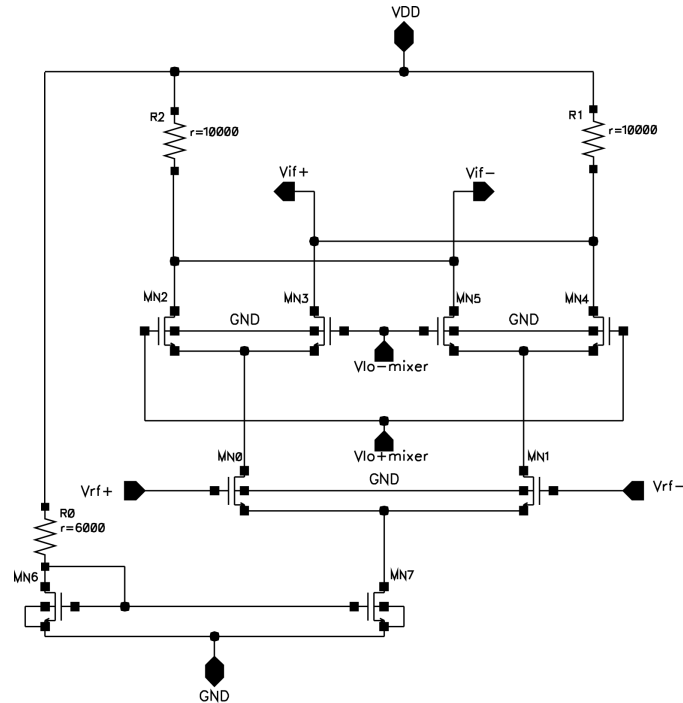


Figure 4.40.: Gilbert cell mixer.

Differential low-pass filter and signal conditioning circuitry: In order to realize the phase detection function, the differential RC low-pass filter shown in Figure 4.41 is placed at the output of the mixer. The differential transfer function is given by

$$\frac{V_{TUNE}D(s)}{V_{IF}D(s)} = \frac{1}{1 + \frac{5}{2}sRC} \quad (4.48)$$

with a cut frequency of $f_{CUT} = \frac{1}{5\pi RC}$. For a cut frequency around 63.7 kHz, the values of $R=100\text{ k}\Omega$ and $C=10\text{ pF}$ have been used.

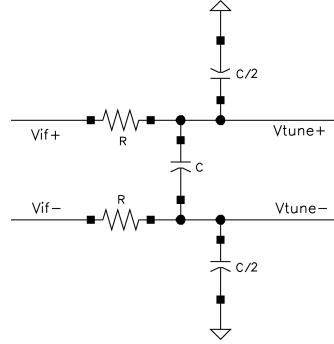


Figure 4.41.: Differential RC low-pass filter.

The VCRO provides a peak-to-peak $V_{LO}D$ signal of $2V_{DD} = 6.6\text{ V}$. Since the maximum differential LO input accepted by the mixer is about 500 mV peak-to-peak it is mandatory to attenuate the LO signal ($V_{LOmixer}D$) to avoid the saturation of the LO input stage of the mixer. Thus, a 3-resistor-chain voltage divider is inserted at the LO input of the mixer. This corresponds to the “differential voltage divider” block of Figure 4.35.

The $V_{LO}D$ signal is attenuated by the SAW sensor to about 190 mV of peak-to-peak amplitude. In order to set the common-mode value of the differential output of the sensor a “common-mode level shifter” is used to provide a common-mode value $V_{RFmixer}CM$ of $V_{DD}/2$ and a differential value $V_{RFmixer}D$ of about 105 mV. The common-mode level shifter block of Figure 4.35 is nothing but a p-MOS amplifier in a common-drain configuration loaded with another diode-connected p-MOS. The transistor sizes are experimentally obtained in order to achieve the right common-mode value.

Design of the digital read-out: In order to measure the gas concentration, a digital read-out circuit has been designed. The idea is to use two counters for the frequency measurement. The frequency of each PLL must be calculated in order to subtract them and obtain the frequency shift. In figure 4.42 the two PLLs and associated circuits are referred to as “active” (A) and “reference” (R). The START signal starts the measurement by enabling the two VCROs of the PLLs. A 13 bit counter called TIMER is used to allow the two PLLs to reach the steady state condition (lock state) before starting the frequency count. The lock phase is over when the TIMER content reaches the value of 2^{12} that is about $60\mu s$ after. The output of the VCRO of each PLL is sent to a frequency divider by four (FD_A and FD_R) and its output is used to increment the two 16 bit counters (C_A and C_R). The countings are stopped as soon as one of the counters sets its most significant bit to 1. Next, the signal “end_measure” is set by the gate P1 and starts the FSM control.

Similarly, the gate P2 selects which counter (between C_A and C_R) is the one that contains the information. Only the less significant byte of the selected counter is kept. The

output byte (bus_conv) is then obtained computing the 2's complement of the data. Finally, the bit READY is set by the FSM when the measure is available on the output (S). Counters C_A and C_R are 16 bit sized in order to detect a minimum gas concentration of 0.1 *ppm*. The 8 bit sized output is the result of the difference between the two counter contents. It allows a gas concentration detection up to 25 *ppm* even if the gas concentration should not exceed 10 *ppm* to ensure the linearity of the measurement.

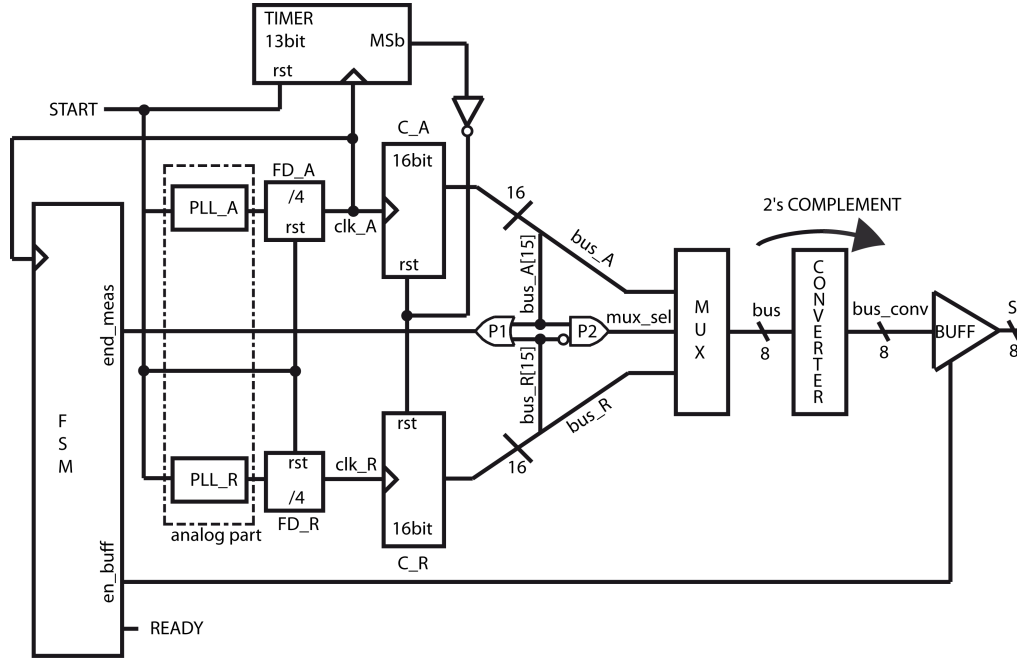


Figure 4.42.: Digital read-out and control unit of the overall system.

The architecture of the digital read-out has been coded in the hardware description language VHDL. The behavioral simulation has been carried out using ModelSim. The code has been optimized and the gate-level circuit has been synthesized by means of Design Vision (graphical user interface to Synopsys synthesis environment). The right functioning of the circuit has been validated considering the worst-case simulations, thus the critical path delay.

Layout of the chip: The circuit is designed in a 0.35 μm standard CMOS technology. The layout of the analog part is hand-made while the layout of the digital part has been conceived using the Cadence Soc Encounter tool. The GDS file coming from the Cadence Soc Encounter tool has been imported to the Cadence layout environment. The layout of the overall chip is shown in Figure 4.43. It contains the analog control PADs (V_{CTR} for the VCRO), the analog PADs from and towards the SAW sensor (external component fabricated on a GaN substrate), the digital PADs (READY, START, output byte, DV_{DD} , $DGND$), the two PLLs and the read-out circuit. The total area is about 1.5 mm^2 .

Simulation results

The simulations presented in this section are post-layout simulations of the PLL circuit. First, the functioning of the circuit is validated through a transient simulation. The variation of V_{TUNED} is shown in Figure 4.44. After an initial transient period, the lock state of the PLL is

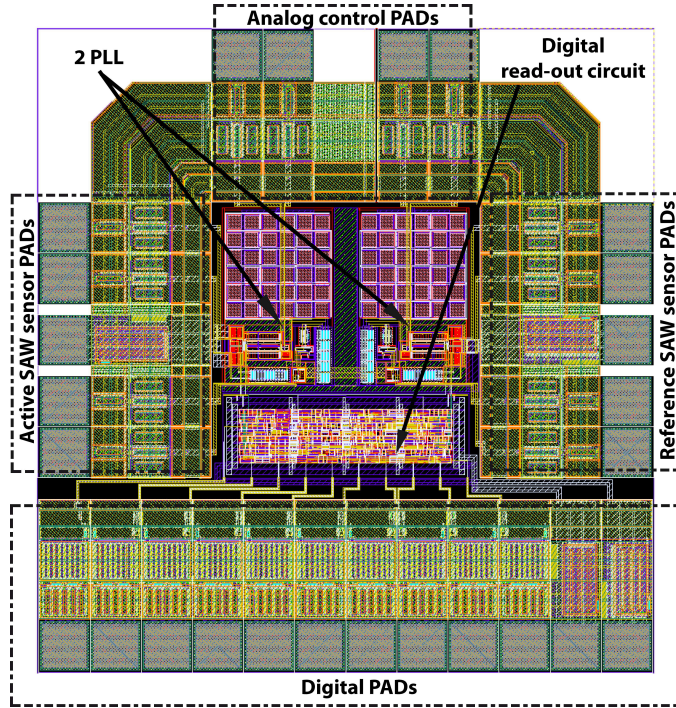


Figure 4.43.: Layout of the overall chip.

reached in conditions of no gas concentration, thus no added delay in the SAW Verilog-A model. The circuit reaches a periodic steady state after about $5 \mu s$ at a frequency of $259.2897 MHz$. At $10 \mu s$ a $10 ppm$ variation of gas concentration is emulated by adding a $1.07 ns$ delay to the SAW Verilog-A model. The PLL reaches another lock state at about $15 \mu s$ at the frequency of $258.509 MHz$. This results in a V_{TUNED} voltage drop of $14.1 mV$. This value is lower than $15 mV$ that ensure a linearity of the VCRO better than 1%.

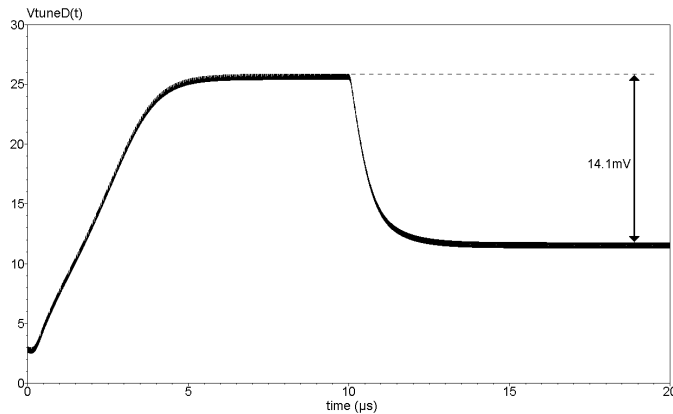


Figure 4.44.: Simulation of a 10ppm gas concentration variation.

Table 4.2 shows the results of transient simulations for different values of simulated gas concentration. The nominal frequency is the lock frequency in no gas conditions. In the case of a gas concentration of $0.1 ppm$ the frequency shift is $8.2 kHz$. In the case of $1 ppm$, the frequency shift is around ten times higher. However, for a $10 ppm$ concentration, the frequency

Gas concentration	Nominal frequency	Lock frequency	Δf
[ppm]	[MHz]	[MHz]	[kHz]
10	259.2897	258.5090	780.7
1		259.2072	82.5
0.1		259.2815	8.2

Table 4.2.: Frequency shifts for different simulated gas concentration.

shift with respect to a 1 *ppm* concentration roughly increases by a factor of 10, and the non linearity reaches about 5%. The microelectronics frontend has a conversion factor of about 80 *kHz/ppm*, and linearity decreases as the gas concentration increases.

Figure 4.45 shows the phase noise referred to the output of the VCRO in conditions of no gas (carrier frequency of 259.2897 MHz).

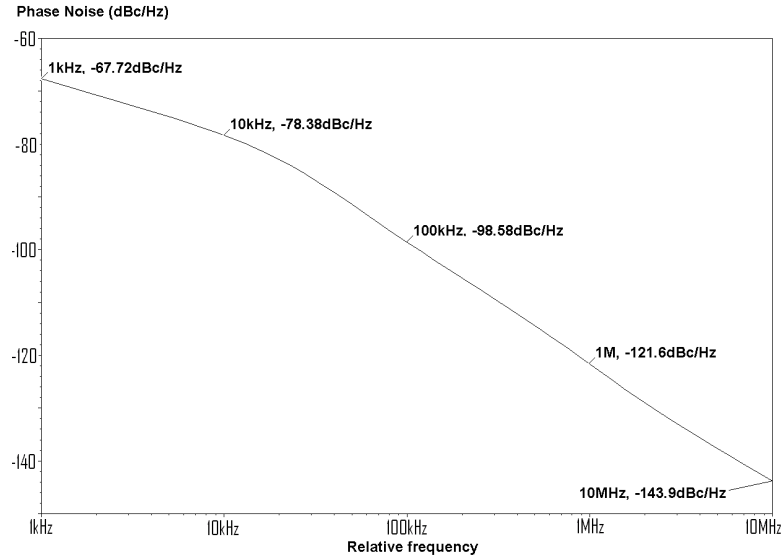


Figure 4.45.: Phase noise analysis on the V_{LOD} signal.

The phase noise allows us to obtain the sensitivity of the PLL (minimum detectable gas concentration). Considering the phase noise power P_{dBc} in the band from 1 *kHz* to 10 *MHz* from the carrier f_c , the RMS jitter J_{rms} given by Equation 4.49 [Kester 09] results in about 20 *ps*. Since the conversion factor is of about 100 *ps/ppm*, we will have a resolution of about ± 0.2 *ppm*. It must be noticed that changes of the gas concentration result in a reduction of the lock frequency that still remains in the sensor pass band.

$$J_{rms} = \frac{\sqrt{2 \cdot 10^{\frac{P_{dBc}}{10}}}}{2\pi \cdot f_c} = \frac{\sqrt{2 \cdot 10^{\frac{-32.53}{10}}}}{2\pi \cdot 259.2897 \cdot 10^6} \quad (4.49)$$

Test results

Figure 4.46 shows the fabricated chip with the microelectronics frontend for the chemical sensor. We have carried out an electrical test of this frontend, without yet considering the SAW sensor. Figure 4.46 also shows the pads that have been used for an analog test of one of the chip PLLs. These pads include the analog power supplies ($AV_{DD} = 3.3$ V and $AGND$), the VCRO frequency tuning pads (V_{CTR+} connected to V_{DD} and V_{CTR-} connected to GND), the pads to/from the SAW sensor (V_{LO} and V_{RF} differential signals) and the START signal.

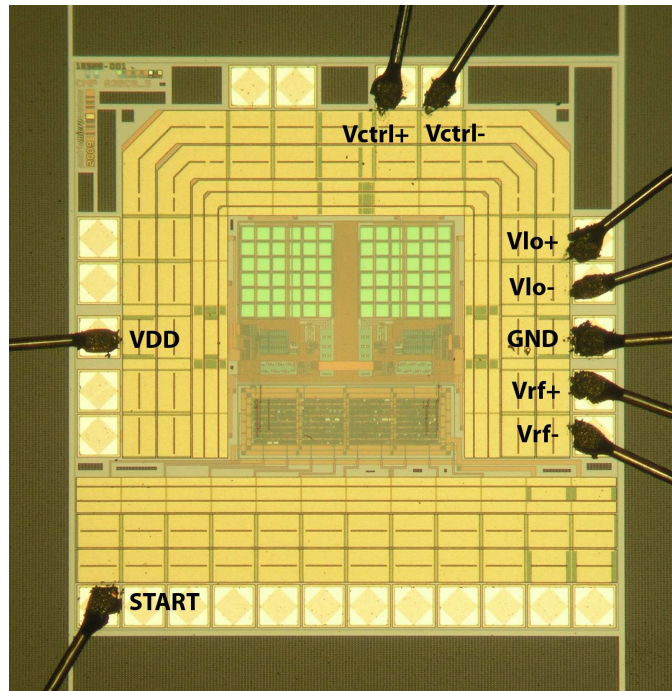


Figure 4.46.: Photo of the fabricated chip with the pads used for electrical testing.

For the test of the PLL frontend, we have used the test bench shown in Figure 4.47. The V_{RF} single-ended sinusoidal signal comes from a signal generator. A balun (SE-D) is used to convert this signal into a differential signal. The V_{LO} differential signal of the PLL is also converted by means of another balun (D-SE) into a single-ended signal that is monitored with a spectrum analyzer or an oscilloscope.

The chip has been glued on an Evaluation Printed Circuit Board (EPCB) shown in Figure 4.48. The electrical pads needed for the validation of one PLL have been wire bonded to the EPCB lead frame. The EPCB contains the two baluns and some additional components to form a pass-band filter that can be used as an emulator of the SAW chemical device. Finally, the EPCB is inserted into a commercial metal box for preventing external interferences.

Experimental measurements show that the PLL locks and provides a steady state signal at its output when the input signal V_{RF} is within the bandwidth of the PLL. By sweeping the input signal with steps of 10 kHz, the PLL has shown a lock range from 240.13 MHz to 254.55 MHz, below the design nominal frequency. The VCRO tuning terminals can be used to shift up the PLL lock range, in order to cover the frequency band of the chemical sensor.

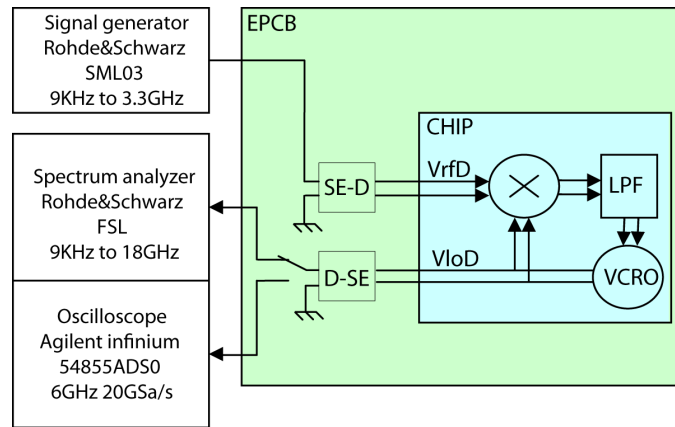


Figure 4.47.: Experimental test bench.

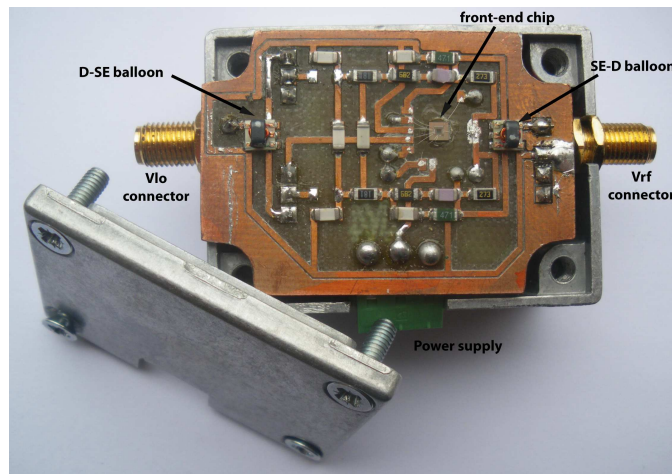


Figure 4.48.: Photo of the EPCB inside the protection metal box.

Figure 4.49 shows the spectrum of the V_{LO} output signal using a span of 2 MHz . The input signal has an amplitude of 0 dBm and a frequency of 245.7 MHz . Since the design of the EPCB is not optimized, the output signal has significant levels of distortion and noise which prevent a direct comparison with simulation results. Unexpected lobes are also observed due to interferences probably caused by the EPCB.

The waveform of the V_{LO} signal, as captured by means of the oscilloscope, is shown in Figure 4.50. The expected square wave has been distorted, due to the test board. While the frequency of the input signal is 245.7 MHz , the average output frequency calculated by the oscilloscope is slightly different. The peak-to-peak amplitude obtained is 1.15 V , lower than the expected $2 \cdot V_{DD}$.

Conclusions and exploitability

A reduced-order behavioral model of the SAW sensor aimed at gas detection has been developed for performing an overall simulation of its microelectronics interface. A Verilog-A behavioral

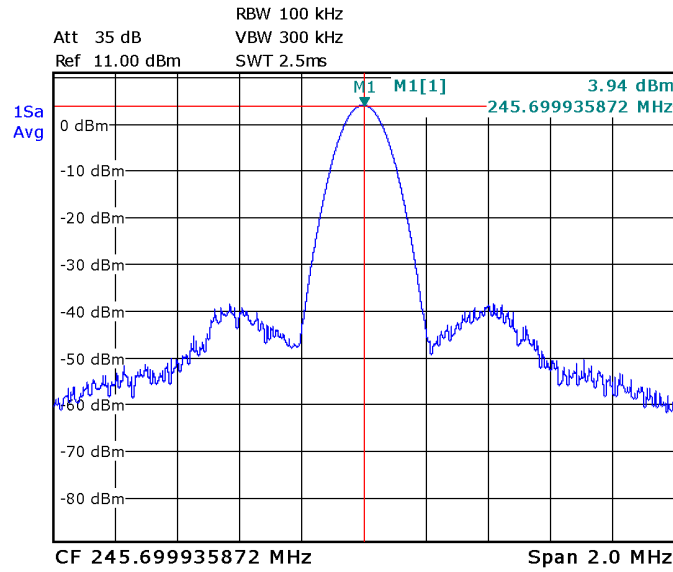


Figure 4.49.: Spectrum of the VCRO output obtained by the spectrum analyzer, with a span of 2 MHz.

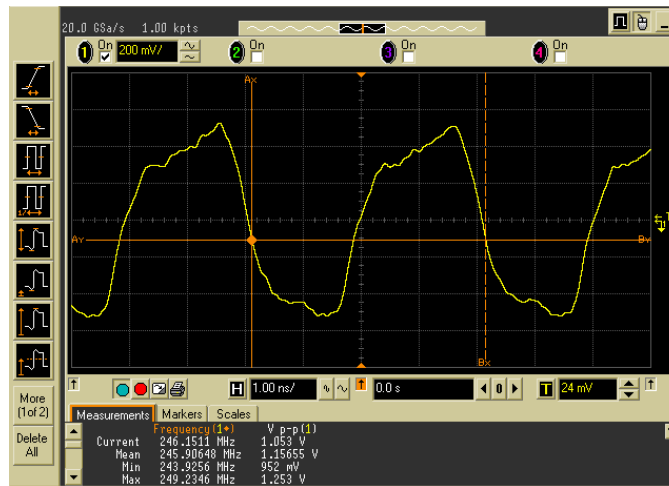


Figure 4.50.: Waveform of the VCRO output obtained with the oscilloscope.

model has been obtained as an approximation of the crossed-field Mason model for a SAW sensor. The PLL-based microelectronics interface has been designed at transistor level and post-layout simulations have been performed using the Verilog-A model. The simulation results allowed to show a potential resolution of the chemical sensor of about 0.2 *ppm*, with a linearity better than 5% up to 10 *ppm* of gas concentration. Real prototypes of the microelectronics interface chip have been fabricated and electrical tests performed. The PLL frontend has shown a suitable lock range that can be shifted to fit the requirements of the actual SAW sensor.

4.6.5. High-level SystemC / SystemC AMS based modeling

The simulation of the microelectronics frontend using a Verilog-A / Cadence-Spectre joint simulation allows simulating the frontend only in significant amount of simulation times, since a transistor level simulation is involved. Indeed, a higher abstraction level than the one offered by this Verilog-A / Cadence-Spectre simulation is needed if system level simulations have to be done. For instance let's suppose that a WSN node containing this sensor frontend has to be simulated together with the the rest of the HW and the embedded SW, and maybe together with several WSN nodes including an RF channel model a higher abstraction level is needed. Such a scenario would be similar to the WSN scenario shown in section 5.6. The main difference is the type of sensors integrated in the WSN nodes. For the case of the pre-collision mitigation braking system of section 5.6 [Lévêque 12], CMOS image sensors are embedded on each WSN node while in this scenario a WSN of chemical sensing nodes is imagined.

For this purpose the frontend of a chemical sensor described above has also been modeled using the SystemC AMS analog and mixed-signal extensions to the SystemC kernel. The concepts describing the methodology for behavioral modeling of both dynamic and static analog components has been formalized in sections 4.3.1 and 4.3.2 respectively. In the following the methodology is applied to the SAW-based chemical sensor case study for the SystemC AMS high-level modeling of the microelectronics frontend [Cenni 09b].

In this section it will be shown how the concepts explained for the modeling of both static and dynamic analog components are used in the case of the frontend of the SAW sensor that is intended to be simulated in a WSN application.

As already presented in section 4.6.3 a rational Laplace transfer function was obtained and the Matlab script file (see section 4.3.1 and Annex A.3) automatically generates a .h file (Annex A.4) starting from the transfer function equation, the result file contains the code of the SystemC AMS reduced order model of the SAW filter. For the reasons discussed in section 4.6.3, the effect of a change in gas concentration has been modeled with an additional SystemC AMS delay module that is placed in cascade with the fitted SAW sensor model. The value of the delay is proportional to the gas concentration detected.

Microelectronics interface for the SAW sensor

As already mentioned, in order to measure the input-output delay provided by the SAW the phase-locked loop (PLL) architecture of Figure 4.33 has been used [Sternhagen 02]. The microelectronics interface architecture places the SAW sensor in a PLL. The PLL includes a voltage controlled oscillator (VCO), a phase detector (PD) and a frequency counter. As shown in Figure 4.33 the PD output is used in a closed loop to maintain a constant phase shift across the SAW sensor by controlling the voltage V_{tune} , thus the loop frequency. Changes of the loop frequency are proportional to changes of the SAW velocity. The PLL-based architecture offers both robustness and high resolution thanks to the frequency output.

From the SystemC AMS modeling viewpoint the mixer is a `SCA_TDF` module that performs a multiplication between its two input values at each computation time step. The low-pass filter is implemented by means of a 1 pole Laplace transfer function instantiated within a `SCA_TDF` module.

SystemC AMS modeling of the VCO: since the microelectronics frontend for a SAW chemical sensor has already been designed at transistor level [Cenni 09a], the SystemC AMS modeling of the interface comes from the need of a behavioral model to be simulated inside the digital environment of a WSN node. In order to keep the SystemC AMS model of the VCO as accurate as possible to the behavior of the VCO designed at transistor level the macro-modeling technique for static block seen in section 4.3.2 is exploited. Figure 4.51 shows the simulation of the VCO designed at transistor level (dashed line) and the result of the 5th-order polynomial interpolation (continuous line) used to model the relation “output frequency vs V_{tune} ”. As a specification of the generic code of Listing 4.4, Listing 4.6 shows the code for the VCO modeling. For each computational time steps, in the signal processing function of an SCA_TDF module, the output frequency is calculated by means of the polynomial in V_{tune} obtained through the interpolation.

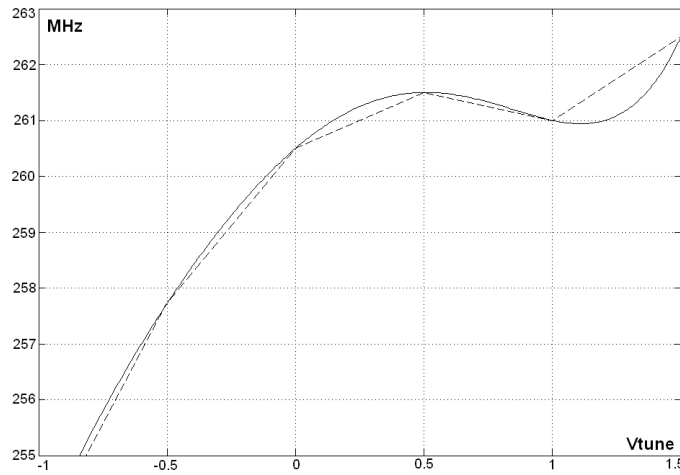


Figure 4.51.: Polynomial interpolation of the VCO static behavior.

Listing 4.6: SystemC AMS modeling of the VCO static behavior using a polynomial interpolation.

```

1 SCA_TDF_MODULE(vco)
2 {...
3 double gain;
4 void processing() {
5     double t = get_time().to_seconds(); // actual time
6     double input = in.read();           // control voltage Vtune
7     double wvco = polynomial_function(input); // angular velocity
8     out.write(gain * sin(wvco*t));      // sinusoidal output
9 }
10 };

```

Overall chemical sensor

The overall detection architecture is composed by two PLL loops as shown in Figure 4.52. Since SAW devices are sensitive to temperature changes, two identical measurement units are used in order to cancel temperature effects. Both sensors are fabricated on the same substrate. While only one is coated with a sensitive layer (indicated as “active” in Figure 4.52), the other is not (“reference”). The difference of SAW velocities is only associated to the absorption of the target

gas since temperature changes affect the two sensors in the same way. Frequency measurements are obtained by means of two frequency counters that use the zero-crossing detection principle. For each sensor, the number of crossing points counted in a given time frame is stored and, after the counting phase, the content of both frequency counters is subtracted. The result of the subtraction is then divided by the temporal frame time in order to get the frequency shift between the two loops. That is, the counting time is inversely proportional to the frequency resolution. However, it must be noted that a constant part of the difference of SAW velocities is inherently caused by the difference of structure of the sensors. Therefore when the detected concentration in the active SAW is zero a difference of frequencies between the two loops exists. This frequency “offset” may be compensated by software once the frequency offset is known.

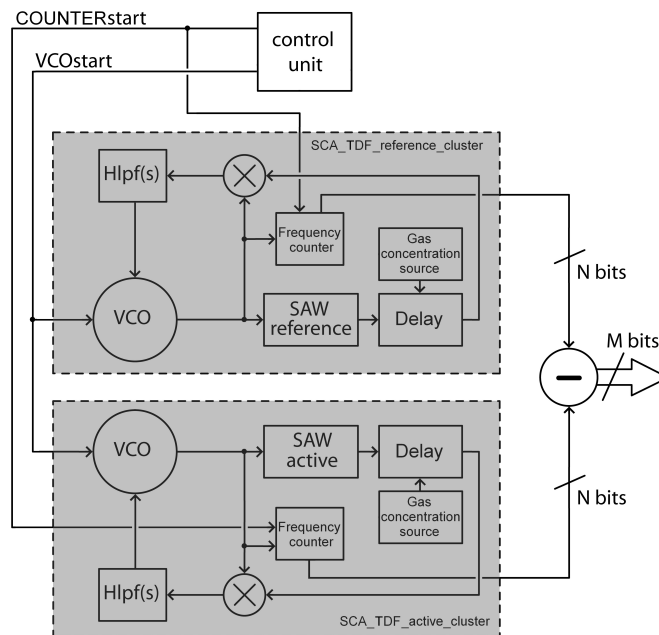


Figure 4.52.: SystemC model of the overall chemical sensor.

In Figure 4.52 the control unit manages two signals through which the measurement system is switched on and off. The “VCOstart” signal activates the VCO on its positive edge and stops it on the negative one. As already seen for the VCO the frequency counter is controlled with the same principle. The counter must start counting on the “COUNTERstart” positive edge and stop on the negative one. From the modeling point of view the two loops that contain the VCO, SAW model, delay, gas concentration source, mixer, low-pass filter and the frequency counter are “SCA_TDF_clusters”. The clusters in question are defined as SC_MODULES wherein the models just mentioned (SCA_TDF_MODULES) are instantiated. In the “sc_main” four entities are instantiated: the control unit which is a pure SC_MODULE with two outputs, the two SCA_TDF_clusters and the subtracter. The gas concentration sources inside the two clusters are set as “no added delay” for the reference cluster and a “test delay” for the active one. The control unit schedules the phases of the simulation through its implementation as shown in Figure 4.53. A measuring cycle starts with the positive edge of “VCOstart” thus entering in the S0 state. Afterwards a transient period is necessary in order to reach the periodic steady states, one for each loop. The transient period is called “lock_time” and it is extracted from experimental observations. Once the lock has been reached the “COUNTERstart” goes high (state S1) thus the frequency counters start counting for a time depending on the required

resolution. Afterwards the “COUNTERstart” signal goes low (state S3) and the “VCOstart” signal stays high for a brief phase wherein the loops keep running (VCO running) while the counting is already finished. In the S3 phase the subtraction between the two frequency counter contents is performed. Finally the “VCOstart” goes low and the measurement cycle is over.

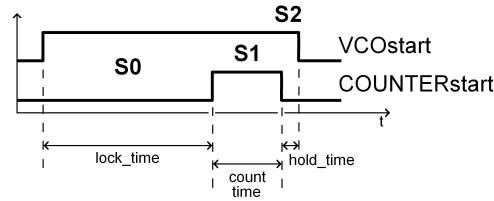


Figure 4.53.: Control unit outputs.

Simulation results

The simulations shown in Figure 4.54 are carried out using two different gas concentration values (delay values). The low-pass filter outputs, that is the VCO inputs V_{tune} , are plotted. The line on the top is the low-pass filter output of the reference loop, so with no added delay. The line in the center is related to an active loop detecting a gas amount that causes an added delay corresponding to a $\frac{\pi}{4}$ rad phase. Similarly, the bottom line corresponds to an active loop for a $\frac{\pi}{2}$ rad phase. Figure 4.54 shows that the steady state is reached in the three cases. V_{tune} is directly linked to the frequency of the loop since it is the input of the VCO. The reference loop has the highest frequency and adding delay to the output of the SAW sensor reflects in a decrease of the loop frequency. Therefore the “lock_time” increases for increasing gas concentrations.

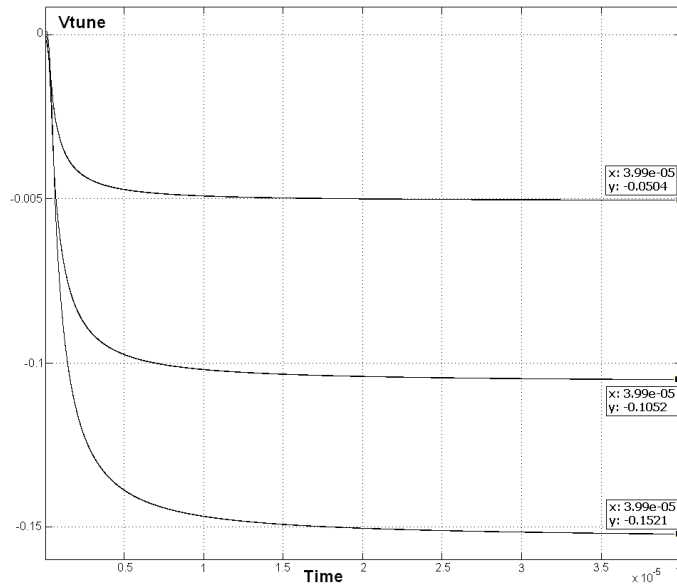


Figure 4.54.: Low-pass filter outputs.

Figure 4.55 shows the two control signals “VCOstart” and “COUNTERstart”, the outputs of the frequency counters and the result of their difference. In the simulation the injected delay

Timing diagram showing VCOstart, COUNTERstart, active_count, reference_count, and subtraction signals over time. The diagram is divided into three sections: 0 ns to 5 ns, 5 ns to 25010 ns, and 25010 ns to 40 us. The signals are: VCOstart (0 to 1), COUNTERstart (0 to 1), active_count (0 to 1), reference_count (0 to 1), and subtraction (0 to 1). The active_count and reference_count signals are shown with values 1, 2, and 3. The subtraction signal is shown with values 1, 2, 3, 4, 5, and 6.

4.6.6. Conclusions

Conclusions and perspectives for the frontend chip manufactured and tested: Further work will consider the integration of the microelectronics frontend with two real SAW device test chips, one of them will be coated with the sensitive golden layer and the other will not. Tests will then be performed in a chemical chamber in order to characterize the overall device.

Conclusions of the SystemC / SystemC AMS based modeling: The facilities offered by the SystemC AMS libraries are still in phase of development thus more and more updates are progressively coming out. Using the available tools of the current release of SystemC AMS it has been shown how macro-models for static and dynamic blocks can be obtained.

For the case of dynamic blocks, a way to generate these models from their detailed frequency response has been presented. These modeling techniques have been illustrated for the case of a complex chemical sensor that is intended to be simulated in a WSN application.

4.7. Summary

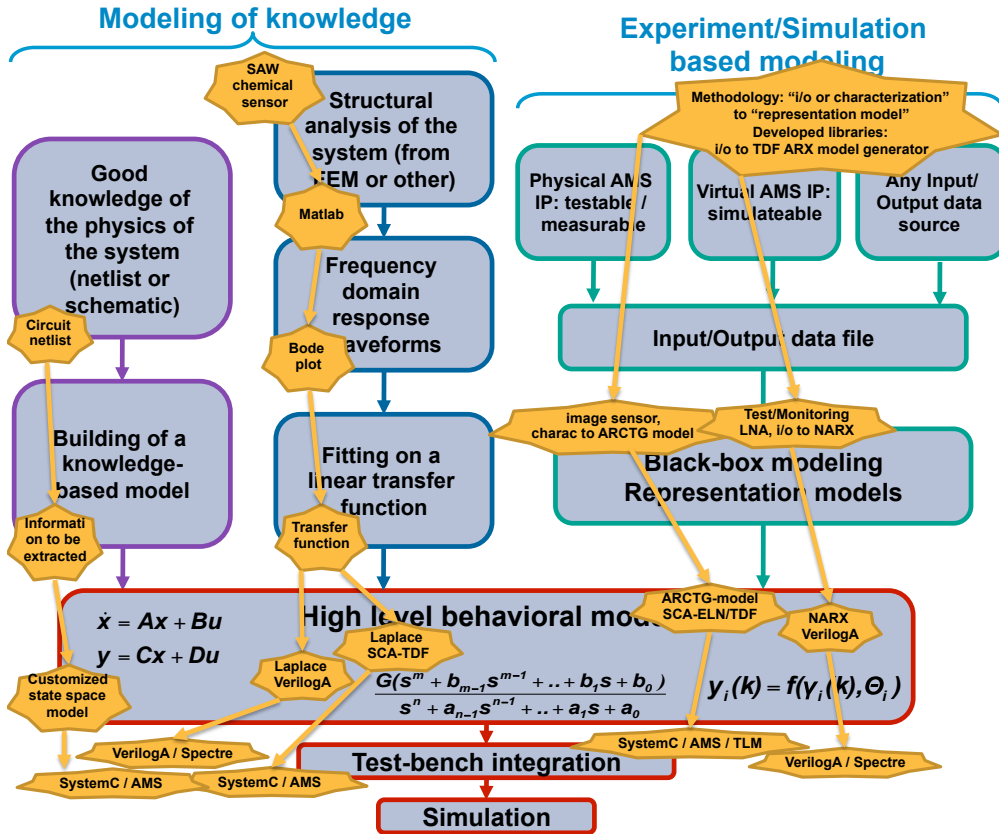


Figure 4.56.: Case studies for the validation of analog behavioral modeling flows.

Figure 4.56 depicts our contributions inserted in different behavioral modeling techniques. In the frame of a *modeling of knowledge* the methodology for the *macro-model building from frequency responses* (blue branch) is applied to a Surface Acoustic Wave (SAW) based chemical sensor case study; the sensor itself has been modeled and simulated together with its micro-electronic frontend interface and fabricated. For the case of a modeling of knowledges from a starting *netlist* the procedure for a customized state space model extraction has been studied and implemented for the SystemC AMS environment allowing to generate an instantaneous power consumption signal.

In the scenario of an *experiment/simulation-based modeling*, an extension library for building Auto-Regressive models with eXogenous (ARX) variable was developed and the methodology of black-box modeling is applied to an LNA case study. On the extreme right of Figure 4.56 a behavioral model of a Low Noise Amplifier (LNA) is obtained via a system identification by means of *simulation data* not only for simulation purposes in an overall RF transceiver but

also for adaptively monitoring and controlling the LNA performances. The concepts of the statistics are here exploited for monitoring, test and performances control purposes. Chapter 5 will present the industrial CMOS image sensor case-study. Behavioral models of the latter were developed by means of *experimental data* issued by an opto-electrical characterization of the response to the light of the image sensor.

Chapter 5.

Industrial case study: CMOS video sensor

In this chapter a CMOS image sensor (CIS) based video acquisition platform is modeled using the SystemC framework. An image acquisition system is composed by three main blocks: an image sensor followed by an image signal processor (ISP) and a central processing unit (CPU).

This work focuses on the development of a reliable and accurate behavioral model of an STMicroelectronics CIS with the purpose to simulate its functioning within its surrounding environment. Nowadays the embedded software development starts only when the CIS is physically available. The software is debugged and validated by means of image sets issued by a real sensor. If delays are accumulated during the design process there won't be much time left for software development, this will either result in a software non-reliability or in a time-to-market miss if the market is not hit in time with respect to the competitors. Since the ISP must also be able to control the sensor parameters, the interoperability between sensor and ISP is normally checked by connecting a board with the sensor chip and the ISP implemented on field-programmable gate array (FPGA) to the PC-hosted instruction set simulator (ISS) of the target CPU through an interface board. A virtual prototype of the overall system would avoid, at early design stages, the use of boards for a first validation of the system. This would also allow to validate the ISP algorithms at simulation-level by tuning them and stressing them up to determine their limit working conditions. With respect to a CIS, analog and digital electrical signals are involved together with analog optical values.

The model of the CIS describes many optical and electrical effects at a high level of abstraction, going beyond the limitations of the previously developed VHDL-AMS model. Section 5.1 describes the target CIS, the VHDL-AMS starting model and the SystemC AMS models using different models of computation are shown. Section 5.2 describes the fastest SystemC AMS TDF-based model in all its blocks. Section 5.3 will show the comparison among the CIS modeling styles analyzed in terms of their simulation performance and accuracy. Section 5.4 will generically discuss the issues to be faced when integrating the SystemC AMS model inside a SystemC platform, the section will also illustrate the types of platform target for the integration. Sections 5.5, 5.6, 5.7, 5.8 will describe each integration in such platforms. Finally the conclusions will be given.

5.1. CMOS video sensor and SystemC AMS models

The CMOS video sensor studied is designed by STMicroelectronics in its 90nm imaging-specific CMOS technology. The size of each pixel of the matrix is $1.4\mu\text{m} \times 1.4\mu\text{m}$. The pixel architecture

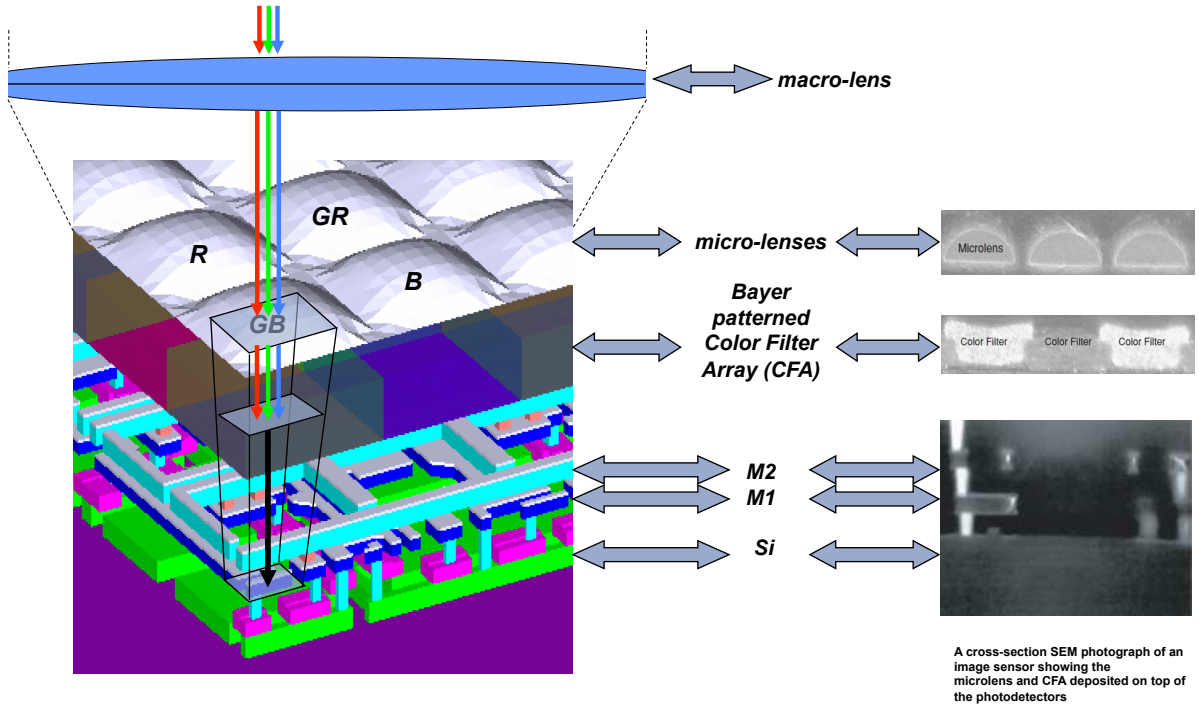


Figure 5.2.: Section of the optical path of a CMOS image sensor [Cohen 06].

the Y-decoder) had been accurately described. The VHDL-AMS model had high simulation duration due to the accurate description of the control signals timing and the differential equation system to be solved for each photodiode. The simulation time was about 2 hours and 30 minutes for a 2 by 496 pixel array as shown in Table 5.1, but allowed to accurately verify the digital control. Other VHDL-AMS models of image sensors can be found in the literature [Dadouche 08, Navarro 05], typically with simulation times of a few minutes for a 100 by 100 array [Navarro 05].

In the following, the SystemC AMS models evolution is regrouped by MoC and sorted by increasing level of abstraction which also reflects the chronological order of development.

5.1.2. SystemC AMS ELN-TDF model

Following the same principle of the VHDL-AMS a SystemC AMS ELN-TDF first model has been developed by reproducing the schematic 1T75 architecture of the pixel. As shown in Fig. 5.3, for what concerns the conservative domain of energies, each photodiode has been described as a capacitor charged by a current source and a parallel R_{on}/R_{off} switch for its reset. The charging current value is regarded as constant between two image captures. A snippet of the code is shown in Listing 5.1 for the conservative part, the four components (*capacitor*, *current_gen*, *switch* and *converter*) are instantiated and connected as shown in Fig. 5.3 for each photodiode.

The ELN voltage across each capacitor was converted to a TDF signal and the timing of transfer-gate (TG) signals and the other read-out control signals managed the driving of the

Vx column line. Those read-out control signals are provided by the video timing block, the timing of these signals had been accurately described.

The accuracy and simulation time of the model depend on the chosen TDF time step. For a good accuracy, a TDF time step of $0.5 \mu\text{s}$ has been chosen, which allows to simulate a 48×48 pixel array within ten minutes. Both the VHDL-AMS and the SystemC AMS ELN-TDF models describe the sensor at a low level of abstraction comparable to the register-transfer architectural level (RTL). The capability of SystemC AMS to cover RTL level descriptions is proved. The SystemC AMS ELN-TDF model allows to gain a speed-up factor of about 35 times compared to the VHDL-AMS model but the performances really depend on the time step. Despite that, the desired SystemC AMS model is intended to raise the level of abstraction even further in order to be able to perform simulations of the feedback control loop (CIS/ISP) while keeping a (relatively) accurate modeling of the analog behavior of the sensor. Therefore, a TDF-based model has been studied.

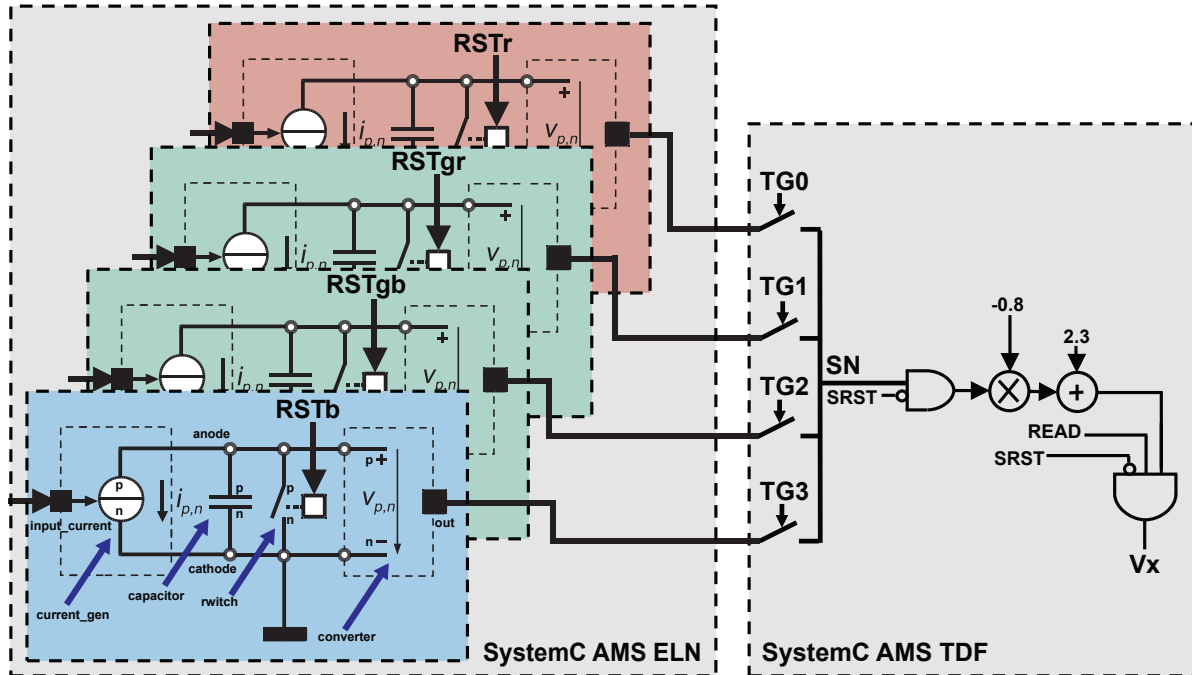


Figure 5.3.: Structure of SystemC AMS ELN-TDF model.

Listing 5.1: Structure of the ELN based photodiode model.

```

1 SC_MODULE(photo_diode)
2 { // terminal declaration
3   sca_eln::sca_terminal  anode;
4   sca_eln::sca_terminal  cathode;
5   sc_core::sc_in<bool>   rst;
6   sca_tdf::sc_in<double> input_current;
7   sca_tdf::sc_out<double> out;
8
9   // child module declaration
10  sca_eln::sca_c*         capacitor;
11  sca_eln::sca_tdf_isource* current_gen; //Current source driven by a TDF input signal
12  sca_eln::sca_de_rswitch* rswitch; //Switch controlled by a discrete-event input signal
13  sca_eln::sca_tdf_vsink* converter; //Converts voltage to a TDF output signal

```

```

14
15 SC_CTOR(photo_diode): anode("anode"),cathode("cathode")
16 {
17     current_gen = new sca_eln::sca_tdf_isource("current_gen");
18     current_gen -> p(anode);
19     current_gen -> n(cathode);
20     current_gen -> ctrl(input_current);
21     //photodiode capacitance=1fF
22     capacitor = new sca_eln::sca_c("capacitor",1.0e-15);
23     capacitor -> p(anode);
24     capacitor -> n(cathode);
25     //false=off-position, Ron=100μΩ, R off=1TΩ
26     rswitch = new sca_eln::sca_de_rswitch("rswitch", 0.1e-3, 1e12);
27     rswitch -> ctrl(rst);
28     rswitch -> p(anode);
29     rswitch -> n(cathode);
30     converter = new sca_eln::sca_tdf_vsink("converter");
31     converter -> p(anode);
32     converter -> n(cathode);
33     converter -> outp(out);
34 }
35 };

```

5.1.3. SystemC AMS TDF models

Entire array instantiated, many samples per discharge

The first TDF model describes the discharge of the photodiodes as a linear sampled discharge generated by a multiplication between the current value due to the light and the time.

A code snippet referring to this modeling style is shown in lines 1–4 of Listing 5.2. The two code lines are intended to be inserted inside the processing method of the photodiode TDF module. *LRT* is the last reset time registered at the front of the reset signal (DE to TDF converter port) and *anode* is the output. At each TDF time step the time elapsed from the *LRT* to the current time is multiplied by the light intensity (*L*) hitting the photodiode and a conversion factor *K* calculated from the pixel sensitivity and the pixel capacitance. The *temp* value is compared to the saturation value *SAT* for implementing the saturation.

The accuracy of the model relies on the granularity of the TDF time step since the read signal could arrive at any time between two samples of the TDF signal. To avoid this, an interpolation between two samples is needed. Alternatively, a future dynamic time step extension to SystemC AMS could be used for introducing a calculation step between the two statically scheduled sampling instants. In this model, the whole array of pixels was represented by a two dimensional array of instances of the pixel module. This model presents a simulation time of about 2 minutes and 40 seconds for a 48 by 48 pixel array using a TDF time step of 0.5 μs for the discharge (Table 5.1).

Listing 5.2: Photodiode code snippets for different TDF-based modeling styles.

```

1 // Uniformly sampled linear discharge
2 temp = L*K*(get_time().to_seconds()-LRT.to_seconds());
3 if (temp>SAT) temp=SAT; //saturation check
4 anode.write(temp);

```

```

5
6 // Final value of the linear discharge
7 IT = integration_time.to_seconds();
8 temp = IT*L*K;
9 if (temp>SAT) temp=SAT; //saturation check
10 anode.write(temp);
11
12 // Final value of the arc-tangent based discharge modeling equation
13 IT = integration_time.to_seconds();
14 S = K*L; //sensitivity [electrons/sec]
15 IP = SAT/S; //intersection point [sec]
16 AF = 0.5 + (atan(150*(IT-IP))/M_PI); //averaging function
17 SN = K*L*IT*(1-AF) + SAT*AF; //sensing node[electrons]
18 anode.write(SN*CVF);

```

One pixel sweeping the array, many samples per discharge

In order to reduce the simulation time, a new modeling style has been introduced. The array of instances is no more fully instantiated. Instead, only one instance of the pixel sweeps the whole array row-after-row, like it occurs in an analog-TV like image refresh. The discharge of the photodiodes is still described as a constant-slope sampled TDF signal and the accuracy of the model still relies on the granularity of the TDF time step. This new modeling style allowed to reduce considerably the simulation time thanks to the elimination of huge amounts of non-relevant processing. It presents a simulation time of about 10 seconds for a 640 by 480 (VGA size) pixel array, with 50 TDF time steps per discharge, (Table 5.1).

One pixel sweeping the array, one sample per discharge

Further improvements have led to consider only the final value of the discharge once the integration time has elapsed. Two different photodiode discharge models have been implemented:

Linear: On the one hand, in order for the simulation to be fast, the discharge is considered linear as far as the saturation occurs with a discontinuity of the first order derivative. The code snippet is shown in lines 6–10 of Listing 5.2. The integration time is stored in the *IT* variable and *anode* is the output. The *temp* value is compared to the saturation value *SAT* for implementing the saturation.

Non-linear: On the other hand, in order for the the model to well represent the real behavior, the optical characterization results are used for building an approximating analytical equation of the photodiode discharge. The model is based on the arc tangent trigonometric function. The algorithm corresponding to the discharge model is shown in lines 12–18 of Listing 5.2. The discharge model will be detailed in section 5.2.5.

The TDF time step is reduced to one TDF calculation per pixel discharge. Therefore a fictitious pixel time (*p_t* in Fig. 5.4) is introduced for representing the SystemC time of processing of one pixel over the SystemC time of processing of the array. The tremendous gain in simulation time allowed to simulate a 1920 by 1080 (2 mega pixels) pixel array within about 7 seconds (Table 5.1) and up to 2 seconds in the optimized model.

5.2. SystemC AMS TDF fastest model

The AMS model is composed by a TDF cluster, as shown in Figure 5.4. Three TDF time steps are mainly present in the model: the frame time (TDF modules fired at the frame rate), the pixel time (TDF modules fired every pixel time) and the row time step for the firing of the bank of ADCs. The frames of the video stream are passed from block to block in the chain. The images can be generated in three ways in the model (see Figure 5.4 for reference):

- *IIB*: the first emulates the capture of a moving object on a fixed background, in the following called *input image builder* (IIB). The IIB is composed of two sub-modules, the *background builder* (BGB) and the *object builder* (OB). The BGB builds the image background and the OB draws an object upon it (a car for instance). The OB also models the electronic rolling shutter (ERS) effect on a running car captured.
- *DB*: the second emulates the situation of the CIS inserted in a dark box with controlled input scenarios and tunable light, in the following called *dark box* (DB). The *environment* (ENV) block supplies the *image loader* (IL) with three TDF control signals. The controls are: what chart has to be loaded among predefined standardized charts (scenarios), what color and intensity the light illuminates the chart with. The DB emulates the functioning of a dark box containing a set of red, green, blue and white light emitting diodes (LEDs) driven by pulse width modulated (PWM) signals.
- *File Loader*: the third allows a direct loading of reference image files.

The image issued by the DB or IIB or the file loader (see Figure 5.4) is coded upon a parametric number of bits (typically between 8 and 10) and sent to the *lens* module. The image is sent to the *Bayer filter* (BF) module which, in turn, sends the data to the inner part of the model where the TDF time step is reduced to the *pixel* time step and the whole matrix is swept by an instance of *pixel* module. The *controller* drives the *pixel* with the light signals coming from the *Bayer filter*. The *pixel* contains four *photodiodes* driven by the four light signals and the integration time information. The light signals are updated at the beginning of each photodiode discharge processing. The value V_x of the discharge process at the end of the integration time is output by each *pixel* following the discharge modeling described in Section 5.2.5. The column voltage V_x is then sent to the bank of ADCs (*ADC* module) and sampled. Finally, a *decimator* is needed to convert the row rate TDF signal into a frame rate SystemC signal.

5.2.1. Input image builder

An input image is needed to emulate a real life scene presented in front of the sensor. It has been decided to internally synthesize the input images instead of loading them from external image files. This prevents the input images from being affected by other noise sources, such as noise resulted from an image format compression or noise present in raw images shot from other sensors.

As mentioned above the IIB is composed of the BGB module which generates the background for the input image and the OB which draws an object upon the background. The background generated is made of 8 horizontal stripes that start with a saturated tint (left side) and gradually

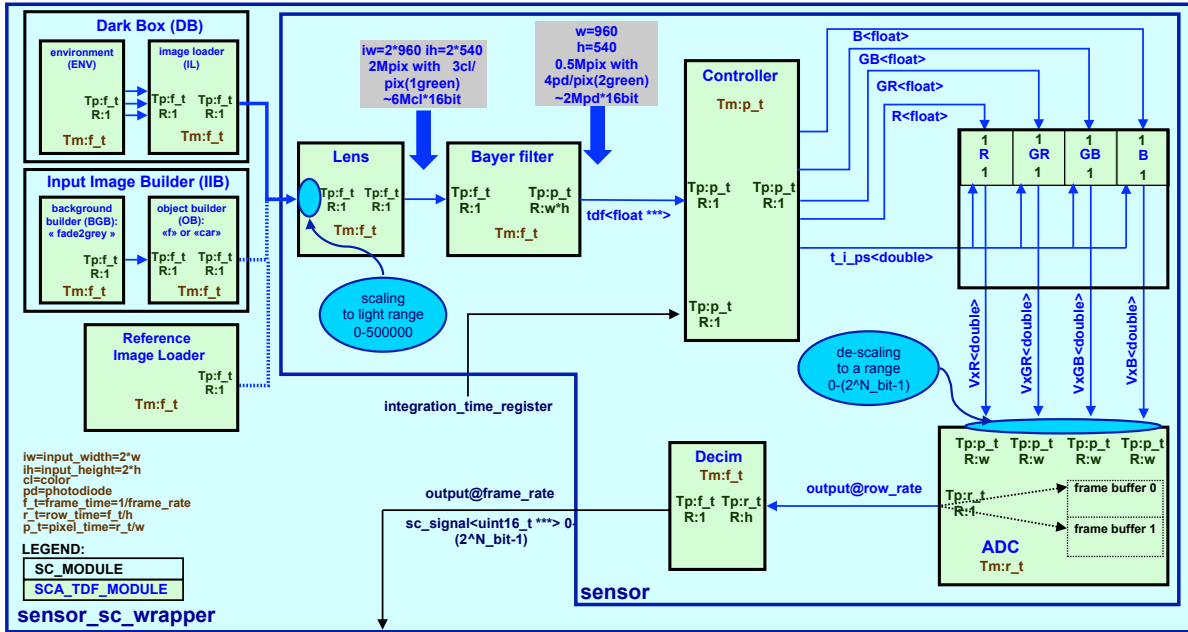


Figure 5.4.: Block diagram of the SystemC AMS model.

fade to a common grey value (right side). The object is superimposed by the OB onto the background image, for test purposes it has been chosen an outlined car whose horizontal offset is shifted by a constant value at every frame.

The OB also takes into account the modeling of the electronic rolling shutter (ERS). This phenomenon is due to the time skew present between the readout of the different lines of the visible array. Its effect is modeled by shifting right the position of the lines of the object of a varying amount that depends on their latitude. The amount is calculated for each row as a function of the speed of the car, the frame time, and the percentage of the frame time needed for the overall matrix scan. The synthesized input image is shown in Figure 5.5. The red, green, and blue (RGB) values are coded upon a parametric number of bits, for test purposes 8 bits have been chosen.

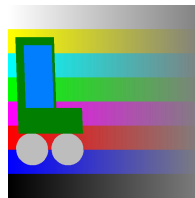


Figure 5.5.: IIB output affected by ERS effect.

5.2.2. Lens effect model

From the modeling viewpoint the light hitting the lens is considered composed of three rays at three given frequencies for being packed in an RGB raw image file (three RGB arrows above the lens in Figure 5.2).

The lens causes a fainting of the light intensity on the periphery of the image compared to the center of the image. This is due to a longer path of the light for reaching the periphery of the CMOS sensor. The visual consequence on the output image is known as natural (or optical) vignetting and it consists on the reduction of the image brightness at the periphery compared to the image center.

There exist other types of vignetting, one over all, of noteworthy interest is the pixel vignetting. It occurs only in digital cameras. It is caused by the sensor nature to produce stronger signal from incident photons at a right angle, than other photons of same energy which reach the sensor by any other angle [Laskaris 05]. The effect of pixel vignetting on the captured image is similar to the natural (or optical) vignetting.

In [Laskaris 05] different modeling techniques are proposed for different sources of vignetting. The attenuation provided by the natural vignetting has been modeled with the cosine of the normalized radius at the power of four as shown by Figure 5.6). R is the distance to the image center and R_{MAX} is the distance at the corners of the image.

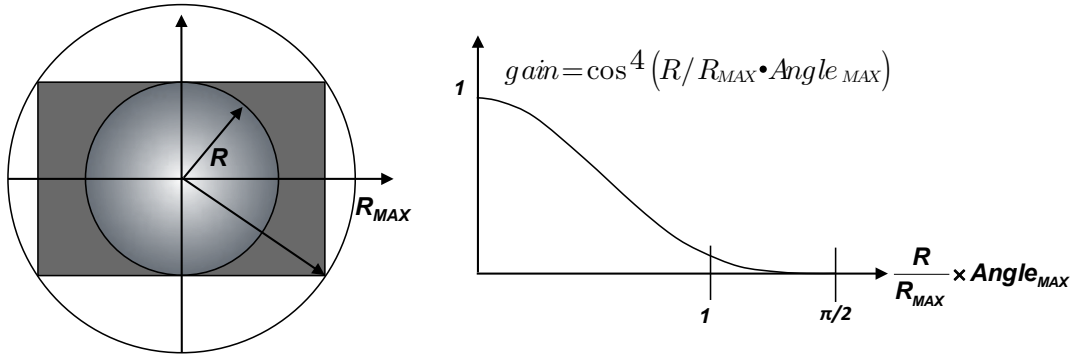


Figure 5.6.: Modeled attenuation of the light intensity.

5.2.3. Bayer filter model

The Bayer patterned Color Filter Array (CFA), well known in the literature, is a mosaic of band-pass light filters. The pattern is composed of quadruplets of light filtering resins, each quadruplet has one blue, one red, and two green resins. The green resin on the line of the red resin is called green-red, the other green is called green-blue. The Bayer pattern is often referred to as R-Gr-Gb-B. Each light filter has a band-pass transfer function, for instance, the red filter pass-band is centered on the red light wavelengths, since the response is not ideal a part of the blue and green wavelengths are absorbed as well.

For modeling the adsorption to the light of the Bayer filter, each RGB triplet of each pixel is regarded as a light ray hitting the surface of the corresponding filter of the Bayer filter. For instance, it has been assumed that the light hitting the red filter is a ray composed of a linear combination of three wavelengths, the RGB components. The quantity of photons passing through the filter and hitting the photodiode underneath the filter is called "quantum efficiency". The graph of the quantum efficiency as a function of the wavelength (Figure 5.7) has been used to calculate 9 coefficients. There are three filter types (red, green and blue) and three coefficients per filter. For instance, the RinR coefficient defines the quantity of red

passing through the red filter, $GinR$ and $BinR$ are the green and blue passing through the red filter, and so on for the other filters. The 9 coefficients are used to calculate the intensity of light that passes through the filters as shown in Figure 5.8. The coefficients of the *Bayer filter* module are technology-dependent since different technologies lead to different coefficients of adsorption. The *Bayer filter* input image has 1920×1080 R-G-B triplets, where R, G and B are 8 bit integer. The *Bayer filter* output image has 960×540 R-Gr-Gb-B quadruplets, where R,Gr,Gb and B are float values.

In addition, the *Bayer filter* module also inserts stuck-at defect spots randomly located such as hot pixels and dark pixels.

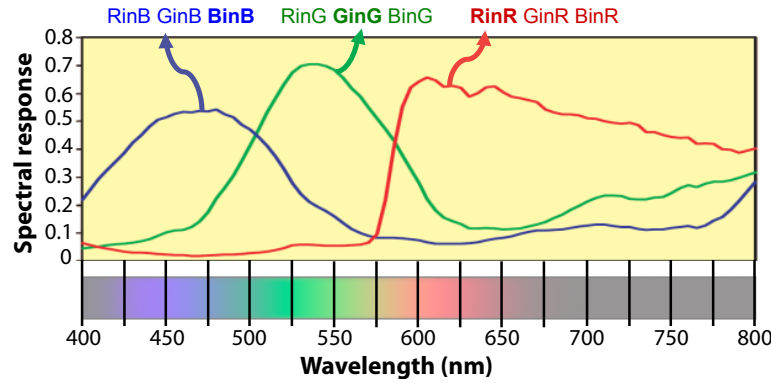


Figure 5.7.: Quantum efficiency of the CFA.

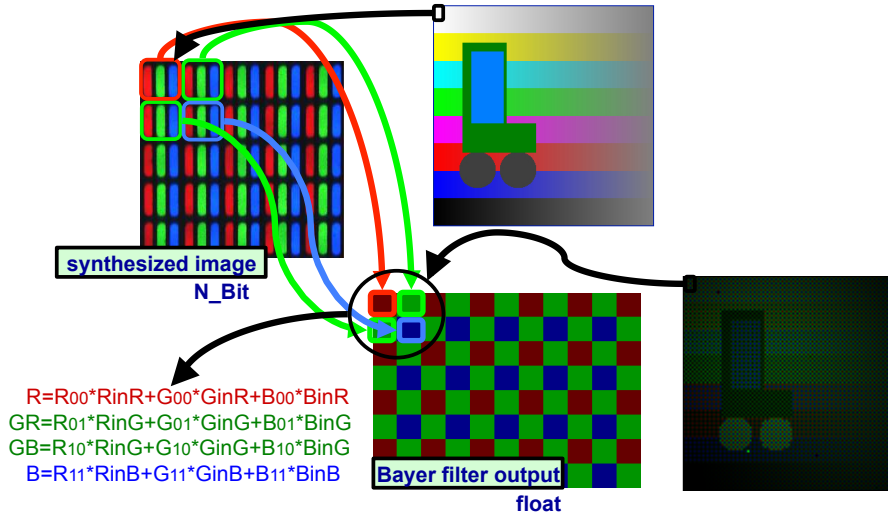


Figure 5.8.: Bayer filter modeling.

5.2.4. Video timer model

The *controller* module is fired at the frame time. It extracts the values of every R-Gr-Gb-B quadruplets of the image row after row and it sends them to the array of *pixel* modules. The *controller* also reads the integration time information in terms of *pixel time* and *row time* from

the control registers of the sensor. It calculates the actual integration time in picoseconds and sends it to the each *pixel* module of the array.

5.2.5. Pixel module

Each *pixel* module is composed of four TDF *photodiode* modules, the R-Gr-Gb-B modules. The photons that pass through the Bayer filters cause a discharge of the voltage node of the photodiode during the integration time. The discharge has been characterized by measures and results in being almost linear before the saturation occurs. An arctangent based function has been used to represent the response of the electrons collected in the photodiode's well.

Equation 5.1 describes the equations of the model, lines 12–18 of Listing 5.2 show the SystemC AMS implementation and Figure 5.9 shows the corresponding curves.

- K represents the sensitivity of the well to the light and the time. K is expressed in electrons/(sec · lx).
- SAT is the saturation value expressed in average number of electrons collected in the well (full well).
- L is the intensity of the light passing through the Bayer filter and hitting the photodiode. L is an illuminance quantity, thus it is expressed in lx. L is regarded as an independent variable.
- IT is the Integration Time regarded as the second independent variable. IT is expressed in seconds.
- IP is the Intersection Point between the constant slope line ($K \cdot L \cdot IT$) and the saturation value SAT . IP depends on L and it is expressed in seconds.
- AF is the arc tangent-based Averaging Function centered in IP . AF is a dimensionless quantity.
- SN represents the average number of electrons collected in the well. SN is a function of IT and L . SN uses AF for smoothing the conjunction between the constant slope line ($K \cdot L \cdot IT$) and the saturation value SAT . SN is calculated by adding the two terms, red ($K \cdot L \cdot IT \cdot (1-AF)$) and black ($SAT \cdot AF$) curves of Figure 5.9.
- Vx is the column voltage whereto the information flows when the readout of the electrons occurs.
- CVF is the ConVersion Factor representing the conversion of the electrons of the well into the column voltage Vx . The CVF has been experimentally characterized and takes into account the gain of the source follower as well.

Figure 5.9 shows the discharge model. On the Y-axis is SN and on the X-axis is IT . The data points to be fitted by the model are depicted using the red (x), green (+), and black (x) markers. These markers are the results of the image sensor optical characterization obtained by averaging the measures of real test chips. The markers data points are obtained for a 10 lx illuminance on the sensor plane. The SN obtained fits well the red markers (Γ equals to 150).

In order to well fit the green and black markers, thus the green and blue photodiode behaviors, a calibration of Γ and K is done.

Figure 5.9 also shows the SN discharge model for 5, 10 and 15 lx illuminance values on the sensor plane. Further details can be found in [Cenni 11b].

$$\begin{aligned}
 IP(L) &= \frac{SAT}{K \cdot L} \\
 AF(IT, L) &= \frac{1}{2} + \frac{1}{\pi} \arctan(\Gamma(IT - IP)) \\
 SN(IT, L) &= K \cdot L \cdot IT \cdot [1 - AF(IT, L)] + SAT \cdot AF(IT, L) \\
 Vx &= CVF \cdot SN
 \end{aligned} \tag{5.1}$$

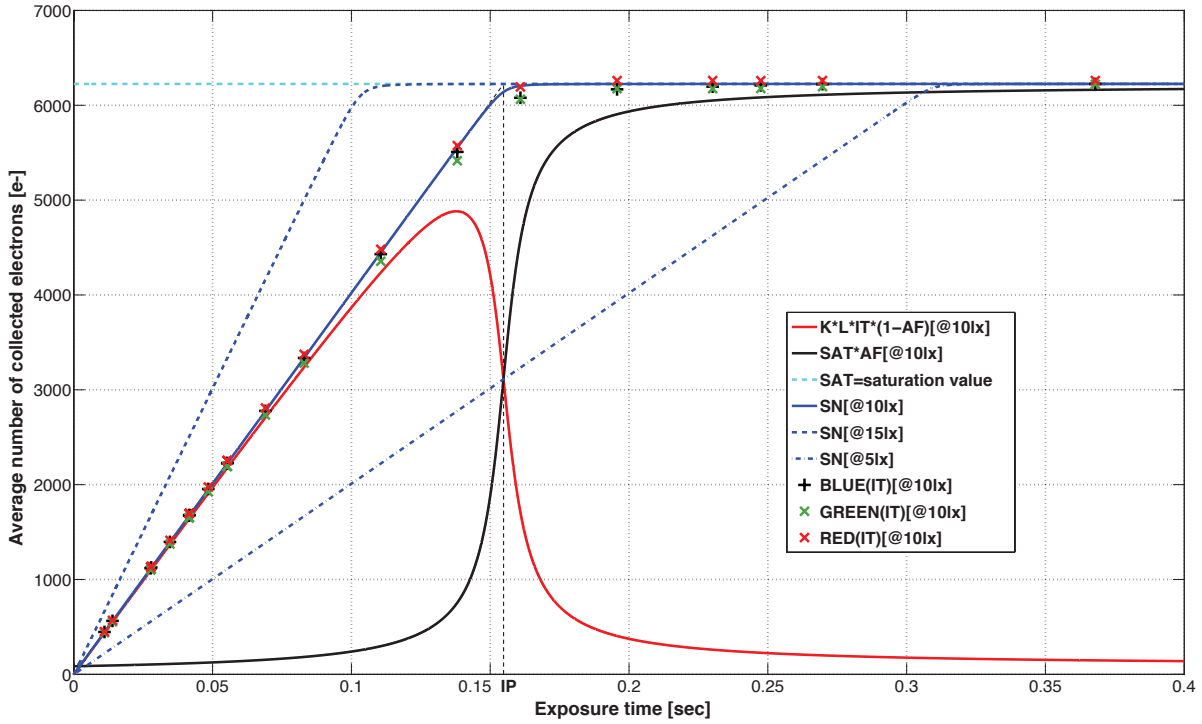


Figure 5.9.: Arc tangent based discharge model issued from characterization data.

The three dimensions graph of Figure 5.10 shows SN as function of the integration time (IT) and the illuminance (L in lux). By cross-sectioning the graph at the illuminance value of 10 lux (red points) the SN curve values shown in Figure 5.10 are observable. Further improvements will consider the possibility to define pixel with higher/lower sensitivity for modeling a fixed pattern noise (FPN).

5.2.6. ADC bank module

The ADC module is fired at the row rate, at every firing it performs a number conversions equal to the image width in pixel multiplied by 4. The conversions are from a double precision

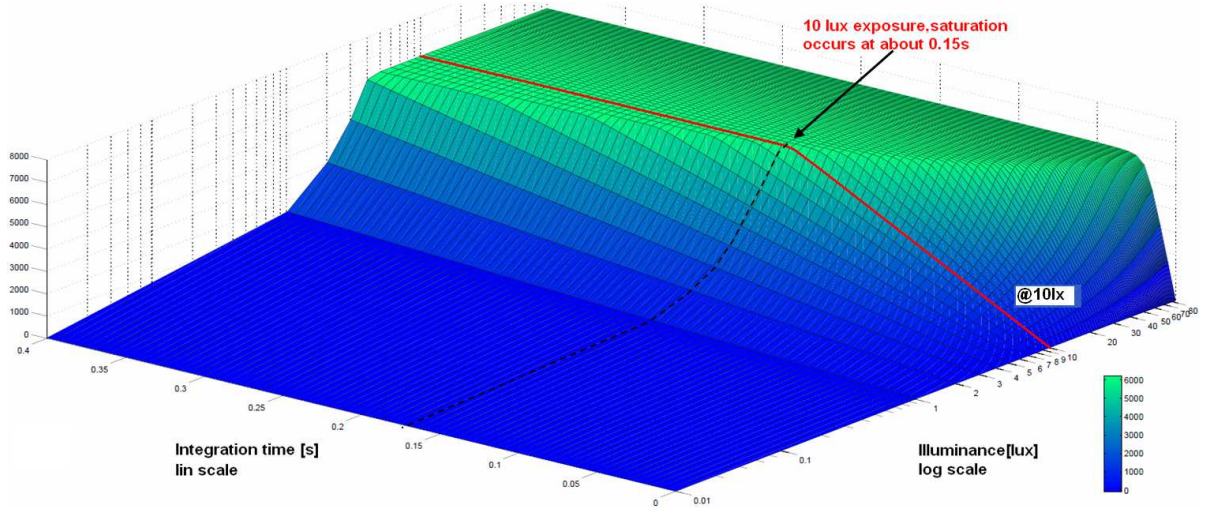


Figure 5.10.: Electrons collected as function of the integration time and the illuminance.

float number to a 16 bit integer ranged from 0 to the maximum value reachable with $nbitADC$ bits (casting to integer variable shown in Equation 5.2). The ADC is a ramp comparator based ADC, two control registers are used for setting the ADC gain, that is the ramp's minimum $V_{RMP}P_{MIN}$ and maximum $V_{RMP}P_{MAX}$ values. The conversion Equation 5.2 is shown below where $ADCCVF$ is the Conversion Factor calculated by means of both the $nbitADC$ number and the ramp minimum and maximum and VxR is the column input voltage for the red photodiode. out_R is the output value for the red photodiode and it is calculated so that $0 \leq out_R \leq 2^{nbitADC}-1$.

$$ADCCVF = \frac{2^{nbitADC}-1}{V_{RMP}P_{MAX} - V_{RMP}P_{MIN}}$$

$$out_R = (int)((VxR - V_{RMP}P_{MIN}) * ADCCVF) \quad (5.2)$$

Figure 5.11 sums up all the data conversions taking place in the SystemC AMS TDF CIS model. The four quadrants relate to the different steps. First, quadrant 1 shows the Bayer filter adsorption (for the ideal color filter array $(RinR, GinR, BinR) = (1, 0, 0)$) from the 8-bit input values (in Figure (255, g, b) as an example) to the illuminance value called L that hits the photodiode. A fictitious K compression is then performed to reduce from an illuminance range of $[0, 500K]$ to $[10, 510]$ in order not to have completely flat discharges (low-light condition) or extremely high values (high-light condition). In quadrant 2 the discharge model is selected, the choice is among the arctang-based (blue curves), the piece-wise linear (orange curves) and a hyperbolic tangent-based model (not shown in quadrant 2). The curves are not represented as a function of the time but of the light, the curves are iso-integration-time, hence for high values of the integration times the saturation value is reached at lower values of illuminance. The conversion from the sensing node SN to the column voltage Vx is then done by modeling the read-out circuitry conversion factor CVF . Even if a very low value of electrons have been collected in the well a minimum Vx (Vx_{MIN}) is anyway present on the column voltage once the read operation is enabled. The Vx voltage is then compared to the ramp shown in a stand-alone

graph in figure 5.11, the ramp is generated by a Digital-to-Analog Converter (DAC). The minimum and maximum ramp values are set by means of control registers. All these bounds are parameterized since they largely differs depending on the target CIS.

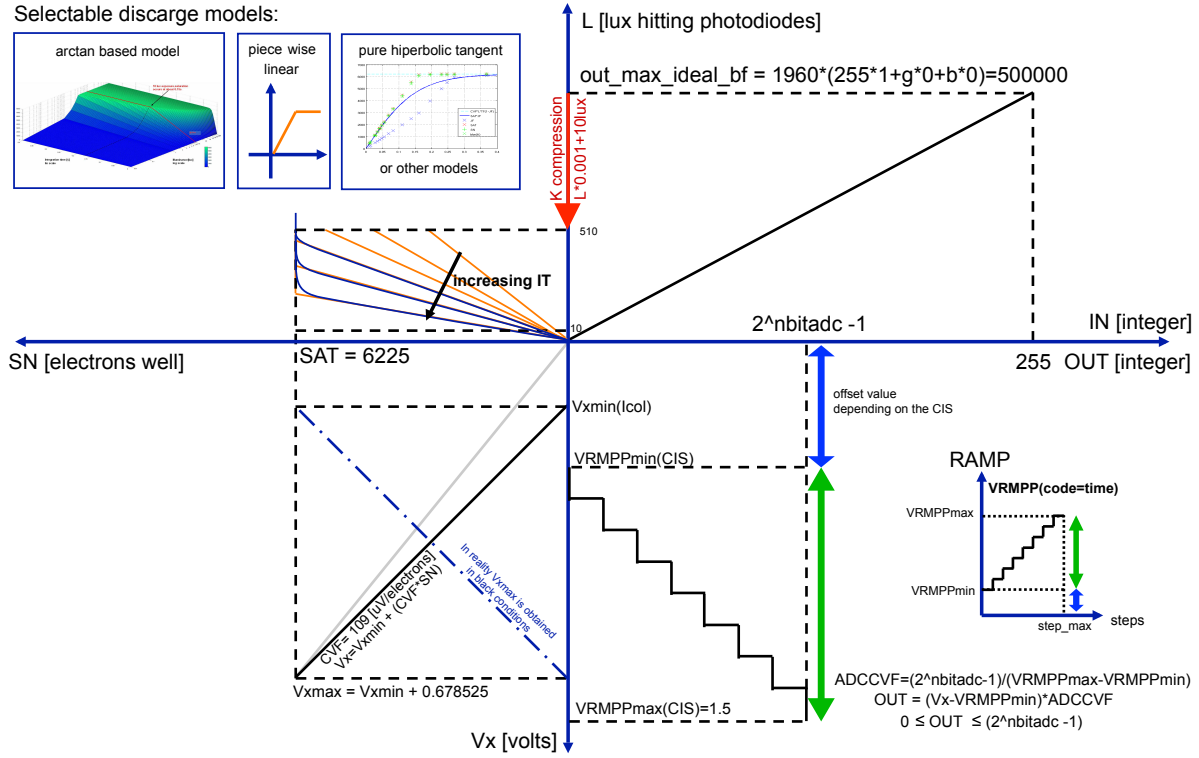


Figure 5.11.: Synoptic of the data processing within the SystemC AMS TDF CIS model.

Since the time step of the ADC TDF module is the *row time* a decimator is inserted after the *ADC* in order to provide a *sc_signal* carrying a triple pointer to the 3-dimension array which is updated at each frame time.

5.3. CIS modeling styles performance comparison

Table 5.1 evaluates the performances of the different CIS models developed, without taking into account the *tlm_isp* and *tlm_comparator* simulation time overheads, that, however, are not very influent. Since sizes are different, the simulation times for one frame have been relativized to one pixel for comparison purpose. The simulation time for one pixel is calculated by dividing the simulation time for one frame by the number of pixel. The reference for the speed-up ratio is the VHDL-AMS model. A further simplification of the TDF model described in Figure 5.4 consists in encompassing the body of 4 TDF modules (controller, pixel, ADC, decimator) into one TDF module, this leads to a change in the time steps of the model and only the *frame_time* is used for firing all the TDF modules. This simplification leads to 2 seconds of simulation time for the acquisition of a 2 megapixel frame (bottom line of Table 5.1). In table 5.1 the fastest SystemC AMS TDF model obtained reaches a tremendous speed-up factor of about 5 orders of magnitude compared to the low-level SystemC ELN-TDF model and of 7 orders of magnitude compared to the VHDL-AMS model, but the accuracy level is necessarily reduced.

A low simulation time is crucial if the CIS model is intended to run with the SystemC TLM description of the overall platform. We can observe that the ELN-TDF model is equivalent to the VHDL-AMS model in terms of abstraction level but presents a speed-up factor of about 35 times. This demonstrates the suitability of SystemC AMS as a help to RTL level design. The fastest SystemC AMS TDF model reaches a tremendous speed-up factor but the accuracy level is reduced in the sense that the simulation of the fastest SystemC AMS TDF models does not allow to trace voltage waveforms anymore for assisting the hardware/RTL designer. However, this high-level model can take into account many aspects that would not be possible to model using VHDL-AMS because of their computational weight. Such aspects are: the Bayer filter adsorption, the lens and, obviously, the interaction with a SystemC or SystemC TLM description of the digital hardware of the surrounding platform. These reasons make the model suitable for embedded software development/debugging or ISP algorithm validation.

Table 5.1.: Simulation performances.

Model	Size	Sim time for 1 frame	Time step	Sim Time for 1 pixel	Speed-up ratio	Speed-up ratio
VHDL-AMS reference	2x496	2h 30mn	Maximum time step = 5ns	Ref = 9s		1
ELN for photo diodes only+TDF	48x48	10mn	TDF time step =0.5 μ s	0.26s	1	x35
TDF array of pixels instantiated	48x48	2mn 40s	TDF time step =0.5 μ s	69ms	x3.7	x131
TDF one pixel sweeping the array	640x480	10s	50 TDF steps per pixel→ waveform	32.6 μ s	x7 949	x278 220
TDF one pixel sweeping the array	1920x1080	7s	1 TDF step per pixel→ no waveforms	3.4 μ s	x77 142	x2.7e6
↓ Enhancement to one “controller”	1920x1080	2s	1 TDF step per pixel→ no waveforms	0.96 μ s	x270 000	x9.2e6

5.4. CIS model integration in different SystemC-based platforms

The SystemC AMS model of the video sensor is integrated in different SystemC-based platforms for a demonstration of the concept and for industrial use, that is, a SystemC AMS/SystemC (TLM) joint simulation allowing to simulate the interoperability between the analog and the digital part.

Generally speaking, on the one hand, the analog part is controlled by control signals driven by the digital part. The analog behavior must change according to the changes of the digital

control signals. On the other hand, the analog part must send the relevant information to the digital part. This can be done in different ways, typically, analog signals are sampled and these samples are packaged in packets and sent to the digital part.

In the case of a video sensor platform, on the one hand the model of the sensor must be able to react according to a large number of control inputs. Notably these can be: changes of the integration time, changes to the auto-focus control signal, changes to column amplifier digital and analog gains, changes to the frame rate etc. Therefore many analog aspects have to be modeled in the SystemC AMS model. On the other hand the output of the sensor model is coarsely (frame-by-frame) or finely (row-by-row) packaged and sent to the ISP. The analog and digital parts can be modeled at different levels of details.

In the following sub-sections the interfacing will be done at different levels of abstraction aiming different target platforms. In the case of the interfacing with the OSCI TLM-2.0 a proof-of-concept platform is described in next section. Such a platform has been developed in the frame of a collaborative project called *Beyond-Design Refinement of Embedded Analogue and Mixed-Signal Systems* (Beyond-DREAMS) [Beyond-DREAMS 11]. Such a project is a CATRENE MEDEA+ European project involving both industrial and academic partners. For almost all TLM-based case-study platforms the sensor sends the entire frame as a transaction, this makes the interfacing to the TLM straightforward with no particular complexities. A finer interaction is modeled in the SystemC bit-cycle accurate platform achieved within the frame of a collaborative project called *Wireless systems And SystemC AMS Basic Infrastructure* (WASABI) [WASABI 10] of the French *National Research Agency* (ANR). Clearly, an accurate modeling requires a complex model demanding a high simulation time. In order to validate specific functionalities a solution is to reduce the complexity of the model by focusing on the target functionality only, such as the auto-exposure.

5.5. Beyond-DREAMS OSCI SystemC TLM 2.0 platform integration

Some works on the interfacing between SystemC AMS TDF and TLM 2.0 can be found in the literature and have already been introduced in section 3.6, however each application often requires different interfacing solutions.

5.5.1. SystemC TLM 2.0 proof-of-concept platform

This section deals with the structure of the SystemC TLM platform. Three TLM blocks are identified (Figure 5.12): the sensor (*tlm_sensor*), the ISP (*tlm_isp*) and the comparator (*tlm_comparator*). The SystemC AMS model of the CIS is wrapped by the *tlm_sensor*. The *tlm_isp* performs the interpolation for recovering the input image and the *tlm_comparator* performs the comparison between the synthesized input image and the output image.

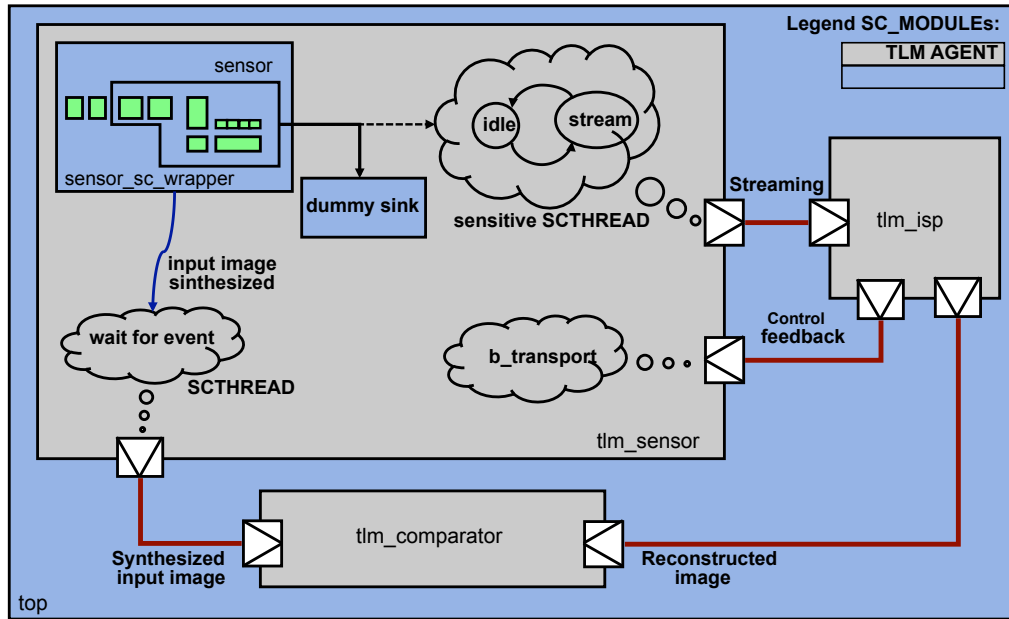


Figure 5.12.: SystemC TLM 2.0 platform: top view.

TLM Sensor

The SystemC AMS TDF chain (Figure 5.4) of the CIS model is represented by the light green blocks of Figure 5.12; it is contained by the *sensor_sc_wrapper* module. The output is an *sc_signal* (black arrowed line in Figure 5.12) driven by the TDF output converter port to the discrete-event domain of the decimator module. The output *sc_signal* carries a triple pointer to a 3-dimension array, the 3 dimensions allow to select the row, the column and the R-Gr-Gb or B values. Each time the last block of the AMS TDF chain changes the pointer value, the “TDF to SystemC” converter port makes synchronize the SystemC simulation time with the SystemC AMS one. When this happens the output *sc_signal* triggers a sensitive SC_THREAD (black dashed connection) that makes the inner 2-state-machine (cloud on top of Figure 5.12) initialize a *tlm_generic_payload* containing a pointer to the image matrix and sends the transaction to the *tlm_isp* by means of the TLM 2.0 blocking interface (delay argument not used) on the output streaming socket (red connection named “streaming”). All the transactions of this platform are done this way with the *b_transport* method. Meanwhile, once the IIB has built the input image it notifies an *sc_event* (light blue curved connection) that is detected by the SC_THREAD that sends the input synthesized image to the comparator. The comparator will wait for the reception of the reconstructed image in order to perform the comparison.

TLM ISP

The *tlm_isp* is activated by the reception of the image issued by the sensor. The ISP performs two main tasks: correction of the image and feedback the control. With respect to the correction of the image, the first step consists on correcting the Bayer filter inequality of light wavelength adsorption. In the real system this correction is done by applying coefficients stored in some sensor-specific read-only register, whose values are issued by the optical characterization and

written at the manufacturing stage. For simulation purposes, the parameters used for the correction of the ISP are the inverse of the coefficients of the Bayer filter model, as shown in top of Figure 5.13.

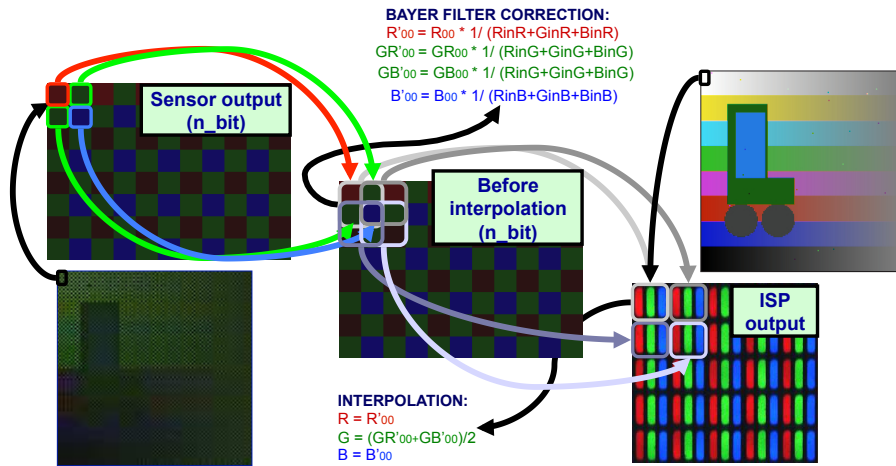


Figure 5.13.: ISP correction and interpolation.

The second step is to retrieve a raw RGB image from a Bayer patterned image. This step is called “demosaicing” in the literature and many complex algorithms can be found, the majority of them are based on an interpolation [Ramanath 02]. In our case and the demosaicing is done by means of a simple interpolation (bottom of Figure 5.13). It has been decided to keep the interpolation simple for the demonstrator, but the final aim is the integration of the AMS model with a state-of-the-art industrial ISP. With respect to the feedback of the control signals the ISP performs an estimation of the light of the reconstructed image and decides whether the integration time has to be updated or not (auto-exposure). The ISP sends the new values to the *tlm_sensor* using a blocking transport through the “control feedback” transaction link of Figure 5.12. The *tlm_sensor* receives the new values and updates the integration time *sc_signals*, which are read by the controller. The reconstructed image is then sent to the *tlm_comparator* through the “reconstructed image” transaction link (Figure 5.12).

TLM Comparator

The comparator performs a pixel-by-pixel comparison following the formula and color legend on bottom right of Figure 5.14. For each RGB triplets the Euclidean distance between the input triplet and the output one is calculated and the corresponding pixel of the comparison image is colored according to the legend on bottom right of Figure 5.14. A comparison technique is needed in order to make visible a difference between input and output otherwise imperceptible to the naked eye. This comparison equation is slightly different from the one presented in [Cenni 11b], the Euclidean norm has been chosen in order not to penalize any color value, since colors are considered as qualitative information instead of quantitative. A comparison of the different techniques is shown in [Cenni 11c].

5.5.2. SystemC AMS/SystemC OSCI TLM 2.0 platform simulation results

Figure 5.14 shows the results of a simulation of a CIS of size 400 by 400 (200x200 R-Gr-Gb-B quadruplets). The IIB synthesizes the input frames (instantaneous shots of the moving “f” top-left sequence). The “f” shifts horizontally at every frame. The Bayer patterned image output by the SystemC AMS sensor is initially darker because of a low initial integration time value, the image is visibly affected by a strong vignetting (lens effect on image), and stuck-at faulty pixels (top-center sequence). The *tlm_isp* interpolates (Figure 5.13) the images and builds the top-right sequence of Figure 5.14 without correcting nor the faulty pixels nor the vignetting. The *tlm_isp* estimates an under-exposure on the initial frames and sends a request to the *tlm_sensor* for increasing the integration time. The *tlm_comparator* compares the instantaneous shots at the CIS model input with the ERS-effect-affected output images and builds the comparison images (bottom sequence). In the first images the error is higher since a small surface is white colored, this is due to the fact that the first reconstructed images are darker. The output images gradually become brighter therefore the error decreases (wider white colored surface on the comparison images). An interesting observation is that the strong presence of the vignetting makes the output image periphery darker, the ISP tries to compensate by increasing the integration time, this results in an output image over-exposed in its center. The electronic rolling shutter effect is slightly visible in the CIS output and ISP output image sequences, it allows to see the outlined “f” as if it is leaning backward. The ERS effect is also noticeable in the comparison images where the borders of the “f” are affected by a stronger error.

Such a SystemC AMS/SystemC OSCI TLM 2.0 platform has proved the interfaceability of the SystemC AMS TDF with the OSCI TLM 2.0. The platform allows validating the concept of a SystemC AMS/TLM assisted design of a simple overall image acquisition system. The platform also constituted a delivery for the Beyond-DREAMS project and gave birth to the [Cenni 11a] and [Cenni 11c] papers.

5.6. WASABI SystemC bit-cycle accurate platform integration

The integration of the SystemC AMS based CIS model within a SystemC bit-cycle accurate level virtual platform has also been addressed. The modeling and simulation, at the component level, of an heterogeneous system composed of Wireless Sensor Network nodes is here presented. The system exhibits complex multi-discipline feedback loops that are likely to be found in many state-of-the-art applications such as cyber-physical systems. A Precollision Mitigation Braking System (PMBS) is used as a pragmatic case study to validate the whole approach.

The CIS model together with the other component models presented (60 GHz communication channel, QPSK RF transceiver, digital microcontroller, simplified car kinetic engine) are written in SystemC and SystemC AMS, and belong to five distinct yet highly interwoven disciplines: newtonian mechanics, opto-electronics, analog RF, digital and embedded software.

The work aims at simulating the complex multi-discipline feedback loop of this automotive application and the related model composability issues. Using the opto-electrical stimulus and the received RF inter-vehicle data, a car is able to exploit its environmental data to autonomously adjust its own velocity. This adjustment impacts the physical environment that

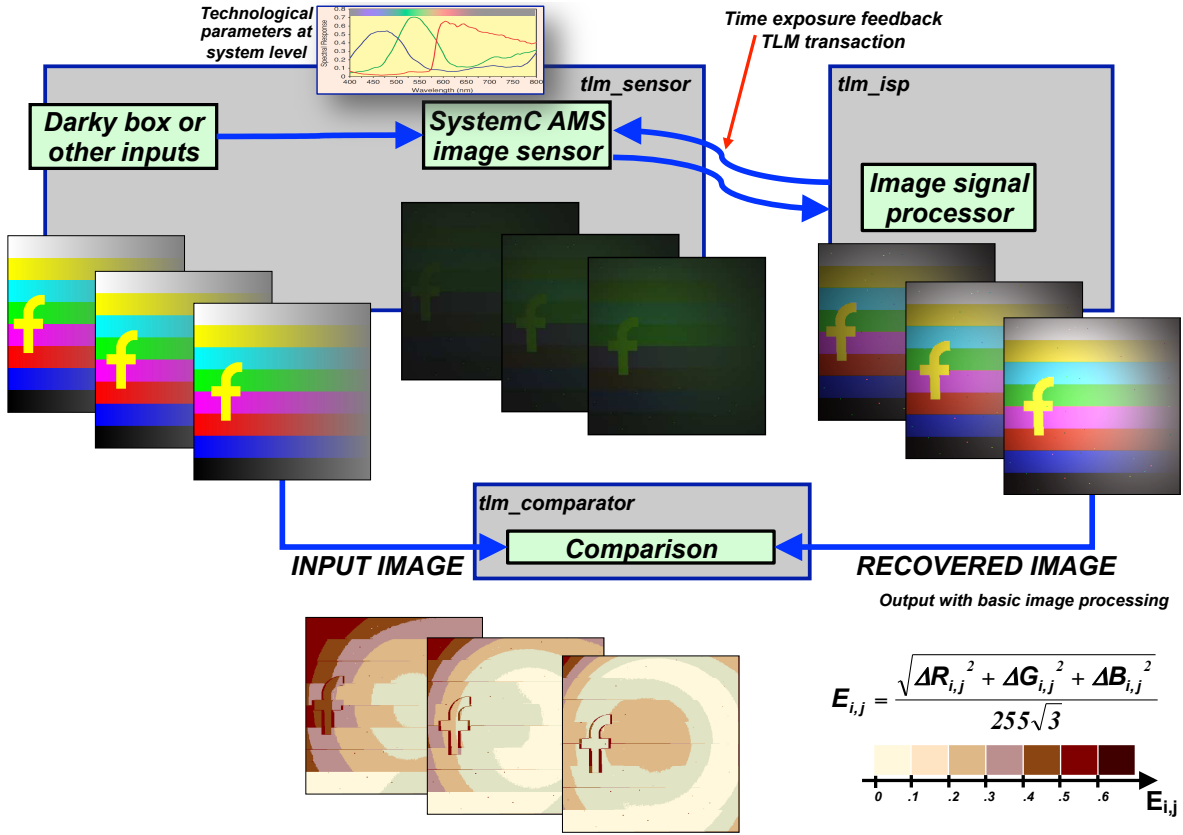


Figure 5.14.: Simulation results of the SystemC AMS/TLM platform.

in turns modifies the RF communication conditions. Results show that this holistic first-order virtual prototype can be advantageously used to jointly develop the final embedded software and to refine any of its hardware component part. In particular, cyberphysical systems featuring a tight combination/coordination between the system's computational and the physical elements exhibit by nature complex feedback loops that can yet hardly be captured in currently available design environments, especially when it comes to integrate software in these loops.

The sub-models have been developed especially for the application (complete 60 GHz QPSK baseband RF transmission scheme) or reused from previous designs (digital, embedded software templated microkernel).

The presented heterogeneous system is a direct application of Inter-Vehicle Communication (IVC) principles for road safety by dissemination of warning information [Luo 05, Lambert 10]. In the automotive industry, such a system is called a Precollision Mitigation Braking System (PMBS) [Lambert 10].

Figure 5.15(a) depicts the frame of this application, the PMBS is modeled and simulated at the component level and actually corresponds to a wireless sensor network where each node represents a car i moving, at night, at velocity v_i and acceleration a_i on a flat rectilinear road, as shown in Figure. This component-based modeling approach allows for the refinement at any time of any of the system's constituent parts. d_{ij} and v_{ij} respectively represent the distance and relative velocity between car i and j . The vehicles are all equivalent and are not allowed to

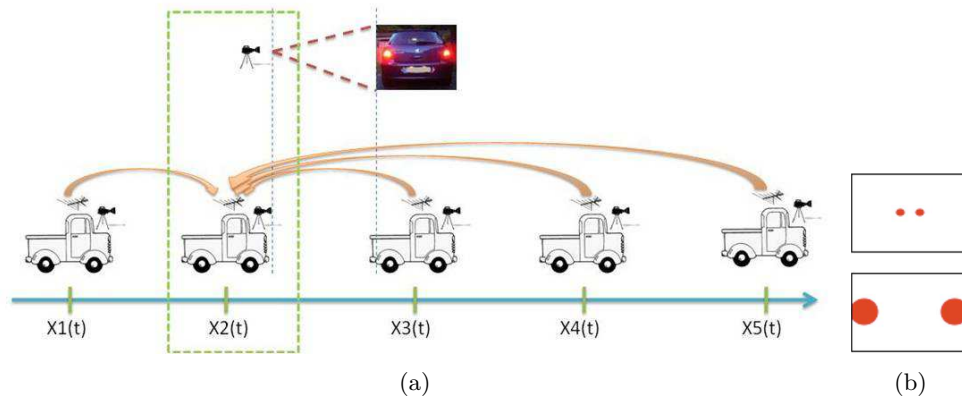


Figure 5.15.: The modeled PMBS system, user's view.

overtake each other. Each vehicle is equipped with two red tail lamps (turned on), a 320x200 pixels color video sensor at the front, and a 60 GHz RF transceiver for inter-vehicle propagation of meaningful kinetic values for each car. Periodically, each car sends its kinetic information and distance estimation to all other cars, allowing them to locally rebuild a quite accurate representation of the traffic. The video sensor captures the two red light cones emitted by the preceding car and generates a digital color bitmap image in a memory buffer. Figure 5.15(b) shows video sensor images corresponding to the far (image on top) and near (image on bottom) vehicles situations. The image buffer is stored in a 32-bit micro-controller memory that can be accessed directly by the embedded software in order to compute an estimate of the distance with the upstream vehicle. Considering several successive sensor frames, it is even possible to calculate an estimate of the relative speed with the preceding car. To prevent several vehicles to emit RF data at the same time, a simple Time-Division Multiple Access (TDMA) scheme is used. A given car is allowed to send information if and only if it has gained the TDMA token. A special RF token is sent periodically to resynchronize the RF receivers.

5.6.1. Modeling the whole system

The global PMBS system, detailed in Figure 5.16, has been completely modeled in SystemC and SystemC AMS. The model is composed of 4 main parts and addresses 5 tightly coupled disciplines, each characterized by its color in the figure: newtonian mechanics (Yellow), optoelectronics (Red), analog RF (Green), digital electronics (Blue) and embedded software (Mauve). The fading colors indicate the points of interest, the interdisciplinary interfaces. The top part (Part 1, Physical environment) of the figure represents the physical environment, i.e. physical reality. It is modeled in SystemC AMS TDF MoC, receives the real kinetic data from the kinetic engine of each car, and is responsible for generating the image stimuli for each car, for gathering the real kinetic data of all cars and for propagating these data to the RF communication channel. The bottom part (Part 2, Communication channel) represents the 60 GHz communication channel. The channel, also modeled in SystemC AMS TDF, receives as inputs the RF transmitter outputs of each car as well as all the kinetic data (inter-vehicle distances and relative velocities) that considerably impact the channel behavior. The communication channel is responsible for generating the RF output data that is propagated to each car RF receiver. The middle part (Part 3, Car wireless sensor nodes) of the figure corresponds to the Wireless Sensor

Nodes representing the cars. A car is mainly characterized by its digital control that runs the embedded software and reacts to an external video sensor event (a new video frame has been received) emitted by the video sensor, to an external RF event (an RF message emitted by another car has been received) emitted by the RF transceiver, and to an internal timer interrupt event (used to compute the owner of the TDMA token). The right part (Part 4, Online display) takes in charge the online (during simulation) display in a graphical window of the real and approximated kinetic data. The display module is itself composed of two subparts: a SystemC AMS compatible subpart capable of hardware and software introspection (used to obtain the values of interest from any software or hardware simulator resource) and a graphical subpart based on the portable Simple DirectMedia Layer (SDL) graphic library that generates a vivid bitmapped representation of the simulation. Figure 5.16 clearly exhibits the complex multi-discipline feedback loop. Using the opto-electrical stimulus (i.e. Reading Video Sensor and calculating the interpixel distance) and the received RF data (i.e. Reading RF transceiver receiver), a car is able to adjust its own velocity (i.e. Writing to kinetic engine). This adjustment impacts the physical environment (i.e. Write to Environment) that in turns modifies the RF communication conditions (i.e. Write to Communication channel). It is worth noticing that the environment and RF parts are tightly connected through the kinetic data and thus considered altogether from a simulation viewpoint. After elaboration of the whole system model (just before the simulation starts), and by means of a configuration file, each vehicle i is assigned its initial position on the road x_i , velocity v_i and acceleration a_i . RF can be activated or not, allowing many different simulation scenarios with open or closed feedback loops. Details on the modeling of each component of the system can be found in [Lévêque 12], next sub-section will show how the analog and digital parts are interfaced.

5.6.2. Modeling of the SystemC / SystemC AMS interfacing

The interfacing is done by means of TDF/DE converter ports which read/write from/to discrete event SystemC signals. For sending the Bayer patterned image captured by the CIS model to the SystemC model of a 32-bit MIPS32 processor the data are packed and sent according to the VCI protocol used as a standard communication protocol in the development of the SoCLib model database¹. Figure 5.17 shows how the interfacing is achieved, the image from the CIS model is retrieved in an `sc_signal` and the integration time control signals are sent to the SystemC AMS TDF CIS model by means of `sc_signals`. Each components has a VCI interfacing module implementing an FSM machine allowing to read/write on the interconnect module. The interconnect module, in turn, handles the calls and dispatch them to the good candidate, which is another VCI interfaced component. The functioning of FSM-based VCI interface is controlled by a clock signal to which each VCI interface is connected. The sensor module itself dialogues with its interface by means of `sc_signals`: the input image (issued by the environment which loads the right images depending on the distance), the CIS control signals (e.g., integration time) and the output image.

The presented application virtual prototype contains a embedded software, not so complex for the current platform size (the `main.c` code is shown in Annex A.3). However the methodology

¹SoCLib is an open platform for virtual prototyping of multi-processors system on chip (MP-SoC). The project started as an ANR-funded project and it is now maintained at Lip6 (Laboratoire d'Informatique Paris 6). The communication protocol used in the model library is the VCI (Virtual Component Interface) standard sponsored by VSIA. In the SoCLib context APIs have been developed for supporting the VCI protocol.

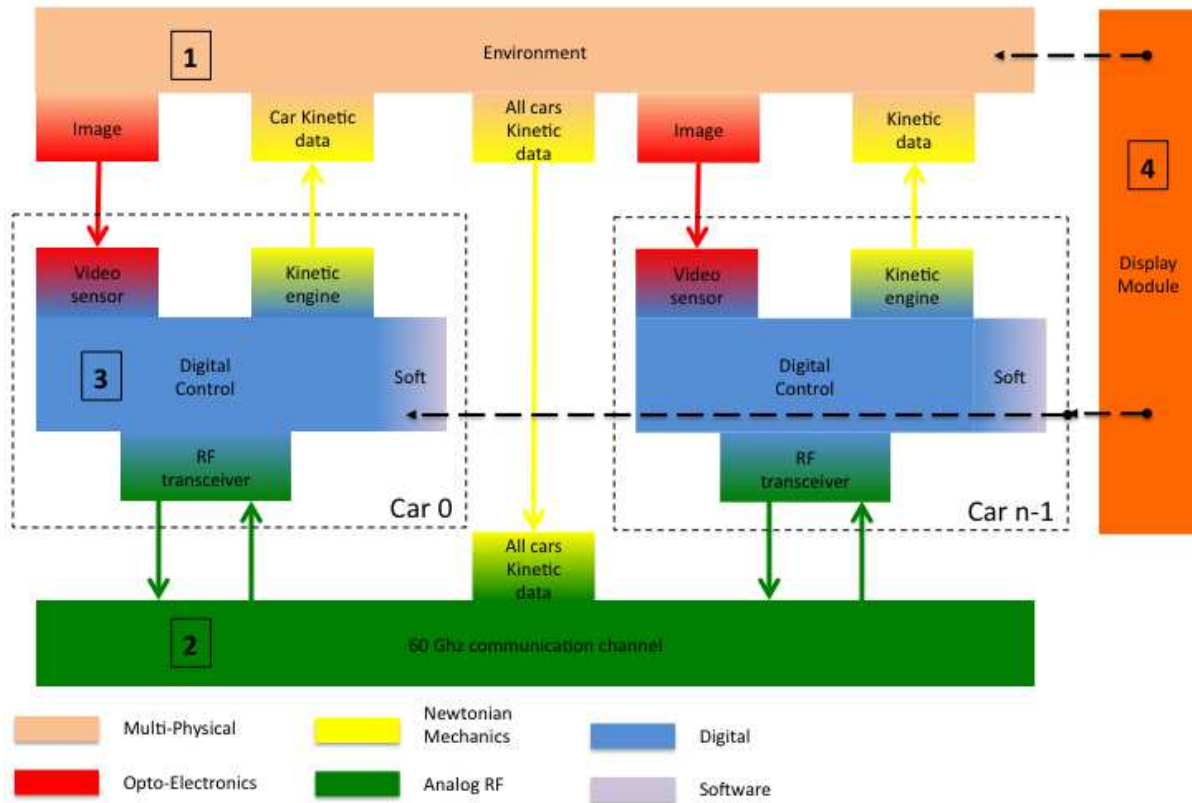


Figure 5.16.: The modeled PMBS system.

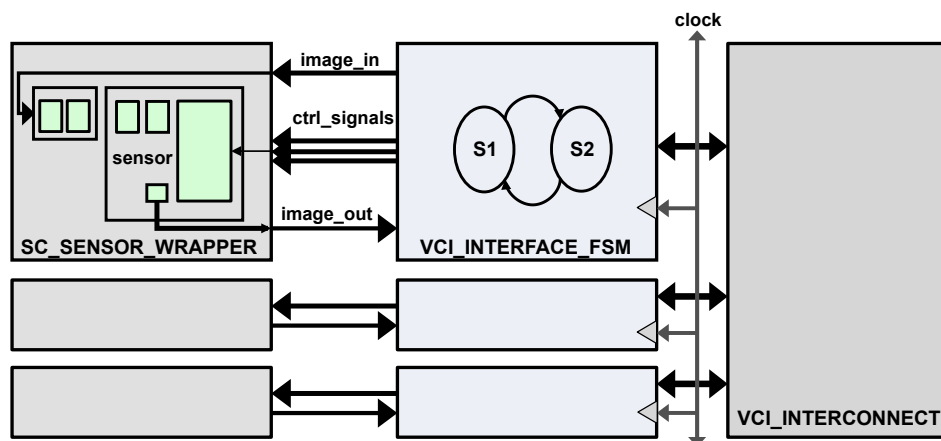


Figure 5.17.: SystemC / SystemC AMS interfacing through VCI interconnects.

is proved to be efficient for an early validation of the application via early embedded software development/debug possibilities.

Such a SystemC / SystemC AMS platform has been developed thanks to the partners of the consortium of the French *National Research Agency (ANR) Wireless systems And SystemC*

AMS Basic Infrastructure (WASABI) project [WASABI 10]. The author of this thesis, in quality of the developer of the CIS model, has been actively working at the interfacing to the SystemC cycle-bit accurate platform. This work allowed validating the complex multi-discipline feedback loop of this automotive application and gave birth to the [Lévêque 12] paper.

5.7. STMicroelectronics CATSEYE SystemC TLM platform integration

One industrial application where the SystemC AMS based modeling of the CIS has been proved to be a promising methodology is an STMicroelectronics' SoC for energy-saving purpose. This SoC is mounted on Personal Computers (PCs) or TV screens exactly where the web-cam normally is. The SoC embeds a small CIS aimed at face-detection and ambient-light sensing. The CIS is a grey-level sensor and its size is of 600 by 200 pixels, therefore no Color Filter Array (CFA) is present in the sensor. An ISP is present in the SoC and a first processing is done with the purpose to de-noise the image and improve its quality for running the face detection algorithm on a 32bit micro-processor. On bottom of the sensitive array a group of rows is devoted to the other function of the SoC, that is the ambient light sensing. The aim of the product is to be able to react by switching on/off the screen whether nobody is standing in front of it (absence), or, in generic terms if nobody is watching it (closed eyes), further, other actions will be taken depending on the level of ambient light sensed. Figure 5.18 shows the block that composes the processing steps: for the overall captured image (flow on top) a calibration of the dark signals is done, then the image is descrambled (for recomposing the entire image row after row), an anti-vignetting processing is performed, then a gamma compression from 10 to 8 bits and an image cropping. On the bottom of the image the rows read from the sensitive array are used to evaluate the ambient light and used to perform some adjustment on the integration time. The tasks allocated to the embedded firmware of the micro-processor are depicted by the blue blocks, the face detection algorithms are implemented on it. The output images are not intended to be retrieved from the SoC but some decisions have to be taken.

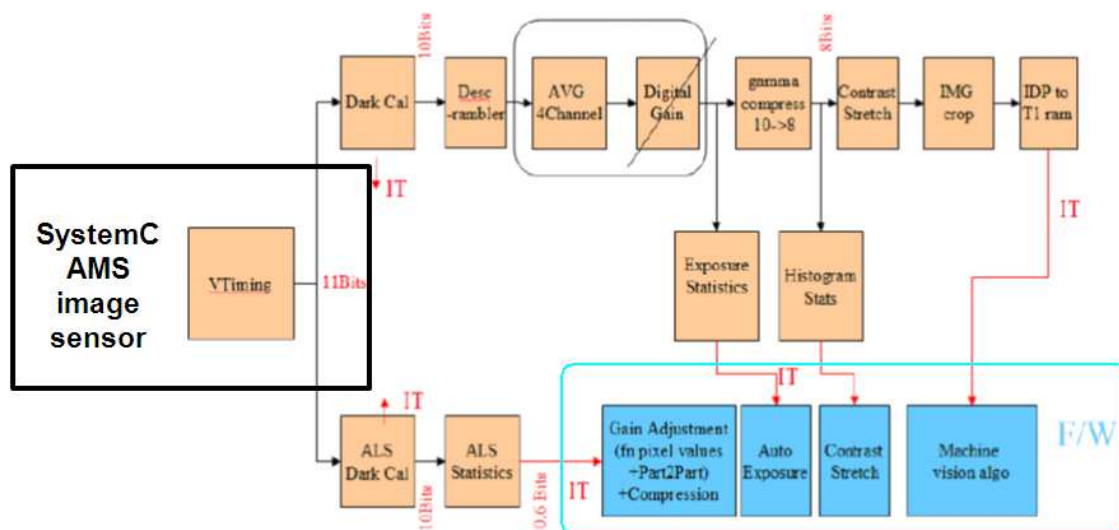


Figure 5.18.: Overview of the CATSEYE image processing and embedded firmware.

Figure 5.19 shows the interfacing between the SystemC AMS CIS model and the SystemC-TLM virtual platform. The clouds are methods of the class where they appear in. The white TLM sockets are sockets related to the STMicroelectronics' TAC_TLM protocol except for the *interrupt* white socket that is related to the WIRE_TLM protocol. The `vtiming` module provides three solutions in order to send an image stream to the following platform. On top left the `file_reader` module allows to load predefined generated image files or images pre-acquired from a previous run of the real sensor test chip. On bottom right the four white blocks together with 2 clouds are used for connecting the SystemC-TLM virtual platform on the PC to the real sensor test chip via PCI-express ports.

5.7.1. Digital to analog control

Three banks of registers are instantiated in the `vtiming` class (top right of Figure 5.19). Read or write accesses to the registers of the banks of registers take place via TAC_TLM transactions and different methods are triggered once read or/write operations occur, these are the so called *side effect methods*. The `sensor_tlm_wrapper` module encompasses the SystemC AMS CIS model (green blocks on the image) that, in turn, is contained by the `sensor_sc_wrapper` module. The integration time is handled by the CIS by using two registers for the tuning, a coarse tuning and a fine tuning. The `vtiming_coarse/fine_integtime` *side effect methods* are configured for being triggered on a write access, the latter is ordered by the rest of the platform. The two methods redirect to two other methods contained in the `sensor_sc_wrapper` module. These two methods update the values of the `sc_signals` that carry the information of the last modified values of the integration time. Changes to these signals will then make synchronizing the SystemC AMS time with the SystemC one at the next firing of the TDF module. The SystemC AMS CIS model will take into account the new integration time values from the next image acquisition as it is really occur in the I2C (Inter-Integrated Circuit) connection when accessing the CIS register banks in order to modify the parameters of the image acquisition.

5.7.2. Analog to digital information passing

The process `compute()` of the `sensor_tlm_wrapper` class is sensitive to the output `sc_signal`, once the latter is updated (be aware that the content of the signal has to change in order for it to trigger the method) a packet specific to the streaming protocol is prepared and sent in two phases via the `writer_port` streaming socket to the `vtiming::compute()` process. First the *extension* (additional information for specifying the conditions/parameters of the following data) is sent then the `compute()` process blocks and wait for the receiver to acknowledge the reception of the *extension*, subsequently the payload referring to the acquired image is sent and the process blocks until the acknowledgement. The same procedure takes place for the transaction across the `idp_out` streaming socket for delivering the image to the following ISP blocks shown in Figure 5.18.

Such an integration has been done as a proof of the CIS model reusability. The CIS model was not initially developed for fitting the specifications of the CIS employed in the CATSEYE product. However, since the model is parameterizable its adaptation is fast and straightforward. The STMicroelectronics *Imaging* is currently performing analysis and validations of the platform itself.

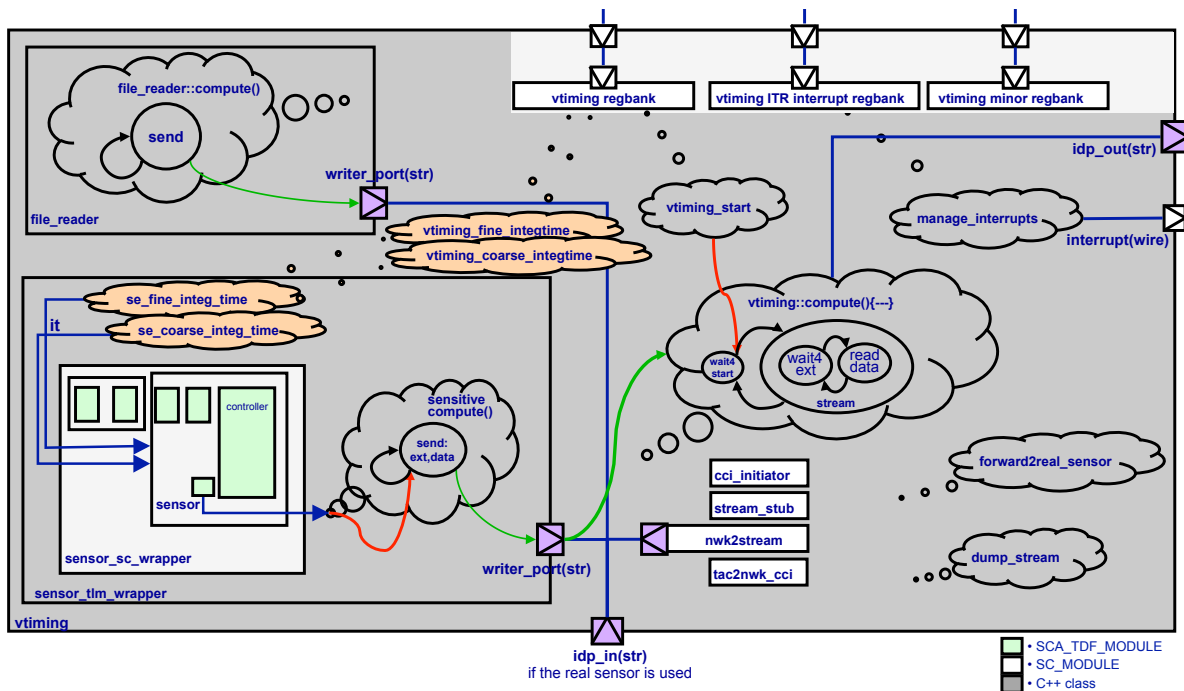


Figure 5.19.: SystemC AMS/ TLM interfacing inside the CATSEYE virtual platform.

5.8. ST-Ericsson SystemC TLM mobile platform integration

The ST-Ericsson application is a SystemC-TLM virtual platform for mobile applications, typically high-performance smartphone and tablets combining a state of the art application processor with a HSPA+ (High-Speed Packet Access) modem. The platform incorporates a SMP (Symmetric Multi-Processing) dual-core technology. The platform is sketched in Figure 5.20, its high complexity justifies the effort demanded by the virtual prototyping in order to early debug the embedded software and to enable the reuse of IPs. The analog parts have to be virtually prototyped as well, and in these aspects locates the incoming need for a SystemC AMS based modeling of sensors and actuators. In this application the sensor is much more complex in the sense that it has bigger size, it is a color CIS, auto-focus capabilities etc. Not to mention the fact that at least two image sensors are expected to be part of the platform (rear located, bigger in size, up to 20MPix and the front-one for video-calls 5MPix from the specifications).

In the specific, Figure 5.21 shows the interfacing between the SystemC AMS CIS model and the SystemC-TLM virtual platform. The interfacing takes place roughly as in the previous CATSEYE application. Some differences can be listed: some class names are changed, only one register bank is now present. Additionally, only one *side effect method* named `ams_side_effect` calls two inner methods for updating the integration time coarse and fine `sc_signals`. The thunder-shaped arrows on the extreme left shows the parameters to be passed to the constructors. The particularity in this application is that a high number of functions performed by a more complex state-of-the-art ISP are directly integrated in the `tac_smia_sensor::compute()` process. To mention some of them: image crop, pixel sub-sampling, mirroring, image scaling, digital gain, bit compression etc. Additional information (called Intelligent Status Lines (ISL)

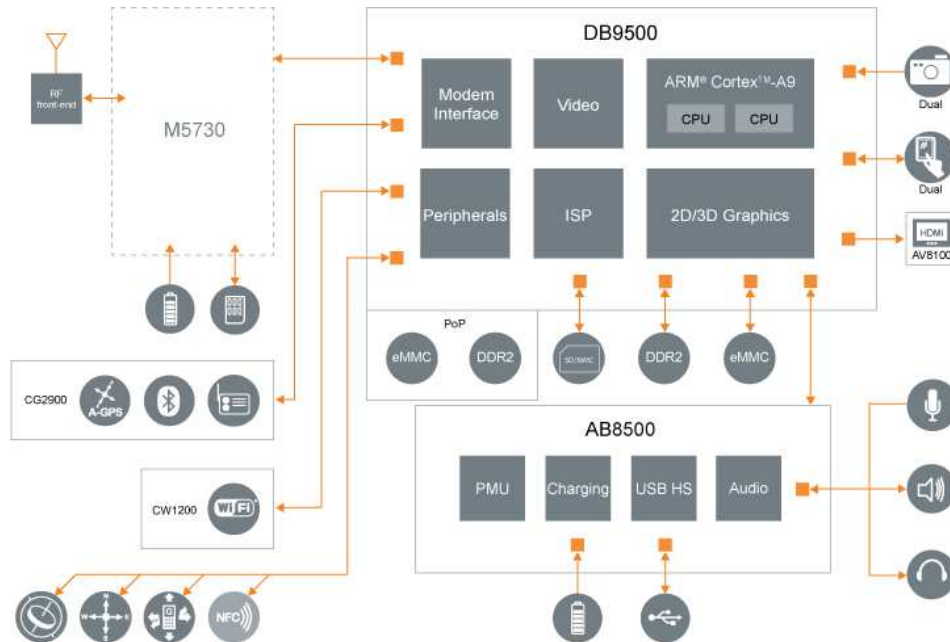


Figure 5.20.: Overview of ST-Ericsson's platform for mobile applications.

in Figure) are then added for tagging the pure image data with the settings of the image acquisition (integration time values, analog gains etc.) that have been used for acquiring that specific frame. The image is then sent according to the SMIA (Standard Mobile Imaging Architecture) protocol for imaging devices.

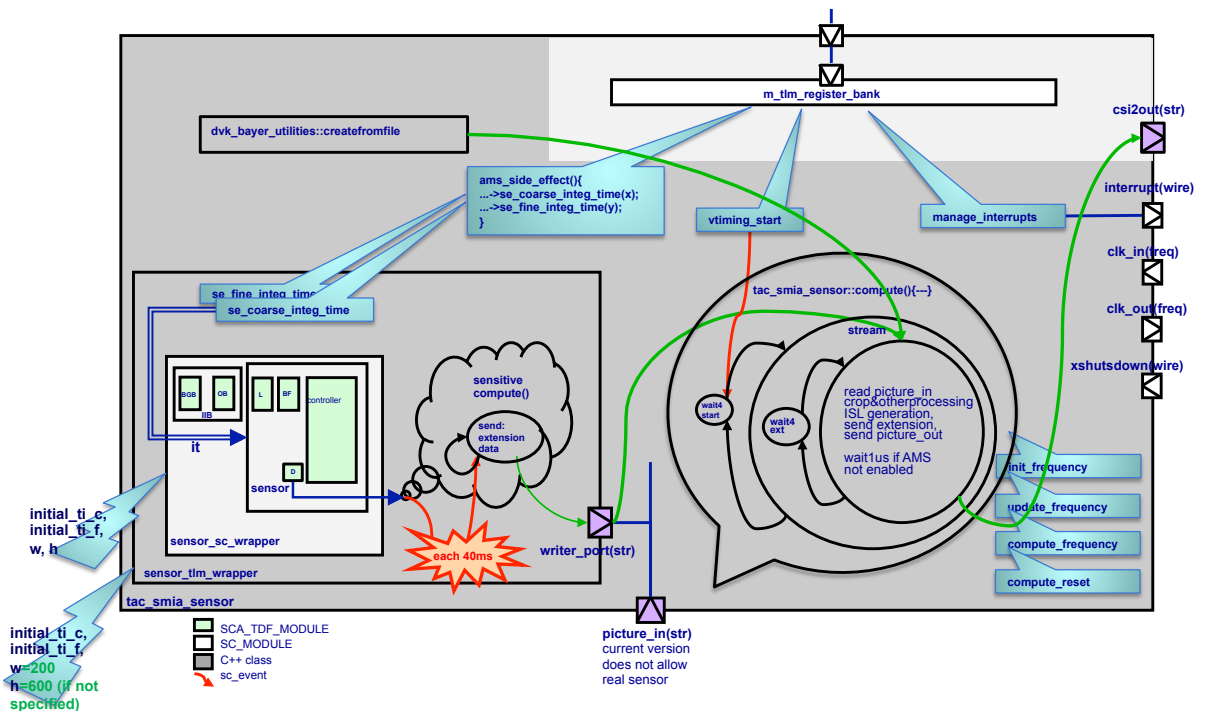


Figure 5.21.: SystemC AMS/TLM interfacing with ST-Ericsson's virtual platform.

Figure 5.22 shows the simulation results of the image acquisition chain of the ST-Ericsson's virtual platform. *Image 1* is the input image built by the *darky-box* model and provided to the SytemC-AMS CIS model. The image is captured by the CIS model by applying the lens shading effect, the CFA adsorption, stuck-at hot spot pixels, photodiode discharge, analog-to-digital conversion. The Bayer patterned output of the CIS model is *Image 2*, it is visibly affected by vignetting and stuck-at black and white pixels. It should be noted that, for a matter of clarity, *Image 2* is a Bayer patterned image colored according to a grey-scale, while in Figures 5.8, 5.13 and 5.14 the pixels of the Bayer patterned images were colored according to the Bayer pattern itself, i.e. R-Gr-Gb-B, this gave them a green aspect. The image is then sent to the ISP with its different steps. *Image 3* is the output of the *de-noising* stage, it applies statistical calculations for correcting different known CIS noise sources such as Fixed Pattern Noises (FPNs) (Photo-Response Non-Uniformity (PRNU), Dark-Signal Non-Uniformity (DSNU)). White-colored hot pixels have visibly been corrected by this stage but black hot pixels are still present. *Image 4* is the output of the *lens shading correction* stage, the image is brighter than *Image 3* in its borders and corners. *Image 5* is the output of the *demosaicking* stage, the image is converted from a Bayer-patterned to an RGB image. *Image 6* is the output of the *white balance* stage, the overall green aspect of the previous image is reduced. Finally, *Image 7* is the output of the *color gain* stage, it consists of an improvement of both the color contrast and rendering.

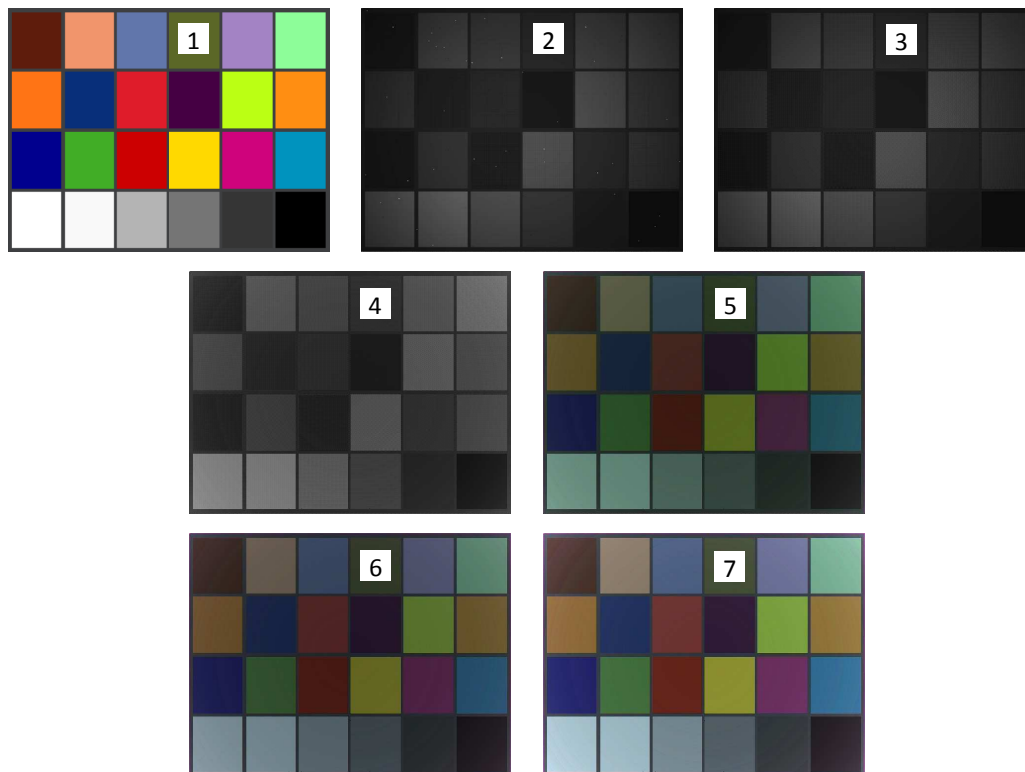


Figure 5.22.: Results of the image processing from the acquisition (sensor input) through the ISP processing.

The integration of the CIS model in this platform (together with the CATSEYE platform of section 5.7) showed that no issues have been encountered when interfacing the SystemC AMS-based model with the STMicroelectronics' proprietary TLM protocols. ST-Ericsson is currently validating the SystemC TLM-based virtual platform of their SoC for mobile application (Figure 5.20). First embedded software validations can already be done thanks to such virtual platform.

5.9. Conclusion and future works

We have demonstrated the suitability of SystemC AMS for image sensor modeling. Different SystemC AMS MoCs were used for modeling different abstraction levels resulting in models showing different accuracy levels. The performances of the respective models have been compared. The fastest model of the CIS has been developed by means of the TDF MoC. The interoperability between analog and mixed-signal components described using SystemC AMS and their associated digital environment has been proved by integrating the model of a CMOS image sensor in different digital virtual platforms modeled with SystemC and SystemC-TLM. The low execution time of the SystemC AMS TDF model, in the same order of magnitude of the SystemC TLM software prototype, allows performing validation and debug of the whole system before the hardware availability. Further analog and mixed-signal modeling improvements will concern with the optical system composed of the lens controlled by a voice coil motor (VCM) driver for the auto focus capability. A further solution will be to supply the CIS model input image by means of the rendering of a 3D scene in order to reproduce movie effects (camera or subject in motion).

No blocking points have been reported concerning the interfacing between the AMS sensor and the proprietary STMicroelectronics TLM protocols. The first user of the developed CIS model is ST-Ericsson for their SoC for mobile application, as described in section 5.8 the model has been successfully integrated and first validations are ongoing in the matter of the Auto-Exposition (AE) feature. The next-step aim is to be able to validate the stability and efficiency in terms of convergence speed of two other well known control loops, the Auto-Focus (AF) and the Automatic White Balancing (AWB).

The validation of the SystemC AMS CIS model in comparison to a CIS test chip has not been carried out because of two main reasons.

First, a *complex inter-division framework*, the work inside STMicroelectronics has been done transversally with different divisions, the team that I have been integrating is aimed at providing a centralized support for the behavioral modeling-based design flow of AMS devices, formerly VHDL-AMS based and now moving upward at system level. In order to develop a CIS model, I have been collaborating with the *Imaging* division that is in charge of what concerns with the design and manufacturing of image sensing systems, among others. In order to collect the model specifications I have been collaborating with the customer/user of the SystemC-AMS based CIS model, that is the ST-Ericsson's division in charge of the system level design of the image acquisition & post-processing frontend of their SoC. Discussions have been conducted for agreeing on the specifications of the model performances in terms of simulation time and behaviors to be reproduced for validating the compliance to specific performances under test. From a technical viewpoint I have been collaborating with the *System Platform Group*, this group has a centralized competence on the digital-only SystemC-TLM-based virtual

prototyping for system architecture exploration and early embedded software development. The SystemC AMS CIS model has been developed and the interfacing to the SystemC-TLM platform has been carried out by myself. A first validation of the integration has then been done by the SystemC-TLM group, the latter acts as methodology provider and support to the final user/customer ST-Ericsson.

Second, a comparison of the model behavior with a test board for the image sensor chip demands to set up exactly the same benchmark environmental conditions. Such an environment setup implies both the configuration of the control registers (fast for the model but time-consuming for the real CIS) and the calibration of the dark box light conditions for matching the model stimuli.

Despite this, the model has been built by means of the measure results hence a very high confidence is put on it and users are satisfied with the trade-off between accuracy and simulation speed. Our work has demonstrated the applicability at the industrial scale of an AMS system design methodology based on a SystemC AMS virtual prototyping.

Chapter 6.

Conclusions and perspectives

Conclusions

In my thesis I have been working on the modeling of heterogeneous multiphysics and mixed-signal systems at a behavioral level.

Such a behavioral modeling of AMS devices is intended to provide many enhancements to the design flow of a complex System on Chip containing sensors and actuators, hence a significant analog part.

By composing the behavioral models of single IPs into a virtual platform of the overall system within a unified C++-based framework it is possible to perform a full verification of the system. More precisely many benefits are provided such as architecture exploration, performance estimation, validation of reused parts, verification of the interfaces between RF, analog and digital parts, early verification of the embedded software development together with its eventual debugging, verification of the interoperability with other systems, and assessment of the impact of the future working environment and the silicon technologies used to realize the system.

In addition, validation through virtual prototypes will test the fit of not-yet-available IPs inside the target System on Chip, this would obviate the dependency of the design kick-off date on the availability of the IP in order to anticipate the design process of future products therefore, why not, anticipate technology advancements.

With respect to the modeling techniques, in this thesis three flows for building models and for reducing the order of analytical models of heterogeneous mixed-signal are presented.

First, a behavioral modeling technique from schematic netlist entry description, has been described and automatized from extracting desired information under the guise of state space equations.

Second, techniques based on analytical fittings of frequency response are explored for either reducing the model order or for identifying analytical/simulatable models starting from analysis carried out with other application-specific CAD tools.

Finally, system identification techniques are studied for the extraction of black-box models from empirically obtained data, either from simulations of accurate models or from measure data. A proof-of-concept library implemented using SystemC AMS shows the applicability of the methodology.

The behavioral modeling techniques have been exploited for the design and for the test/control in two major case studies. First, the design of the frontend of a Surface Acoustic Wave-based chemical sensor exploits the approximation-based modeling techniques for performing simulations down to the layout view. Second, the system identification-based modeling techniques are applied to a Low Noise Amplifier-based (LNA) case study for an automated control of the LNA itself through a performance estimation from the parameters of the model. This feedback control loop is aimed at the power consumption minimization of an RF transceiver.

Although the tools developed for behavioral model extraction are mostly based on the AMS extension of the SystemC kernel, the methodology can be applied to other Analog Hardware Description Languages (AHDL) such as Verilog-A and VHDL-AMS.

Subsequently, the industrial nature of the Ph.D. led to concentrate the modeling efforts on a CMOS image sensor (CIS) case-study. The image sensor model is abstracted at different levels using different SystemC AMS Models of Computation showing impressive simulation time performances.

The integration of such a SystemC AMS-based CIS model into SystemC-based image acquisition virtual platforms has been done for different case-studies. First, the integration into a SystemC TLM 2.0 proof-of-concept platform has been shown. Second, the SystemC AMS CIS model has been simulated inside a SystemC-based bit-cycle accurate model of WSN nodes for an automotive pre-collision mitigation application. Subsequently, the model has been integrated in two different industrial applications. In both cases the analog/digital interfacing between the SystemC AMS MoCs and the STMicroelectronics' proprietary SystemC-TLM protocols has been validated.

ST-Ericsson is currently performing validations of the TLM virtual platform including the CIS model. Both the analog and digital part are described using the SystemC framework, respectively AMS and TLM. The SystemC AMS based design and verification methodology is now being adopted by ST-Ericsson in the case of an AMS SoC for mobile applications. The applicability of the methodology to the industrial design flow is proved and will enable the early embedded software development and debug for AMS SoCs.

Considerations and perspectives

The birth of analog hardware description languages (VHDL-AMS and Verilog-AMS) has enabled the validation of AMS/digital systems with small quantities of embedded SW. Driven by the ever-growing complexity of Systems on Chip, modeling tools/languages are requested to abstract the view at a higher system-level.

For the digital side the SystemC framework is covering this need at system level with SystemC-TLM. The AMS extensions to SystemC are now offering the generic high-level modeling capabilities for covering such needs concerning to the AMS parts. Further models of computation can always be plugged on the SystemC architecture but a standardized interface definition for plugging other models of computation must be provided.

The team leader in charge of the SystemC-TLM-based virtual prototyping at ST-Ericsson once stated what follows: "The software development/debug normally starts once hardware prototypes are available and only 2 months are allocated for the SW debug before the product hits the market. The TLM-based methodology makes available virtual prototypes of the platform about 9 months before hardware prototypes. These virtual prototypes are suitable for

developing and debugging the embedded software. Such a development/debug can therefore start before the availability of the hardware prototype. This ensures a higher software reliability, hence a lower error proneness, and up to 2 months of time-to-market gain in the optimistic ideal case” (see Figure 6.1).

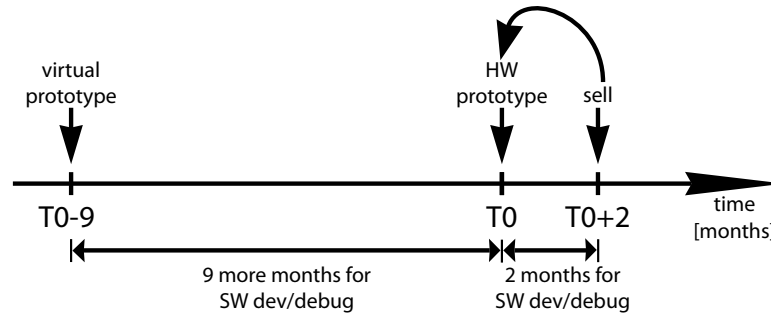


Figure 6.1.: Embedded software development/debug anticipation for the design of AMS SoCs.

A fully automated design flow of AMS systems from specifications to architecture exploration via system level simulations, to RTL synthesis down to the layout place and route is still quite a faraway achievement; but a system validation through an intertwined simulation of AMS/RF/Digital HW and SW on a unified C++ environment is now becoming a reality.

In order to enhance the IP reuse for speeding up the design phase the concept of the model reuse is not sufficient. The packaging aspect is being covered by the IP-XACT IEEE 1685-2009 standard developed by the Accellera Systems Initiative consortium for digital IPs. With respect to the AMS side, the partners of the European CATRENE MEDEA+ project called *Beyond-Design Refinement of Embedded Analogue and Mixed-signal Systems* (Beyond-DREAMS) [Beyond-DREAMS 11] have submitted a proposal for AMS extensions to the IP-XACT board in order to handle specifications for analog ports when an IP has to be packaged.

As a continuation of the Beyond-DREAMS European project another European project called Heterogeneous-INCEPTION is starting. This project will aim at enhancing the capabilities of the SystemC AMS extensions for modeling multi-sensors/actuators platforms and for power consumption optimization. From the technical viewpoint it is planned by the SystemC AMS community to add support for Finite Element Analysis MoCs together with non-linear extensions, dynamic TDF time step and bond-graph description entry capabilities. The versatility of the SystemC framework paves the way to future applications and related research fields on heterogeneity.

Appendix A.

Technical Annex

A.1. SystemC TDF based system identification

Listing A.1: SystemC-AMS TDF ARX model class

```
1  /*****
2  *          Copyright(c) 2012 TIMA Laboratory          *
3  *          Author: Fabio Cenni                      *
4  *****/
5  #ifndef ARX_01_H_
6  #define ARX_01_H_
7
8  #include <sstream>
9  #include <vector>
10 #include <iterator>
11 #include "ap.h"
12 #include "matinv.h"
13
14 template <class param_type, class data_type>
15 SCA_TDF_MODULE(ARX)
16 {
17     sca_tdf::sca_in <data_type>* in;
18     sca_tdf::sca_out <data_type>* out;
19
20 private:
21     double k;
22     int nb_in;
23     int nb_out;
24     unsigned int iteration;
25     std::vector <data_type> input_buffer;
26     std::vector <data_type> output_buffer;
27     data_type accum;
28
29     //std::vector <param_type> P;
30     std::vector< std::vector<data_type> > > u;
31     std::vector< std::vector<data_type> > > y;
32
33     int na_min, nb_min;
34     int na_max, nb_max;
35     int nk_max;
36
37     int na_opt, nb_opt, nk_opt;
38     std::vector<param_type> Popt;
39
40     template <class T>
41     bool from_string( T& t, const std::string& s, std::ios_base& (*f)(std::ios_base&))
42     {
43         std::istringstream iss(s);
44         return !(iss >> f >> t).fail();
45     }
46 }
```

```

47 void load_waveforms(std::string _filename)
48 {
49     std::vector<data_type> u_vec;
50     std::vector<data_type> y_vec;
51     std::ifstream in_file(_filename.c_str());
52     std::string line;
53     int linenum = 0;
54
55     //For each line of the csv file
56     while (getline (in_file, line))
57     {
58         std::istringstream linestream(line);
59         std::string item;
60         int itemnum = 0;
61
62         //For each field of a line
63         while (getline (linestream, item, ','))
64         {
65             // From string to data_type conversion: item to data
66             data_type data;
67             if(this -> from_string<data_type>(data, item, std::dec))
68             {
69                 //std::cout << data << std::endl;
70             }
71             else
72                 std::cout << "from_string_failed" << std::endl;
73
74             // Storing of input and output waveforms into u(nb_input) and y(nb_output)
75             if (itemnum < nb_in)
76             {
77                 u_vec.push_back(data);
78             }
79             else
80             {
81                 y_vec.push_back(data);
82             }
83             itemnum++;
84         }
85         linenum++;
86     }
87
88     std::vector<data_type> temp;
89     for(int j=0; j<nb_in ;j++)
90     {
91         for(unsigned int i=0; i<(u_vec.size()/nb_in); i+=nb_in)
92         {
93             temp.push_back(u_vec.operator [] (i+j));
94         }
95         u.push_back(temp);
96     }
97
98     temp.erase(temp.begin(), temp.end());
99     for(int j=0; j<nb_out ;j++)
100     {
101         for(unsigned int i=0; i<(y_vec.size()/nb_out); i+=nb_out)
102         {
103             temp.push_back(y_vec.operator [] (i+j));
104         }
105         y.push_back(temp);
106     }
107
108 #ifndef PLOT
109     cout << u[0].size() << endl;
110     cout << y[0].size() << endl;
111     while(!temp.empty())
112     {
113         cout << temp.back() << endl;
114         temp.pop_back();

```

```

115     }
116 #endif
117 }
118
119 void init_attributes()
120 {
121     // Init ports
122     in = new sca_tdf::sca_in<data_type>[nb_in];
123     out = new sca_tdf::sca_out<data_type>[nb_out];
124 }
125
126 void structure_quest()
127 {
128     double Jopt = 1e9;
129     double Jabs_opt;
130
131     for(int na = na_min; na <= na_max; na++) //na_max
132     {
133         for(int nb = nb_min; nb <= nb_max; nb++) //nb_max
134         {
135             for (int nk = 1; nk <= nk_max; nk++) //nk_max
136             {
137                 //std::vector<param_type> P(na+nb);
138                 std::vector<param_type> P;
139                 double Jabs;
140                 double J;
141
142                 //CORE OF THE QUEST
143                 this -> arx(P, Jabs, na, nb, nk);
144
145             }
146 #ifndef PLOT
147             cout << "P" << endl;
148             for(unsigned int i=0; i<P.size();i++)
149                 cout<< P[i] << endl;
150             cout << "Jabs_=" << Jabs << endl;
151 #endif
152
153             J = k*Jabs + (1-k)*(na+nb);
154
155             if (J < Jopt)
156             {
157                 Jopt = J;
158                 Jabs_opt = Jabs;
159                 na_opt = na;
160                 nb_opt = nb;
161                 nk_opt = nk;
162                 Popt = P;
163             }
164         }
165     }
166 }
167
168 #ifndef PLOT_BEST
169     cout << "-----" << endl;
170     cout << "Popt" << endl;
171     for(unsigned int i=0; i<Popt.size();i++)
172         cout<< Popt[i] << endl;
173     cout << "Jabs_opt_" << Jabs_opt << endl;
174     cout << "Jopt_" << Jopt << endl;
175     cout << "na_opt_" << na_opt << endl;
176     cout << "nb_opt_" << nb_opt << endl;
177     cout << "nk_opt_" << nk_opt << endl;
178 #endif
179 }
180
181 void arx(

```

```

183     std::vector<param_type>& P,
184     double& Jabs,
185     int na, int nb, int nk
186 )
187 {
188     //Init P
189     //P.erase(P.begin(), P.end());
190     //for (int i=0; i<(na+nb); i++)
191     //    P.push_back(0);
192
193     //Init variables needed for the matrix construction
194     int t = u[0].size();
195     int rownum = t - std::max(na, nb+nk-1);
196     std::vector<data_type> temp1;
197     std::vector<data_type> temp2;
198     typename std::vector<data_type>::iterator it;
199     std::vector< std::vector<data_type> > X;
200     std::vector<data_type> Y;
201
202     //Matrix X construction
203     for (int row=1; row<=rownum; row++) //rownum
204     {
205         temp1.erase(temp1.begin(), temp1.end());
206         temp2.erase(temp2.begin(), temp2.end());
207
208         it = y[0].begin();
209         temp1.insert(temp1.begin(), it+t-na-1-(row-1), it+t-1-(row-1)); // temp1 = y((t-na-(row-1)):(t-1-(
210             row-1)))'; MATLAB CODE
211         //Swap elements
212         for(unsigned int i=0; i<temp1.size(); i++)
213         {
214             temp1.insert(temp1.begin()+i, temp1.back());
215             temp1.pop_back();
216         }
217
218         it = u[0].begin();
219         temp2.insert(temp2.begin(), it+t-nk-nb-(row-1), it+t-nk-(row-1)); // temp2 = u((t-nk-nb+1-(row-1))
220             :(t-nk-(row-1)))'; MATLAB CODE
221         //Swap elements
222         for(unsigned int i=0; i<temp2.size(); i++)
223         {
224             temp2.insert(temp2.begin()+i, temp2.back());
225             temp2.pop_back();
226         }
227
228         temp1.insert(temp1.end(), temp2.begin(), temp2.end());
229
230         X.push_back(temp1);
231     }
232
233     //Construction of the Y vector of real output, for the solving of the equation system
234     it = y[0].begin();
235     Y.insert(Y.begin(), it+t-rownum, it+t); // Y=y((t-rownum+1):t); MATLAB CODE
236     //Swap elements
237     for(unsigned int i=0; i<Y.size(); i++)
238     {
239         Y.insert(Y.begin()+i, Y.back());
240         Y.pop_back();
241     }
242
243     //convert Y,X,P to Y_,X_,P_ using "alglib" libraries
244     ap::template_1d_array<data_type,true> Y_;
245     Y_.setlength(Y.size());
246     ap::template_1d_array<data_type,true> P_;
247     P_.setlength(na+nb);
248     ap::template_2d_array<data_type,true> X_;
249     X_.setlength(Y.size(), na+nb);
250     ap::template_2d_array<data_type,true> transpX_;

```

```

249     transpX_.setlength(na+nb, Y.size());
250
251     // Init Y_
252     for(unsigned int i=0; i<Y.size(); i++)
253     {
254         Y_(i) = Y[i];
255     }
256
257 #ifndef PLOT
258     cout << "Y_ " << endl;
259     for(unsigned int i=0; i<Y.size(); i++)
260         cout << Y_(i) << endl;
261 #endif
262
263     // Init P_
264     //for(unsigned int i=0; i<na+nb; i++)
265     //{
266     //    P_(i) = P[i];
267     //}
268
269     // Init X_ and transpX_
270     for(unsigned int r=0; r<Y.size(); r++)
271     {
272         for(int c=0; c<na+nb; c++)
273         {
274             X_(r,c) = X[r][c];
275             transpX_(c,r) = X[r][c];
276         }
277     }
278
279     // LEAST SQUARES LINEAR SYSTEM SOLVING  $Y=X*P \rightarrow P=?$ 
280     // pseudo_inverse =  $\text{inv}(X'*X) * X'$ ;
281
282
283     // temp= $X'*X$ 
284     //ap::template_2d_array<data_type,true> temp_;
285     ap::real_2d_array temp_;
286     temp_.setlength(na+nb, na+nb);
287     for(int i=0; i<na+nb; i++)
288     {
289         for(int j=0; j<na+nb; j++)
290         {
291             data_type res = 0;
292             for(unsigned int k=0; k<Y.size(); k++)
293                 res += transpX_(i,k) * X_(k,j);
294             temp_(i,j) = res;
295         }
296     }
297
298     //  $\text{inv}(temp)=\text{inv}(X'*X)$ 
299     int info;
300     matinvreport rep;
301     rmatrixinverse(temp_, na+nb, info, rep);
302
303     // pseudoinverse =  $\text{temp}*X' = \text{inv}(X'*X)*X'$ 
304     ap::real_2d_array pseudo_inv_;
305     pseudo_inv_.setlength(na+nb, Y.size());
306     for(int i=0; i<na+nb; i++)
307     {
308         for(unsigned int j=0; j<Y.size(); j++)
309         {
310             data_type res = 0;
311             for(int k=0; k<na+nb; k++)
312                 res += temp_(i,k) * transpX_(k,j);
313             pseudo_inv_(i,j) = res;
314         }
315     }
316

```

```

317 #ifdef PLOT
318     cout << "pseudo_inv_" << endl;
319     for(unsigned int i=0; i<na+nb; i++)
320     {
321         for(unsigned int j=0; j<Y.size(); j++)
322             cout << pseudo_inv_(i,j) << " ";
323         cout << endl;
324     }
325 #endif
326
327 // P = pseudo_inverse * Y;
328 for(int i=0; i<na+nb; i++)
329 {
330     data_type res = 0;
331     for(unsigned int k=0; k<Y.size(); k++)
332         res += pseudo_inv_(i,k) * Y_(k);
333     P_(i) = res;
334     P.push_back(res);
335 }
336
337 #ifdef PLOT
338     cout << "P_" << endl;
339     for(unsigned int i=0; i<na+nb; i++)
340         cout << P_(i) << endl;
341 #endif
342
343 // h = X * pseudo_inverse;
344 ap::real_2d_array h_;
345     h_.setlength(Y.size(), Y.size());
346     for(unsigned int i=0; i<Y.size(); i++)
347     {
348         for(unsigned int j=0; j<Y.size(); j++)
349         {
350             data_type res = 0;
351             for(int k=0; k<na+nb; k++)
352                 res += X_(i,k) * pseudo_inv_(k,j);
353             h_(i,j) = res;
354         }
355     }
356
357 #ifdef PLOT
358     cout << "h_" << endl;
359     for(unsigned int i=0; i<Y.size(); i++)
360     {
361         for(unsigned int j=0; j<Y.size(); j++)
362             cout << h_(i,j) << " ";
363         cout << endl;
364     }
365 #endif
366
367 // Jabs = e'e = Y'*Y - Y'*h*Y;
368 //1// temp1= h * Y
369 ap::real_1d_array temp1_;
370     temp1_.setlength(Y.size());
371     for(unsigned int i=0; i<Y.size(); i++)
372     {
373         data_type res = 0;
374         for(unsigned int k=0; k<Y.size(); k++)
375             res += h_(i,k) * Y_(k);
376         temp1_(i) = res;
377     }
378
379 #ifdef PLOT
380     cout << "temp1_" << endl;
381     for(unsigned int i=0; i<Y.size(); i++)
382         cout << temp1_(i) << endl;
383 #endif
384

```

```

385 //2// Jabs = Y'*Y - Y'*temp1_;
386 double res = 0;
387 for(unsigned int i=0; i<Y.size(); i++)
388     res += (Y_(i)*Y_(i)) - (Y_(i)*temp1_(i));
389 Jabs = res;
390 }
391
392 void build_u_y_without_input_file()
393 {
394     std::vector<data_type> temp_u0;
395     std::vector<data_type> temp_y0;
396
397     temp_u0.push_back(6.41);
398     temp_u0.push_back(3.41);
399     for(int i=0;i<9;i++)
400         temp_u0.push_back(6.41);
401     for(int i=0;i<10;i++)
402         temp_u0.push_back(3.41);
403     temp_u0.push_back(6.41);
404     temp_u0.push_back(3.41);
405     temp_u0.push_back(3.41);
406     for(int i=0;i<5;i++)
407         temp_u0.push_back(6.41);
408     temp_u0.push_back(3.41);
409
410     u.push_back(temp_u0);
411
412     temp_y0.push_back(4.7661);
413     temp_y0.push_back(4.7637);
414     temp_y0.push_back(4.8394);
415     temp_y0.push_back(5.003);
416     temp_y0.push_back(5.0176);
417     temp_y0.push_back(5.0567);
418     temp_y0.push_back(5.1544);
419     temp_y0.push_back(5.3619);
420     temp_y0.push_back(5.4254);
421     temp_y0.push_back(5.5695);
422     temp_y0.push_back(5.6818);
423     temp_y0.push_back(5.7429);
424     temp_y0.push_back(5.8039);
425     temp_y0.push_back(5.9187);
426     temp_y0.push_back(5.821);
427     temp_y0.push_back(5.4474);
428     temp_y0.push_back(5.0616);
429     temp_y0.push_back(4.6293);
430     temp_y0.push_back(4.2679);
431     temp_y0.push_back(4.0115);
432     temp_y0.push_back(3.8503);
433     temp_y0.push_back(3.7112);
434     temp_y0.push_back(3.5695);
435     temp_y0.push_back(3.5182);
436     temp_y0.push_back(3.6525);
437     temp_y0.push_back(3.8186);
438     temp_y0.push_back(3.8626);
439     temp_y0.push_back(4.0115);
440     temp_y0.push_back(4.3534);
441     temp_y0.push_back(4.705);
442
443     y.push_back(temp_y0);
444 }
445
446 void construction(std::string _filename)
447 {
448     #ifdef ARRAY_OF_30_VALUES_4_DBG_ECLIPSE
449         this -> build_u_y_without_input_file();
450     #else
451         this -> load_waveforms(_filename);
452     #endif

```



```

453
454     this -> init_attributes();
455     this -> structure_quest();
456
457     for(int i=0; i<(nb_opt + nk_opt); i++)
458     {
459         input_buffer.push_back(0);
460     }
461
462     for(int i=0; i<(na_opt); i++)
463     {
464         output_buffer.push_back(0);
465     }
466 }
467
468 public:
469
470     ARX(sc_core::sc_module_name,
471         std::string _filename,
472         double k,
473         int _nb_in, int _nb_out
474         ) : k(k), nb_in(_nb_in), nb_out(_nb_out), na_min(1), nb_min(1), na_max(10), nb_max(10), nk_max(10)
475     {
476         this -> construction(_filename);
477     }
478
479     ARX(sc_core::sc_module_name,
480         std::string _filename,
481         double k,
482         int _nb_in, int _nb_out,
483         int _na_max, int _nb_max, int _nk_max
484         ) : k(k), nb_in(_nb_in), nb_out(_nb_out), na_min(1), nb_min(1), na_max(_na_max), nb_max(_nb_max),
485             nk_max(_nk_max)
486     {
487         this -> construction(_filename);
488     }
489
490     ARX(sc_core::sc_module_name,
491         std::string _filename,
492         double k,
493         int _nb_in, int _nb_out,
494         int _na_min, int _na_max,
495         int _nb_min, int _nb_max,
496         int _nk_max
497         ) : k(k), nb_in(_nb_in), nb_out(_nb_out), na_min(_na_min), nb_min(_nb_min), na_max(_na_max),
498             nb_max(_nb_max), nk_max(_nk_max)
499     {
500         this -> construction(_filename);
501     }
502
503     // In case that P is provided at the instantiation , together with na,nb,nk
504     ARX(sc_core::sc_module_name,
505         int _nb_in, int _nb_out,
506         std::vector<param_type>& P,
507         int _na,
508         int _nb,
509         int _nk
510         ) : nb_in(_nb_in), nb_out(_nb_out), na_opt(_na), nb_opt(_nb), nk_opt(_nk), Popt(P)
511     {
512         this -> init_attributes();
513         for(int i=0; i<(nb_opt + nk_opt); i++)
514         {
515             input_buffer.push_back(0);
516         }
517
518         for(int i=0; i<(na_opt); i++)
519         {
520             output_buffer.push_back(0);
521         }
522     }

```

```

519     }
520 }
521
522 // In case that P is provided in two parts: Pa, Pb
523 ARX(sc_core::sc_module_name,
524     int _nb_in, int _nb_out,
525     std::vector<param_type>& Pa,
526     std::vector<param_type>& Pb,
527     int _nk
528 ) : nb_in(_nb_in), nb_out(_nb_out), na_opt(Pa.size()), nb_opt(Pb.size()), nk_opt(_nk)
529 {
530     //Popt building from Pa,Pb
531     Popt.insert(Popt.begin(), Pa.begin(), Pa.end());
532     Popt.insert(Popt.end(), Pb.begin(), Pb.end());
533
534     this->init_attributes();
535     for(int i=0; i<(nb_opt + nk_opt); i++)
536     {
537         input_buffer.push_back(0);
538     }
539
540     for(int i=0; i<(na_opt); i++)
541     {
542         output_buffer.push_back(0);
543     }
544 }
545
546
547 void processing()
548 {
549     accum = 0;
550
551     #ifdef PLOT
552     cout << "input_buffer_"<< endl;
553     for(unsigned int i=0; i<input_buffer.size();i++)
554         cout<< input_buffer[i] << endl;
555
556     cout << "output_buffer_"<< endl;
557     for(unsigned int i=0; i<output_buffer.size();i++)
558         cout<< output_buffer[i] << endl;
559     #endif
560
561     for(int i=0; i<nb_opt; i++)
562     {
563         accum += input_buffer[nk_opt-1+i] * Popt[na_opt + i];
564     }
565     for(int i=0; i<na_opt; i++)
566         accum += output_buffer[i] * Popt[i];
567
568     out->write(accum);
569
570     output_buffer.insert(output_buffer.begin(), accum);
571     output_buffer.pop_back();
572
573     input_buffer.insert(input_buffer.begin(), in->read());
574     input_buffer.pop_back();
575 }
576
577 };
578 #endif /*ARX_01_H*/

```

Listing A.2: ARX test bench main file

```

1  /*****
2  *                               *
3  *       Copyright(c) 2012 TIMA Laboratory       *
4  *       Author: Fabio Cenni                               *
5  *                               *
6  *****/
7  #include "systemc-ams.h"

```

```

6
7 #define PLOT_BEST
8 //define PLOT
9 //define ARRAY_OF_30_VALUES_4_DBG_ECLIPSE
10
11 #include "golden_model.h"
12 #include "arx_01.h"
13 #include "sub.h"
14
15 typedef double data_type;
16 typedef float param_type;
17
18 int sc_main(int argc, char* argv[])
19 {
20     //nb_in>1 and nb_out>1 handled by the waveform loader but not any further
21     int nb_in = 1;
22     int nb_out = 1;
23     double k = 0.99; // factor for considering the complexity of the system (na+nb)
24     cout << "k=" << k << endl;
25
26     /*
27     std::vector<param_type> Popt;
28     Popt.push_back(0.965887);
29     Popt.push_back(-0.0321963);
30     Popt.push_back(-0.0254143);
31     Popt.push_back(-0.11724);
32     Popt.push_back(0.0527693);
33     Popt.push_back(0.00608647);
34     Popt.push_back(0.0641913);
35     Popt.push_back(0.0617096);
36     Popt.push_back(0.0208348);
37     instance = new ARX < param_type, data_type> ("instance", nb_in, nb_out, Popt,5,4,2);
38     */
39
40     ARX<param_type, data_type>* instance;
41     instance = new ARX < param_type, data_type> ("instance", "waveform_1_1_1000.csv", k, nb_in, nb_out, /*,5*/
42         5, /*,5*/ 5, 5);
43
44     /*il codice genera un "UNKNOWN EXCEPTION ERROR" se :
45     if ((u[0].size() - std::max(na_max, nb_max + nk_max - 1)) < 0)
46         cout << "ERROR: not enough input values to calculate the Least Squares equation system for na,nb,nk up to
47             namax,nbmax,nkmax" << endl;*/
48
49     /*//Custom source instantiation
50     CUSTOM_SRC<data_type>* source;
51     source = new CUSTOM_SRC<data_type>("custom_src", nb_in);*/
52
53     /*//Golden source instantiation, values gathered from the csv-file lecture
54     GOLDEN_SRC<data_type>* source;
55     source = new GOLDEN_SRC<data_type>("golden_src", 1, "waveform_1_1_1000.csv");*/
56
57     /*Golden model instantiation
58     GOLDEN_MODEL<data_type>* g_model;
59     g_model = new GOLDEN_MODEL<data_type>("golden_model", "waveform_1_1_1000.csv");
60
61     /*Subtractor instantiation
62     SUB<data_type>* sub;
63     sub = new SUB<data_type>("sub");
64
65     /*// Automated Bonding
66     sca_tdf::sca_signal<data_type>* in;
67     sca_tdf::sca_signal<data_type>* out;
68     in = new sca_tdf::sca_signal<data_type>[nb_in];
69     out = new sca_tdf::sca_signal<data_type>[nb_out];
70
71     for(int i=0; i<nb_in; i++)
72     {
73         source->out[i](in[i]);

```

```

72     instance->in[i](in[i]);
73 }
74
75 for(int i=0; i<nb_out; i++)
76 {
77     instance->out[i](out[i]);
78 }*/
79
80 // Manual Bonding
81 sca_tdf::sca_signal<data_type> golden_in;
82 sca_tdf::sca_signal<data_type> model_out;
83 sca_tdf::sca_signal<data_type> golden_out;
84 sca_tdf::sca_signal<data_type> error;
85
86 g_model->golden_in(golden_in);
87 g_model->golden_out(golden_out);
88
89 instance->in[0](golden_in);
90 instance->out[0](model_out);
91
92 sub->in1(model_out);
93 sub->in2(golden_out);
94 sub->out(error);
95
96 sca_util::sca_trace_file* atf2 = sca_util::sca_create_tabular_trace_file("ams_trace.dat");
97 sca_util::sca_trace(atf2, golden_in, "in");
98 sca_util::sca_trace(atf2, model_out, "model_out");
99 sca_util::sca_trace(atf2, golden_out, "golden_out");
100 sca_util::sca_trace(atf2, error, "error");
101
102 cout << "-----" << endl;
103 cout << "START_SIMULATION" << endl;
104 sc_core::sc_start(1, sc_core::SC_MS);
105 cout << "STOP_SIMULATION" << endl;
106
107 return 0;
108 }

```

A.2. SystemC AMS extraction of a Laplace transfer function-based fitted model

Listing A.3: Code of the Matlab script generator from a fitted Matlab rational object

```

1  %*****
2  %*          Copyright(c) 2012 TIMA Laboratory          *
3  %*          Author: Fabio Cenni                      *
4  %*****/
5  function [] = toSCA(rational_object)
6
7  SCA_output = fopen('npoles_tf.h', 'wt');
8  fprintf(SCA_output, '#include"systemc-ams.h"\n\n');
9
10 % one pole module
11 fprintf(SCA_output, 'SCA_SDF_MODULE(2poles_tf){\n\n');
12 fprintf(SCA_output, '    sca_sdf_in<double>in;\n');
13 fprintf(SCA_output, '    sca_sdf_out<double>out;\n');
14 fprintf(SCA_output, '    doubletmp;\n');
15 fprintf(SCA_output, '    doubleB0;\n');
16 fprintf(SCA_output, '    doubleB1;\n');
17 fprintf(SCA_output, '    doubleA0;\n');
18 fprintf(SCA_output, '    doubleA1;\n');
19 fprintf(SCA_output, '    doubleA2;\n');
20 fprintf(SCA_output, '    sca_ltf_nd_ltf_1;\n');
21 fprintf(SCA_output, '    sca_vector<double>A,B,S;\n\n');
22 fprintf(SCA_output, '    voidinit()\n\n');

```

```

23 fprintf(SCA_output,'B(0)=B0;\n');
24 fprintf(SCA_output,'B(1)=B1;\n');
25 fprintf(SCA_output,'A(0)=A0;\n');
26 fprintf(SCA_output,'A(1)=A1;\n');
27 fprintf(SCA_output,'A(2)=A2;\n');
28 fprintf(SCA_output,'}\n\n');
29 fprintf(SCA_output,'void_sig_proc(){\n');
30 fprintf(SCA_output,'tmp=utl_1(B,A,S,in.read());\n');
31 fprintf(SCA_output,'out.write(tmp);\n\n');
32 fprintf(SCA_output,'void_post_proc(){}\n\n');
33 fprintf(SCA_output,'SCA_CTOR(2poles_tf){\n\n');
34 fprintf(SCA_output,'};\n\n');
35
36 % adder module
37 fprintf(SCA_output,'SCA_SDF_MODULE(adder){\n');
38
39 for n = 1 : (length(rational_object.A)/2)
40     fprintf(SCA_output,'sca_sdf_in<double>in%0.1u',n);
41     fprintf(SCA_output,';\n');
42 end
43
44 fprintf(SCA_output,'sca_sdf_out<double>out;\n\n');
45 fprintf(SCA_output,'void_init(){}\n\n');
46 fprintf(SCA_output,'void_sig_proc(){\n\n');
47 fprintf(SCA_output,'out.write(');
48 fprintf(SCA_output,'in1.read()+in2.read()');
49
50 for n = 3 : (length(rational_object.A)/2)
51     fprintf(SCA_output,'+in%0.1u.read()',n);
52 end
53
54 fprintf(SCA_output,');\n\n');
55 fprintf(SCA_output,'}\n\n');
56 fprintf(SCA_output,'SCA_CTOR(adder){\n\n');
57 fprintf(SCA_output,'};\n\n');
58
59 % "n" poles module
60 fprintf(SCA_output,'SC_MODULE(npoles_tf){\n\n');
61 fprintf(SCA_output,'sca_sdf_in<double>input;\n');
62 fprintf(SCA_output,'sca_sdf_out<double>output;\n');
63 fprintf(SCA_output,'}\n\n');
64
65 for n = 1 : length(rational_object.A)
66     fprintf(SCA_output,'sca_sdf_signal<double>out%0.1u',n);
67     fprintf(SCA_output,';\n');
68 end
69
70 fprintf(SCA_output,'}\n\n');
71 fprintf(SCA_output,'adder*sigma1;\n');
72
73 for n = 1 : length(rational_object.A)
74     fprintf(SCA_output,'2poles_tf*fract%0.1u',n);
75     fprintf(SCA_output,';\n');
76 end
77
78 fprintf(SCA_output,'SC_CTOR(npoles_tf){\n\n');
79
80 for n = 1 : (length(rational_object.A)/2)
81     fprintf(SCA_output,'fract%0.1u',n);
82     fprintf(SCA_output,'=new_2poles_tf("fract%0.1u',n);
83     fprintf(SCA_output,'");\n');
84
85     fprintf(SCA_output,'fract%0.1u',n);
86     fprintf(SCA_output,'->in(input);\n');
87
88     fprintf(SCA_output,'fract%0.1u',n);
89     fprintf(SCA_output,'->out(output%0.1u',n);
90     fprintf(SCA_output,');\n');

```

```

91
92   IA = imag(rational_object.A(2*n));
93   RA = real(rational_object.A(2*n));
94   IC = imag(rational_object.C(2*n));
95   RC = real(rational_object.C(2*n));
96
97   B0 = (-2 * RC * RA) - (2 * IC * IA);
98   B1 = 2 * RC;
99
100   A0 = RA^2 + IA^2;
101   A1 = -2 * RA;
102   A2 = 1;
103
104   fprintf(SCA_output, '          fract%0.1u', n);
105   fprintf(SCA_output, ' -> B0 = %0.10e', B0);
106   fprintf(SCA_output, ';\n');
107
108   fprintf(SCA_output, '          fract%0.1u', n);
109   fprintf(SCA_output, ' -> B1 = %0.10e', B1);
110   fprintf(SCA_output, ';\n\n');
111
112   fprintf(SCA_output, '          fract%0.1u', n);
113   fprintf(SCA_output, ' -> A0 = %0.10e', A0);
114   fprintf(SCA_output, ';\n');
115
116   fprintf(SCA_output, '          fract%0.1u', n);
117   fprintf(SCA_output, ' -> A1 = %0.10e', A1);
118   fprintf(SCA_output, ';\n');
119
120   fprintf(SCA_output, '          fract%0.1u', n);
121   fprintf(SCA_output, ' -> A2 = %0.10e', A2);
122   fprintf(SCA_output, ';\n\n');
123
124 end
125
126 fprintf(SCA_output, '    sigma1 = new_adder("sigma1");\n');
127 for n = 1 : (length(rational_object.A)/2)
128     fprintf(SCA_output, '    sigma1 -> in%0.1u(output%0.1u);\n', n, n);
129 end
130
131 fprintf(SCA_output, '    sigma1 -> out(output);\n\n');
132 fprintf(SCA_output, '    sca_trace_file* trace = sca_create_tabular_trace_file("in_out.dat");\n');
133 fprintf(SCA_output, '    sca_trace(trace, input, "input");\n');
134 fprintf(SCA_output, '    sca_trace(trace, output, "output");\n');
135 fprintf(SCA_output, '    }\n\n');
136 fprintf(SCA_output, '};\n\n');
137 fclose(SCA_output);
138 open('npoles_tf.h');

```

Listing A.4: SystemC-AMS 0.15RC5 generated code directly instantiable in the SystemC test bench

```

1  /*****
2  *                               Copyright(c) 2012 TIMA Laboratory
3  *                               Author: Fabio Cenni
4  *                               *****/
5  #include "systemc-ams.h"
6
7  SCA_SDF_MODULE(2poles_tf){
8
9      sca_sdf_in<double> in;
10     sca_sdf_out<double> out;
11
12     double tmp;
13     double B0;
14     double B1;
15     double A0;
16     double A1;
17     double A2;
18     sca_ltf_nd ltf_1;

```

```

18 sca_vector<double> A,B,S;
19
20 void init() {
21     B(0) = B0;
22     B(1) = B1;
23     A(0) = A0;
24     A(1) = A1;
25     A(2) = A2;
26 }
27
28 void sig_proc(){
29     tmp = ltf_1(B,A,S,in.read());
30     out.write(tmp);
31 }
32 void post_proc() {}
33 SCA_CTOR(2poles_tf) {}
34 };
35
36 SCA_SDF_MODULE(adder){
37     sca_sdf_in<double> in1;
38     sca_sdf_in<double> in2;
39     sca_sdf_out<double> out;
40
41     void init() {}
42     void sig_proc(){
43
44         out.write(in1.read() + in2.read());
45     }
46     SCA_CTOR(adder) {}
47 };
48
49 SC_MODULE(npoles_tf){
50
51     sca_sdf_in<double> input;
52     sca_sdf_out<double> output;
53     sca_sdf_signal<double> output1;
54     sca_sdf_signal<double> output2;
55     sca_sdf_signal<double> output3;
56     sca_sdf_signal<double> output4;
57
58     adder*      signal1;
59     2poles_tf*  fract1;
60     2poles_tf*  fract2;
61     2poles_tf*  fract3;
62     2poles_tf*  fract4;
63     SC_CTOR(npoles_tf){
64
65         fract1 = new 2poles_tf("fract1");
66         fract1->in(input);
67         fract1->out(output1);
68         fract1->B0 = -7.0653920664e+015;
69         fract1->B1 = 7.9360631032e+006;
70
71         fract1->A0 = 2.6256034035e+018;
72         fract1->A1 = 9.3511469997e+006;
73         fract1->A2 = 1.0000000000e+000;
74
75         fract2 = new 2poles_tf("fract2");
76         fract2->in(input);
77         fract2->out(output2);
78         fract2->B0 = -1.6764782785e+015;
79         fract2->B1 = -1.2206010088e+007;
80
81         fract2->A0 = 2.6586739213e+018;
82         fract2->A1 = 1.0226052047e+007;
83         fract2->A2 = 1.0000000000e+000;
84
85         signal1 = new adder("signal1");

```

```

86     sigma1->in1(output1);
87     sigma1->in2(output2);
88     sigma1->out(output);
89
90     sca_trace_file* trace = sca_create_tabular_trace_file("in_out.dat");
91     sca_trace(trace, input, "input");
92     sca_trace(trace, output, "output");
93 }
94 };

```

A.3. C-code for the MIPS32 embedded firmware

Listing A.5: Embedded C firmware

```

1  /*
2  * SOCLIB_GPL_HEADER_BEGIN
3  *
4  * This file is part of SoCLib, GNU GPLv2.
5  *
6  * SoCLib is free software; you can redistribute it and/or modify
7  * it under the terms of the GNU General Public License as published by
8  * the Free Software Foundation; version 2 of the License.
9  *
10 * SoCLib is distributed in the hope that it will be useful, but
11 * WITHOUT ANY WARRANTY; without even the implied warranty of
12 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
13 * General Public License for more details.
14 *
15 * You should have received a copy of the GNU General Public License
16 * along with SoCLib; if not, write to the Free Software
17 * Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA
18 * 02110-1301, USA.
19 *
20 * SOCLIB_GPL_HEADER_END
21 *
22 * Copyright (c) UPMC, Lip6, SoC
23 */
24
25 #include "system.h"
26 #include "../segmentation.h"
27
28 #include "soclib/uart.h"
29 #include "soclib/timer.h"
30 #include "soclib/icu.h"
31 #include "soclib/camera_reg.h"
32 #include <stdio.h>
33 #include "table.h"
34
35 static const int period = 100000;
36 volatile uint32_t cptms=0;
37 volatile uint32_t irq_flag;
38 uint32_t table_dist[4];
39 uint32_t id;
40
41 void irq_handler(int irq)
42 {
43     uint32_t iv;
44     uint32_t ti;
45     uint32_t d;
46
47     iv = soclib_io_get(base(ICU), ICU_IT_VECTOR);
48
49     switch (iv) {
50     case 0:
51         d=soclib_io_get(base(UART),UART_DATA);
52         if (d!=0) {

```



```

53     table_dist[d>>28]= ((d & 0xffffffff) == 0xffffffff) ? -1 : d & 0xffffffff ;
54     printf("data_%d_received_from_car_%d\n", d & 0xffffffff, d>>28);
55     printf("table_des_ecarts_:_%d_%d_%d_%d\n",table_dist[0],table_dist[1],table_dist[2],table_dist[3]);
56 }
57 break;
58 case 1:
59     cptms++;
60     cptms=cptms%40;
61     soclib_io_set(base(TIMER), 0*TIMER_SPAN+TIMER_RESETHIRQ, 0);
62     irq_flag=1;
63     break;
64 case 2:
65     table_dist[id] = calculate_dist((uint32_t *)base(CAM));
66     soclib_io_set(base(CINE), 0, table_dist[id]);
67     break;
68 }
69 }
70
71 void init() {
72     irq_flag=0;
73     id = soclib_io_get(base(UART), UART_ID);
74     soclib_io_set(base(UART), UART_BR, 416);
75     soclib_io_set(base(UART), UART_CTRL, 5); //reception mode
76     soclib_io_set(base(TIMER), 0*TIMER_SPAN+TIMER_PERIOD, period);
77     soclib_io_set(base(TIMER), 0*TIMER_SPAN+TIMER_MODE, TIMER_RUNNING|TIMER_IRQ_ENABLED);
78     soclib_io_set(base(ICU), ICU_MASK_SET, 7);
79     soclib_io_set(base(CAM), CAM_ITROW, 50);
80     soclib_io_set(base(CAM), CAM_ITPIX, 0);
81     set_irq_handler(irq_handler);
82     enable_hw_irq(0);
83     irq_enable();
84 }
85
86 int main() {
87     uint32_t data;
88     uint8_t send_msg=0;
89     uint32_t first=1;
90
91     init();
92     printf("Hello_from_cars_%d\n", id);
93
94     while (1) {
95         if (irq_flag) {
96             irq_flag=0;
97             if (cptms == (id+1)*8) {
98                 soclib_io_set(base(UART), UART_CTRL, 2);
99                 soclib_io_set(base(UART), UART_DATA, (table_dist[id] & 0xffffffff) | (id << 28));
100                 printf("sending_data...\n");
101                 printf("table_des_ecarts_:_%d_%d_%d_%d\n",table_dist[0],table_dist[1],table_dist[2],table_dist[3]);
102             }
103             while ( (soclib_io_get(base(UART), UART_STATUS) & 0x2) == 0x2 )
104                 pause();
105             soclib_io_set(base(UART), UART_CTRL, 5);
106         }
107         pause();
108     }
109
110     return 0;
111 }

```


List of Figures

1.1. Technology push and market pull.	6
1.2. V-model of the design process of a technological system.	9
1.3. Behavioral modeling cases for heterogeneous mixed-signal devices.	10
2.1. The Cell Broadband Engine is a heterogeneous chip (a), a CPU in combination with a GPU is a heterogeneous system (b), and a CPU in combination with an FPGA is also a heterogeneous system (c).	16
2.2. Many flows co-existing for the design flow of WSN node.	18
2.3. Explosion in levels of abstraction of a heterogeneous analog and mixed-signal system.	20
2.4. Modeling languages for various physical domains at different abstraction levels, inspired by [O'Connor 07].	24
3.1. Abstractions in relation to the SystemC AMS models of computation.	33
3.2. A basic TDF model with 3 TDF modules and 2 TDF signals [Barnasconi 10].	34
3.3. Example of a basic ELN model representing an electrical network [Barnasconi 10].	36
3.4. Example of a basic LSF model composed of 4 LSF modules [Barnasconi 10].	36
3.5. Architecture of the OSCI SystemC AMS extensions 1.0 standard.	37
4.1. Methodologies for analog behavioral modeling from starting knowledge.	47
4.2. State space-based block diagram of a dynamic linear system.	49
4.3. Case study.	50
4.4. Rearrangement step result.	51
4.5. State space model.	51
4.6. SystemC AMS macro-modeling of dynamic components by means of a linear approximation.	54
4.7. Structure of the n-poles model.	56
4.8. SystemC AMS model generation flow.	57

4.9. SystemC AMS macro-modeling of static non-linear components.	57
4.10. System identification principle.	60
4.11. Modeling of a dynamic system.	61
4.12. The concept of system identification extensions to SystemC AMS.	64
4.13. SystemC AMS-based system identification test bench with both operating and test modes.	65
4.14. Hair dryer ARX model constraints and resulting identified model.	66
4.15. Waveforms resulting from the hair dryer ARX model identification.	66
4.16. Model building during the design phase.	68
4.17. Identification algorithm for deriving an input/output model structure.	69
4.18. Transistor level circuits.	71
4.19. Input-output characteristic of the envelope detector.	72
4.20. Adaptive logical control strategy.	73
4.21. LNA performances versus power supply.	75
4.22. LNA consumption versus power supply.	76
4.23. Sampled input and output signal.	76
4.24. Predicted and simulated output values of the CUC.	77
4.25. LNA performances prediction.	78
4.26. Transistor-level simulation of the control strategy.	79
4.27. Structure of the SAW device.	81
4.28. Instantaneous E-field direction in the SAW crossed-field model.	82
4.29. Equivalent circuit for an IDT electrode and its associated gap between metaliza- tions, based on the SAW crossed-field Mason's model (a) and the corresponding three-port building block (b).	82
4.30. Equivalent network of both IDTs and the delay path of the SAW chemical sensor.	82
4.31. Frequency response of the SAW sensor using the Matlab models and the rational fitting function.	83
4.32. Effect of SAW velocity changes in the crossed-field model.	84
4.33. SAW PLL-based measurement architecture.	86
4.34. Block diagram of the overall measurement system.	86
4.35. Detailed block diagram of the PLL structure.	86

4.36. Low phase noise differential delay cell of the VCRO.	87
4.37. VCRO with differential control and differential external tuning.	88
4.38. VCRO output frequency versus differential voltage input.	88
4.39. VCRO non-linear behavior.	89
4.40. Gilbert cell mixer.	89
4.41. Differential RC low-pass filter.	90
4.42. Digital read-out and control unit of the overall system.	91
4.43. Layout of the overall chip.	92
4.44. Simulation of a 10ppm gas concentration variation.	92
4.45. Phase noise analysis on the V_{LOD} signal.	93
4.46. Photo of the fabricated chip with the pads used for electrical testing.	94
4.47. Experimental test bench.	95
4.48. Photo of the EPCB inside the protection metal box.	95
4.49. Spectrum of the VCRO output obtained by the spectrum analyzer, with a span of 2 MHz.	96
4.50. Waveform of the VCRO output obtained with the oscilloscope.	96
4.51. Polynomial interpolation of the VCO static behavior.	98
4.52. SystemC model of the overall chemical sensor.	99
4.53. Control unit outputs.	100
4.54. Low-pass filter outputs.	100
4.55. SystemC DE signals of the digital read-out unit.	101
4.56. Case studies for the validation of analog behavioral modeling flows.	102
5.1. Transistor level schematic of the 1T75 pixel architecture.	106
5.2. Section of the optical path of a CMOS image sensor [Cohen 06].	107
5.3. Structure of SystemC AMS ELN-TDF model.	108
5.4. Block diagram of the SystemC AMS model.	112
5.5. IIB output affected by ERS effect.	112
5.6. Modeled attenuation of the light intensity.	113
5.7. Quantum efficiency of the CFA.	114

5.8. Bayer filter modeling.	114
5.9. Arc tangent based discharge model issued from characterization data.	116
5.10. Electrons collected as function of the integration time and the illuminance.	117
5.11. Synoptic of the data processing within the SystemC AMS TDF CIS model.	118
5.12. SystemC TLM 2.0 platform: top view.	121
5.13. ISP correction and interpolation.	122
5.14. Simulation results of the SystemC AMS/TLM platform.	124
5.15. The modeled PMBS system, user's view.	125
5.16. The modeled PMBS system.	127
5.17. SystemC / SystemC AMS interfacing through VCI interconnects.	127
5.18. Overview of the CATSEYE image processing and embedded firmware.	128
5.19. SystemC AMS/ TLM interfacing inside the CATSEYE virtual platform.	130
5.20. Overview of ST-Ericsson's platform for mobile applications.	131
5.21. SystemC AMS/TLM interfacing with ST-Ericsson's virtual platform.	131
5.22. Results of the image processing from the acquisition (sensor input) through the ISP processing.	132
6.1. Embedded software development/debug anticipation for the design of AMS SoCs.	137
1. Modèle en "V" du processus de conception d'un produit technologique.	184
2. Types de modélisation comportementale pour des dispositifs hétérogènes et à signaux mixtes.	185
3. Abstractions par rapport aux modèles de calcul de SystemC AMS.	188
4. Macro modélisation de composants non linéaires statiques en SystemC AMS.	192
5. Macro modélisation de composants dynamiques en SystemC AMS au moyen d'une approximation linéaire.	193
6. Le concept de l'extension de SystemC AMS pour l'identification de systèmes.	194
7. Méthodes de modélisation comportementales et cas d'études.	195
8. Synoptique du modèle en SystemC AMS TDF du capteur d'images.	198
9. Gain de temps et de fiabilité de SoCs AMS.	203

List of Tables

4.1.	Performance prediction during logical control.	79
4.2.	Frequency shifts for different simulated gas concentration.	93
5.1.	Simulation performances.	119
1.	Comparaison entre les vitesses de simulation des différents modèles.	199

List of Acronyms

ADC	Analog to Digital Converter
AE	Auto Exposition
AF	Auto Focus
AHDL	Analog Hardware Description Languages
AMS	Analog and Mixed-Signal
AMSWG	AMS Working Group
ANR	Agence Nationale de la Recherche
API	Application Programming Interface
ARX	Auto-Regressive, eXogenous
ASIC	Application-Specific Integrated Circuit
AWB	Automatic White Balancing
AT	Approximately Timed
BF	Bayer Filter
BGB	BackGround Builder
CAD	Computer Aided Design
CBEA	Cell Broadband Engine Architecture
CFA	Color Filter Array
CIS	CMOS Image Sensor
CM	Common Mode
CMOS	Complementary Metal-Oxide Semiconductor
CPU	Central Processing Unit
CT	Continuous Time
CUC	Circuit Under Control
DAC	Digital to Analog Converter
DAE	Differential Algebraic Equation

DE	Discrete Event
DSNU	Dark Signal Non-Uniformity
EDA	Electronic Design Automation
ELN	Electrical Linear Network
EPCB	Evaluation Printed Circuit Board
ERS	Electronic Rolling Shutter
ESL	Electronic System Level
FE	Finite Element
FEM	Finite Element Method
FEA	Finite Element Analysis
FIFO	Firs-In First-Out
FPGA	Field Programmable Gate Array
FPN	Fixed Pattern Noise
FSM	Finite State Machine
GPU	Graphics Processing Unit
HDL	Hardware Description Language
HW	HardWare
IC	Integrated Circuit
IDT	InterDigitated Transducer
IIB	Input Image Builder
IP	Intellectual Property
ISP	Image Signal Processor
ISS	Instruction Set Simulator
LoC	Laboratory on a Chip
LMS	Least Mean Square
LNA	Low Noise Amplifier
LRM	Language Reference Manual
LSF	Linear Signal Flow
LT	Loosely Timed

MCM	Multi-Chip Module
MEMS	Micro Electro-Mechanical System
MoC	Model of Computation
MPSoC	Multi-Processor System on Chip
NARX	Non-linear Auto-Regressive, eXogenous
NL	Non Linear
OB	Object Builder
ODE	Ordinary Differential Equation
OSCI	Open SystemC Initiative
PD	Phase Detector
PDES	Parallele Discrete Event Simulation
PLL	Phase Locked Loop
PMBS	Precollision Mitigation Braking System
PoC	Proof of Concept
PRNU	Photo-Response Non-Uniformity
PV	Programmer View
PVT	Programmer View with Time
PRBS	Pseudo-Random Binary Sequence
RF	Radio Frequency
RGB	Red-Green-Blue
RLMS	Recursive Least Mean Square
ROM	Reduced Order Modeling
RoO	Region of Operation
RTL	Register Transfer Level
SAW	Surface Acoustic Wave
SDF	Synchronous Data Flow
SE	Single Ended
SiP	System in Package
SISO	Single-Input Single-Output

SMIA	Standard Mobile Imaging Architecture
SMP	Symmetric Multi-Processor
SoC	System on Chip
SW	SoftWare
TDF	Timed Data Flow
TDMA	Time-Division Multiple Access
TIMA	Techniques of Informatics and Microelectronics for integrated systems Architectures
TLM	Transaction-Level Modeling
VCO	Voltage Controlled Oscillator
VCRO	Voltage Controlled Ring Oscillator
WASABI	Wireless system And SystemC-AMS Basic Infrastructure
WSN	Wireless Sensor Network

Bibliography

- [Abdallah 10] L. Abdallah, H.-G. Stratigopoulos, C. Kelma & S. Mir. *Sensors for built-in alternate RF test*. In Test Symposium (ETS), 2010 15th IEEE European, pages 49–54, may 2010.
- [Accellera 09] Accellera. *Verilog-AMS Language Reference Manual Version 2.3.1: Analog and Mixed-Signal Extensions to Verilog HDL*. Accellera. 1370 Trancas Street, 163, Napa, CA 94558, USA, June 2009.
- [Adhikari 10] Sumit Adhikari & Christoph Grimm. *Modeling switched capacitor sigma delta modulator nonidealities in SystemC-AMS*. In Specification Design Languages (FDL 2010), 2010 Forum on, pages 1–6, sept. 2010.
- [Al-Junaid 05] H. Al-Junaid & T. Kazmierski. *Analogue and mixed-signal extension to SystemC*. Circuits, Devices and Systems, IEE Proceedings -, pages 682–690, dec. 2005.
- [Al-Junaid 06] Hessa Al-Junaid, Tom Kazmierski & Leran Wang. *SystemC-A modeling of an automotive seating vibration isolation system*. In Forum on Specification and Design Languages (FDL 2006), pages 107–113, 2006.
- [Arndt 10] Thomas Arndt, Thomas Uhle, Karsten Einwich & Ingmar Neumann. *Using SystemC AMS for heterogeneous systems modelling at TIER-1 level*. In Specification Design Languages (FDL 2010), 2010 Forum on, pages 1–6, sept. 2010.
- [Balarin 03] F. Balarin, Y. Watanabe, H. Hsieh, L. Lavagno, C. Passerone & A. Sangiovanni-Vincentelli. *Metropolis: an integrated electronic system design environment*. Computer, vol. 36, no. 4, pages 45–52, april 2003.
- [Barnasconi 10] M. Barnasconi, C. Grimm, M. Damm, K. Einwich, M-M. Louërat, T. Maehne, F. Pecheux & A. Vachoux. *SystemC AMS extensions User’s Guide*. Open SystemC Initiative (OSCI), Mar. 8 2010.
- [Beserra 11] G.S. Beserra, J.E.G. de Medeiros, A.M. Sampaio & J.C. da Costa. *System-level modeling of a mixed-signal System on Chip for Wireless Sensor Networks*. In Design, Automation Test in Europe Conference Exhibition (DATE), 2011, pages 1–4, march 2011.
- [Beyond-DREAMS 11] Beyond-DREAMS. *Beyond-Design Refinement of Embedded Analogue and Mixed-Signal Systems (Beyond-DREAMS)*, CATRENE MEDEA+, URL: <https://wiki.eas.iis.fraunhofer.de/beyonddreams/index.php/Abstract> last checked Feb 2012, Partners: Infineon Technologies AG, Robert Bosch GmbH, Fraunhofer IIS/EAS, Infineon Tech-

- nologies Austria AG, Vienna University of Technology, STMicroelectronics, Magillem Design Services, CEA-LETI, TIMA/Institut Polytechnique de Grenoble, NXP Semiconductors, Dizain-Sync, Twente Institute for Wireless and Mobile Communications, Delft University of Technology (TU Delft), IMEC-NL, École polytechnique fédérale de Lausanne, Duration: 2008-2011.*
- [Biagetti 04] Giorgio Biagetti, Marco Caldari, Massimo Conti & Simone Orcioni. *Extending SystemC to Analog Modelling and Simulation*. In Christoph Grimm, editeur, *Languages for System Specification*, pages 229–242. Springer US, 2004. 10.1007/1-4020-7991-5_15.
- [Bjureus 01] P. Bjureus & A. Jantsch. *Modeling of mixed control and dataflow systems in MASCOT*. *Very Large Scale Integration (VLSI) Systems*, IEEE Transactions on, vol. 9, no. 5, pages 690–703, oct. 2001.
- [Black 04] David C. Black & Jack Donovan. *Systemc: from the ground up*. Springer Science+Business Media, LLC, 2004.
- [Bonnerud 01] T.E. Bonnerud, B. Hernes & T. Ytterdal. *A mixed-signal, functional level simulation framework based on SystemC for system-on-a-chip applications*. In *Custom Integrated Circuits*, 2001, IEEE Conference on., pages 541–544, 2001.
- [Bousquet 11a] L. Bousquet, F. Cenni & E. Simeu. *Inclusion of Power Consumption Information in High-Level Modeling of Linear Analog Blocks*. *Journal of Low Power Electronics (JOLPE)*, vol. 7, no. 4, pages 541–551, December 2011.
- [Bousquet 11b] L. Bousquet, F. Cenni & E. Simeu. *SystemC-AMS High-level Modeling of Linear Analog Blocks with Power Consumption Information*. In *12th IEEE Latin-American Test Workshop (LATW)*, pages 1–6, March 2011.
- [Bowen 98] Matthew Bowen. *Handel-C Language Reference Manual Version 2.1*. Embedded Solutions Limited, 1998.
- [Brodtkorb 10] A. R. Brodtkorb, C. Dyken, T. R. Hagen, J. M. Hjelmervik & O. O. Storaasli. *State-of-the-art in heterogeneous computing*. *Scientific Programming Journal*, vol. 18, no. 1, pages 1–33, 2010.
- [Brooks 10] C. Brooks, E.A. Lee & S. Tripakis. *Exploring models of computation with Ptolemy II*. In *Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, 2010 IEEE/ACM/IFIP International Conference on, pages 331–332, oct. 2010.
- [Caluwaerts 08] K. Caluwaerts, D. Galayko & P. Basset. *SystemC-AMS heterogeneous modeling of a capacitive harvester of vibration energy*. In *Behavioral Modeling and Simulation Workshop (BMAS)*, 2008.
- [Cenni 08] F. Cenni. *Behavioral modeling of saw sensors and transistor level design of their microelectronic front-end interface*. Master’s thesis, Università di Bologna, Facoltà di ingegneria elettronica, March 2008.

- [Cenni 09a] F. Cenni, S. Mir & L. Rufer. *Behavioral modeling and simulation of a chemical sensor with its microelectronics front-end interface*. In 3rd IEEE International Workshop on Advances in Sensors and Interfaces (IWASI), pages 92–97, June 2009.
- [Cenni 09b] F. Cenni, E. Simeu & S. Mir. *Macro-modeling of analog blocks for SystemC-AMS simulation: A chemical sensor case-study*. In 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC), pages 211–214, October 2009.
- [Cenni 10] F. Cenni, J. Cazalbou, S. Mir & L. Rufer. *Design of a SAW-based chemical sensor with its microelectronics front-end interface*. *Microelectronics Journal*, Ed. Elsevier, vol. 41, no. 11, pages 723–732, November 2010.
- [Cenni 11a] F. Cenni, S. Scotti & E. Simeu. *Behavioral modeling of a CMOS video sensor platform using SystemC AMS / TLM*. In IEEE Forum for Design Languages (FDL), pages 1–6, September 2011.
- [Cenni 11b] F. Cenni, S. Scotti & E. Simeu. *SystemC-AMS behavioral modeling of a CMOS video sensor*. In 19th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), pages 380–385, October 2011.
- [Cenni 11c] F. Cenni, S. Scotti & E. Simeu. *A SystemC AMS/TLM platform for CMOS video sensors*. In Design and Architectures for Signal and Image Processing (DASIP), 2011 IEEE Conference on, pages 1–6, nov. 2011.
- [Cohen 06] M. Cohen, F. Roy, D. Herault, Y. Cazaux, A. Gandolfi, J.P. Reynard, C. Cowache, E. Bruno, T. Girault, J. Vaillant, F. Barbier, Y. Sanchez, N. Hotellier, O. LeBorgne, C. Augier, A. Inard, T. Jagueneau, C. Zinck, J. Michailos & E. Mazaleyrat. *Fully Optimized Cu based process with dedicated cavity etch for 1.75 μ m and 1.45 μ m pixel pitch CMOS Image Sensors*. In International Electron Devices Meeting (IEDM), pages 1–4, December 2006.
- [Coyitangiye 06] L.-A. Coyitangiye & R. Grisel. *Compact Modeling of MOSFET with VHDL-AMS language*. In IEEE Industrial Electronics, IECON 2006 - 32nd Annual Conference on, pages 2927–2932, nov. 2006.
- [Dadouche 08] F. Dadouche, A. Pinna, P. Garda & A. Alexandre-Gauthier. *Modelling of pixel sensors for image systems with VHDL-AMS*. *International Journal of Electronics*, vol. 95, no. 3, pages 211–225, 2008.
- [Damm 08] M. Damm, C. Grimm, J. Haas, A. Herrholz & W. Nebel. *Connecting SystemC-AMS models with OSCI TLM 2.0 models using temporal decoupling*. In Specification, Verification and Design Languages, 2008. FDL 2008. Forum on, pages 25–30, sept. 2008.
- [De Smedt 99] B. De Smedt & G. Gielen. *Models for systematic design and verification of frequency synthesizers*. *Circuits and Systems II: Analog and Digital*

- Signal Processing, IEEE Transactions on, vol. 46, no. 10, pages 1301–1308, oct 1999.
- [der Plas 02] G. Van der Plas, G. Gielen & W. Sansen. A computer-aided design and synthesis environment for analog integrated circuits. Boston, Dordrecht, London, Kluwer Academic Publishers, 2002.
- [Einwich 06] Karsten Einwich, Jens Bastian, Christoph Clauss, Uwe Eichler & Peter Schneider. *SystemC-AMS Extension Library for Modeling Conservative Nonlinear Dynamic Systems*. In FDL, pages 113–119. ECSI, 2006.
- [Gajski 00] Daniel D. Gajski, Jianwen Zhu, Rainer Domer, Andreas Gerstlauer & Shuqing Zhao. Specc: Specification language and methodology. Springer, 1 edition, mar 2000.
- [Ghenassia 06] Frank Ghenassia. Transaction-level modeling with systemc: Tlm concepts and applications for embedded systems. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
- [Gielen 05] G.G.E. Gielen. *CAD tools for embedded analogue circuits in mixed-signal integrated systems on chip*. Computers and Digital Techniques, IEE Proceedings -, vol. 152, no. 3, pages 317–332, may 2005.
- [Grobelny 06] D. Grobelny & L. Rufer. *Simulation of SAW-based chemical sensor*. In WOFEX, pages 213–218, September 2006.
- [Grotker 02] Thorsten Grotker. System design with systemc. Kluwer Academic Publishers, Norwell, MA, USA, 2002.
- [Hartmann 09] P.A. Hartmann, P. Reinkemeier, A. Rettberg & W. Nebel. *Modelling control systems in SystemC AMS; Benefits and limitations*. In SOC Conference, 2009. SOCC 2009. IEEE International, pages 263–266, sept. 2009.
- [Herrera 06] F. Herrera & E. Villar. *A framework for embedded system specification under different models of computation in SystemC*. In Design Automation Conference, 2006 43rd ACM/IEEE, pages 911–914, 0-0 2006.
- [Herrera 07] F. Herrera, E. Villar & C. Grimm. *A general approach to the interoperability of HetSC and SystemC-AMS*. In Specification Design Languages, 2007. FDL 20097 Forum on, pages 101–110, sept. 2007.
- [Hsieh 06] H.H. Hsieh & L.H. Lu. *Integrated CMOS power sensors for RF BIST applications*. In VLSI Test Symposium, pages 234–239, 2006.
- [IEEE 04] IEEE. *IEC/IEEE Behavioural Languages - Part 4: Verilog Hardware Description Language (Adoption of IEEE Std 1364-2001)*. IEC 61691-4 First edition 2004-10; IEEE 1364, pages 0_1 –860, 2004.
- [IEEE 07] IEEE. *IEEE Standard VHDL Analog and Mixed-Signal Extensions*. IEEE Std 1076.1-2007 (Revision of IEEE Std 1076.1-1999), pages c1

–328, 15 2007.

- [IEEE 09a] IEEE. *Behavioural languages - Part 7: SystemC Language Reference Manual*. IEEE Std 1666 IEC61691-7 Edition 1.0 2009-12, pages 1–447, 14 2009.
- [IEEE 09b] IEEE. *IEEE Standard for System Verilog-Unified Hardware Design, Specification, and Verification Language*. IEEE STD 1800-2009, pages C1 –1285, 2009.
- [IEEE 09c] IEEE. *IEEE Standard VHDL Language Reference Manual*. IEEE Std 1076-2008 (Revision of IEEE Std 1076-2002), pages c1 –626, 26 2009.
- [IIS/EAS 10] Fraunhofer IIS/EAS. *Homepage of the SystemC-AMS Proof-of-Concept (PoC) implementation developed by Fraunhofer IIS/EAS*. 2010.
- [Initiative 12] Accellera Systems Initiative. <http://www.accellera.org/home/>. 2012.
- [Kester 09] W. Kester. *Converting Oscillator Phase Noise to Time Jitter*. Analog Devices MT-008 tutorial, 2009.
- [Khankhua 11] S. Khankhua, M.W. Ashraf, S. Tayyaba, N. Afzulpurkar & C. Punyasai. *Simulation of MEMS based micro-gyroscope using CoventorWare*. In Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011 8th International Conference on, pages 22–25, may 2011.
- [Khereddine 10] R. Khereddine, L. Abdallah, E. Simeu, S. Mir & F. Cenni. *Adaptive Logical Control of RF LNA performances for efficient energy consumption*. In 18th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), Best Paper Award, pages 161–166, September 2010.
- [Khereddine 12] R. Khereddine, L. Abdallah, E. Simeu, S. Mir & F. Cenni. *Vlsi-soc 2010 book: Forward-looking trends in ic and system design*, chapitre Adaptive Logical Control of RF LNA performances for efficient energy consumption, pages 43–68. Springer, 2012.
- [Kreuter 09] H.-P. Kreuter, V. Kosel, M. Glavanovics & R. Illing. *System level modeling of smart power switches using SystemC-AMS for digital protection concept verification*. In Behavioral Modeling and Simulation Workshop, 2009. BMAS 2009. IEEE, pages 37 –42, sept. 2009.
- [Lambert 10] Alain Lambert, Dominique Gruyer, Anthony Busson & Husnain Mansoor Ali. *Usefulness of collision warning inter-vehicular system*. International Journal of Vehicle Safety, vol. 5, no. 1, pages 60–74, January 2010.
- [Laskaris 05] N. Laskaris. *Vignetting*. School of Informatics. University of Edinburgh, 2005.

- [Lévêque 12] A. Lévêque, F. Pêcheux, M-M. Louërat, A. Aboushady, F. Cenni, S. Scotti, A. Massouri & L. Clavier. *Holistic Modelling of Heterogeneous Embedded Systems with High Multi-Discipline Feedback: Application to a Precollision Mitigation Braking System*. In Design, Automation Test in Europe Conference Exhibition (DATE), 2012, page To Appear, march 2012.
- [Ljung 08] L. Ljung. *Perspectives on System Identification*. In 17th International Federation of Automatic Control (IFAC) World Congress, 2008. Plenary.
- [Lu 05] Z.-Q. Lu, F.-C. Lai & J.-G. Ma. *A low-phase-noise CMOS ring oscillator with differential control*. In 6th International Conference on ASIC (ASICON), pages 540–543, October 2005.
- [Lu 09] Ping Lu, D. Glaser, G. Uygur, S. Weichslgartner, K. Helmreich & A. Lechner. *Mixed-signal test development using open standard modeling and description languages*. In Behavioral Modeling and Simulation Workshop, 2009. BMAS 2009. IEEE, pages 78–83, sept. 2009.
- [Luo 05] Jun Luo & Jean-Pierre Hubaux. *A Survey of Research in Inter-Vehicle Communications*. In Embedded Security in Cars –Securing Current and Future Automotive IT Applications. Springer-Verlag, 2005.
- [Mähne 06] Torsten Mähne, Kersten Kehr, Axel Franke, Jörg Hauer & Bertram Schmidt. *Creating Virtual Prototypes of Complex MEMS Transducers Using Reduced-Order Modelling Methods and VHDL-AMS*. In Applications of Specification and Design Languages for SoCs: Selected papers from FDL’05, The ChDL series, pages 135–153. Springer, 2006.
- [Malik 08] A.F. Malik, M. Shoaib, S. Naseem & S. Riaz. *Modeling and designing of RF MEMS switch using ANSYS*. In Emerging Technologies, 2008. ICET 2008. 4th International Conference on, pages 44–49, oct. 2008.
- [Mantooth 03] H.A. Mantooth, L. Ren, X. Huang, Y. Feng & W. Zheng. *A survey of bottom-up behavioral modeling methods for analog circuits*. In Proceedings of the 2003 International Symposium on Circuits and Systems (ISCAS), volume 3, pages III–910 – III–913 vol.3, may 2003.
- [Markert 07] E. Markert, M. Dienel, G. Herrmann & U. Heinkel. *SystemC-AMS Assisted Design of an Inertial Navigation System*. Sensors Journal, IEEE, vol. 7, no. 5, pages 770–777, may 2007.
- [Mathaikutty 06] D. A. Mathaikutty, H. D. Patel, S. K. Shukla & A. Jantsch. Applications of specification and design languages for socs. selected papers from fdl’05, chapitre UMoC++: A C++-Based Multi-MoC Modeling Environment, page 115–130. Springer, 2006.
- [MATLAB 11] MATLAB. Matlab - the language of technical computing. The Math-Works, Inc., 1984-2011.
- [Mello 10] A. Mello, I. Maia, A. Greiner & F. Pecheux. *Parallel simulation of systemC TLM 2.0 compliant MPSoC on SMP workstations*. In Design,

- Automation Test in Europe Conference Exhibition (DATE), 2010, pages 606–609, march 2010.
- [Mitea 11] O. Mitea, M. Meissner & L. Hedrich. *Topology synthesis of analog circuits with yield optimization and evaluation using pareto fronts*. In VLSI and System-on-Chip (VLSI-SoC), 2011 IEEE/IFIP 19th International Conference on, pages 78–81, oct. 2011.
- [Moser 99] Eduard Moser & Wolfgang Nebel. *Case Study: System Model of Crane and Embedded Control*. In DATE, page 721. IEEE Computer Society, 1999.
- [Nathke 04] L. Nathke, V. Burkhay, L. Hedrich & E. Barke. *Hierarchical automatic behavioral model generation of nonlinear analog circuits based on nonlinear symbolic techniques*. In Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings, volume 1, pages 442–447 Vol.1, feb. 2004.
- [Navarro 05] D. Navarro, D. Ramat, F. Mieyeville, I. O’Connor, F. Gaffiot & L. Carrel. *VHDL & VHDL-AMS modelling and simulation of a CMOS imager IP*. In Forum on specification and Design Languages (FDL), pages 179–183, Sempember 2005.
- [O’Connor 06] I. O’Connor. Applications of specification and design languages for socs, chapitre UML/XML-Based Approach to Hierarchical AMS Synthesis, pages 205–225. Springer, 2006.
- [O’Connor 07] I. O’Connor, B. Courtois, K. Chakrabarty, N. Delorme, M. Hampton & J. Hartung. *Heterogeneous Systems on Chip and Systems in Package*. In Design, Automation, and Test in Europe (DATE) conference, pages 1–6, April 2007.
- [OSCI 08] OSCI. *Open SystemC Initiative (OSCI) TLM-2.0 User Manual*. no. June, 2008.
- [OSCI 09] OSCI. *OSCI TLM Working-Group TLM-2.0 Language Reference Manual*. July 2009.
- [OSCI 10] OSCI. *Open SystemC Initiative (OSCI) Standard SystemC-AMS extensions language reference manual*. March 2010.
- [Patel 04] Hiren D. Patel & Sandeep Kumar Shukla. Systemc kernel extensions for heterogeneous system modeling - a framework for multi-moc modeling and simulation. Kluwer, 2004.
- [Patel 05] H.D. Patel & S.K. Shukla. *Towards a heterogeneous simulation kernel for system-level models: a SystemC kernel for synchronous data flow models*. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 24, no. 8, pages 1261–1271, aug. 2005.
- [Paugnat 11] F. Paugnat, K. Morin-Allory & L. Fesquet. *A refinement process for top-down mixed-signal designs thanks to SystemC-AMS*. In New Circuits

- and Systems Conference (NEWCAS), 2011 IEEE 9th International, pages 378–381, june 2011.
- [Pecheux 05] F. Pecheux, C. Lallement & A. Vachoux. *VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems*. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 24, no. 2, pages 204–225, feb. 2005.
- [Pecheux 10] Francois Pecheux & Amr Habib. *Towards high-level executable specifications of heterogeneous systems with systemC-AMS: Application to a manycore PCR-CE lab on chip for DNA sequencing*. In Specification Design Languages (FDL 2010), 2010 Forum on, pages 1–6, sept. 2010.
- [Rafaila 09] Monica Rafaila, Christian Decker, Christoph Grimm, Karsten Einwich, Thomas Markwirth & Georg Pelz. *New Methods for System-level Verification using SystemC-AMS extensions: application to an automotive ECU*. In Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen, 2009.
- [Rafaila 10] Monica Rafaila, Christian Decker, Christoph Grimm, Jerome Kirscher & Georg Pelz. *Design of experiments for reliable operation of electronics in automotive applications*. In Specification Design Languages (FDL 2010), 2010 Forum on, pages 1–6, sept. 2010.
- [Ramanath 02] Rajeev Ramanath, Wesley E. Snyder, Griff L. Bilbro & William A. Sander III. *Demosaicking methods for Bayer color arrays*. J. Electronic Imaging, vol. 11, no. 3, pages 306–315, 2002.
- [Sander 04] I. Sander & A. Jantsch. *System modeling and transformational design refinement in ForSyDe [formal system design]*. Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on, vol. 23, no. 1, pages 17–32, jan. 2004.
- [Schulz 10] Stephan Schulz, Jörg Becker, Thomas Uhle, Karsten Einwich & Sören Sonntag. *Transmitting TLM transactions over analogue wire models*. In Proceedings of the Conference on Design, Automation and Test in Europe, DATE '10, pages 1608–1613, 3001 Leuven, Belgium, Belgium, 2010. European Design and Automation Association.
- [Shcherbakov 11] I. Shcherbakov, C. Weis & N. Wehn. *Bringing C++ productivity to VHDL world: From language definition to a case study*. In Specification and Design Languages (FDL), 2011 Forum on, pages 1–7, sept. 2011.
- [Simeu 00] E. Simeu. *Optimal detector design for on-line testing of linear analog systems*. Hindawi VLSI Design, vol. 11, no. 1, pages 59–74, 2000.
- [Sternhagen 02] J. D. Sternhagen, K. Mitzner, E. Berkenpas, M. Karlgaard, C. E. Wold & D. W. Galipeau. *A direct digital synthesis system for acoustic wave sensors*. IEEE Sensors Journal, vol. 2, no. 4, pages 288–293, August 2002.

- [Study-Group 12] Study-Group. *Homepage of the SystemC AMS Study Group website*. In <http://www.systemc-ams.org/>, Last visited January 2012.
- [Systems 08] Cadence Design Systems & Mentor Graphics Inc. Open verification methodology user's guide – product version 2.0. Sept 2008.
- [Tan 08] Tran Due Tan, S. Roy, Nguyen Phu Thuy & Huu Tue Huynh. *Streamlining the Design of MEMS Devices: An Acceleration Sensor*. Circuits and Systems Magazine, IEEE, vol. 8, no. 1, pages 18–27, quarter 2008.
- [Uhle 10] T. Uhle & K. Einwich. *A SystemCAMS extension for the simulation of non-linear circuits*. In SOC Conference (SOCC), 2010 IEEE International, pages 193–198, sept. 2010.
- [Urbanczyc 02] M. Urbanczyc, Z. Waltar & W. Jakubik. *Interdigital transducer analysis using equivalent PSpice model*. Ultrasonics, vol. 39, no. 8, pages 595–599, June 2002.
- [Vachoux 03] A. Vachoux, C. Grimm & K. Einwich. *SystemC-AMS requirements, design objectives and rationale*. In Design, Automation and Test in Europe Conference and Exhibition, 2003, pages 388–393, 2003.
- [Vachoux 05] A. Vachoux, C. Grimm & K. Einwich. *Extending SystemC to support mixed discrete-continuous system modeling and simulation*. In Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on, pages 5166–5169 Vol. 5, may 2005.
- [Vachoux 06] A. Vachoux. Applications of specification and design languages for socs: selected papers from fdl 2005, chapitre SystemC-WMS: Mixed-Signal Simulation Based on Wave Exchanges. The ChDL series. Springer, 2006.
- [Valdes-Garcia 08] A. Valdes-Garcia, R. Venkatasubramanian, J. Silva-Martinez & E. Sanchez-Sinencio. *A broadband CMOS amplitude detector for on-chip RF measurements*. IEEE Transaction on Instrumentation and Measurement, vol. 57, no. 7, pages 1470–1477, July 2008.
- [Vasilevski 08] M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady & K. Einwich. *Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN*. In Design, Automation and Test in Europe, 2008. DATE '08, pages 134–139, march 2008.
- [Walter 94] É. Walter & L. Pronzato. Identification de modèles paramétriques à partir de données expérimentales. MASSON, Paris, 1994.
- [WASABI 10] WASABI. *Wireless And SystemC-AMS Basic Infrastructure, ANR Programme “Architecture du futur”, URL:<http://www.agence-nationale-recherche.fr/documents/aap/2007/selection/ArchFutur-2007.pdf> last checked Feb 2012, Partners: IEMN, Magillem, STMicroelectronics, Duration: 2007-2010*.

- [Wolf 94] W. H. Wolf. *Hardware-software co-design of embedded systems*. Proceedings of the IEEE, vol. 82, no. 7, pages 967–989, 1994.
- [Xu 09] Tao Xu, H.L. Arriens, R. van Leuken & A. de Graaf. *A precise SystemC-AMS model for Charge Pump Phase Lock Loop with multiphase outputs*. In ASIC, 2009. ASICON '09. IEEE 8th International Conference on, pages 50–53, oct. 2009.
- [Zaidi 10] Yaseen Zaidi, Christoph Grimm & Jan Haase. *On mixed abstraction, languages, and simulation approach to refinement with systemC AMS*. EURASIP J. Embedded Syst., vol. 2010, pages 7:5–7:5, January 2010.
- [Zhu 10] Jun Zhu, Ingo Sander & Axel Jantsch. *HetMoC: Heterogeneous modelling in SystemC*. In Specification Design Languages (FDL 2010), 2010 Forum on, pages 1–6, sept. 2010.

Quatrième partie .

Résumé de la thèse en français

Résumé

Le développement formidable de la technologie utilisée pour la conception de circuits électroniques permet aujourd'hui l'intégration de systèmes de plus en plus complexes. Cette avancée technologique a pu se réaliser grâce à la miniaturisation des dispositifs électroniques qui est aussi liée à l'amélioration de l'efficacité des outils de "conception assistée par ordinateur" (CAO). La conception de systèmes hétérogènes extrêmement complexes tels que les "systèmes sur puce" (SoC) à signaux mixtes, est désormais une réalité. La cohabitation dans les systèmes sur puce d'aujourd'hui de plusieurs domaines physiques tels que mécaniques, chimiques, optiques ou magnétiques justifie l'investissement qui a lieu dans les outils de CAO dédiés à la conception de ces SoCs.

Aujourd'hui, la conception de composants individuels est généralement bien comprise et optimisée grâce à l'utilisation d'une diversité d'outils de CAO et d'automatisation de la conception électronique (EDA), de langages de conception, et de formats de données plus ou moins standard. Ceci implique la mise en œuvre de différents concepts de modélisation et abstraction, de formalismes de description (aussi appelé "modèles de calcul" (MoC)) et de méthodes d'analyse et simulation spécialisées. Il revient au concepteur de combler le fossé qui existe entre les outils et les méthodologies en convertissant manuellement les modèles et en couplant les outils spécialisés pour tel ou tel domaine, ce qui est source d'erreurs et requière beaucoup de temps.

La validation des interactions entre cette grande diversité de composants individuels dans les systèmes récents est d'un intérêt vital pour l'ensemble du système qui doit fonctionner en conformité avec les spécifications demandées par le client. Au début de la phase de conception, l'interaction entre les différents blocs de propriété intellectuelle (IP) intégrés dans le système ne peut être validée que par la mise à disposition d'un modèle dédié à la simulation au niveau système pour chaque bloc. Différents niveaux d'abstraction peuvent être définis lors de la modélisation des IPs. Une abstraction de haut niveau modélise seulement le comportement du système en se contentant d'une précision réduite par rapport au dispositif réel. Des parties analogiques et numériques coexistent généralement dans un même système. Il est alors nécessaire de modéliser les deux parties afin de réaliser une simulation globale de celui-ci. Par ailleurs les parties numériques intègrent des processeurs avec leurs logiciels embarqués. La validation de l'interaction entre les parties analogiques et à signaux mixtes (AMS) avec les parties numériques et le logiciel embarqué devient d'une importance cruciale.

Pour atteindre cet objectif, un environnement de conception et de simulation basée sur le standard IEEE 1666 (SystemC) utilisé pour le numérique s'est récemment affirmé. Cet environnement, proposé par l'OSCI ("Open SystemC Initiative") est appelé SystemC AMS. Il permet de créer et d'affiner un prototype virtuel du système entier à un haut niveau d'abstraction. L'intégration de différents formalismes de description (MoCs) rend possible la description de nombreuses formes de comportement et d'interaction. L'exploration de différentes architectures, l'estimation des performances, la validation de IPs réutilisés ("IP reuse"), la vérification précoce

du développement du logiciel embarqué et son débogage, la vérification des interfaces entre les composants hétérogènes et l'interopérabilité avec d'autres systèmes deviennent alors possibles.

Cette thèse traite de la modélisation de systèmes à la fois hétérogènes car intégrant différents domaines de la physique et à signaux mixtes, numériques et analogiques (AMS). En particulier, le manuscrit présente une étude approfondie de différentes techniques d'extraction de modèles comportementaux à différents niveaux d'abstraction et de précision. Bien que les outils développés pour l'extraction de modèles comportementaux soient principalement basés sur les extensions AMS du standard IEEE 1666 SystemC, la méthodologie peut être aussi appliquée à d'autres langages couramment utilisés pour la description de matériel analogique et mixte (AHDL) tels que VHDL-AMS et Verilog-AMS.

Les techniques d'extraction de modèles peuvent être regroupées en trois branches :

Premièrement, une technique partant d'une description sous forme de schéma au niveau transistor ou d'une netlist d'entrée est étudiée et automatisée afin d'extraire la représentation d'état. D'autres informations souhaitées peuvent aussi être extraites par exemple des fonctions dépendant des tensions ou des courants aux nœuds du circuit. Cette technique peut être utilisée pour extraire l'information sur la consommation en énergie du circuit analysé.

Deuxièmement, des techniques basées sur l'approximation de réponses fréquentielles par ajustement analytique sont explorées, soit pour réduire l'ordre d'un modèle déjà disponible ou bien pour construire un modèle analytique simulable à partir de résultats obtenus avec d'autres outils de CAO de plus bas niveau.

Enfin, les techniques d'identification de modèles paramétriques sont proposées pour l'extraction de modèles dits à "boîte-noire" à partir de données obtenues de façon empirique, de simulations de modèles précis ou de données de mesure. Une bibliothèque preuve-de-concept mise en œuvre en utilisant SystemC AMS démontre l'applicabilité de la méthode par une étude de cas réalisant la minimisation de la consommation d'un amplificateur faible bruit (LNA).

Le caractère mixte de la thèse, universitaire et industriel (CIFRE), en collaboration avec l'entreprise STMicroelectronics, a permis la modélisation d'un capteur d'image CMOS commercial. Cette étude de cas vise la simulation globale d'une plate-forme industrielle pour des applications mobiles décrites au niveau transactionnel, en utilisant le formalisme de modélisation SystemC-TLM ("Transaction Level Modeling"). Pour ce faire, l'interface analogique/numérique entre les MoCs propres à SystemC AMS et les différents styles de codage en SystemC-TLM est étudiée et adaptée aux protocoles TLM propriétaires de STMicroelectronics. Ceci est preuve de l'applicabilité de la méthode et de son insertion dans un flot de conception industriel basé sur SystemC AMS. Les modèles du capteur d'image sont extraits à différents niveaux en utilisant différents modèles de calcul de SystemC AMS. Cette modélisation à différents niveaux montre des gains en temps de simulation impressionnants par rapport aux modèles de plus bas niveau et notamment au modèle en VHDL-AMS qui a servi de référence. La plate-forme virtuelle obtenue permet de calibrer de façon précoce les algorithmes de correction d'image et du logiciel embarqué en augmentant la fiabilité globale du produit. Elle est en cours d'introduction dans le flot industriel standard.

Mots-clés : Flot, méthodologie de conception de systèmes analogiques et signaux mixtes (AMS) ; Modélisation comportementale ; Réutilisation de blocs de propriété intellectuelle ; Modèle de représentation d'état ; Modèle d'ordre réduit ; Modèle de calcul (MoC) ; Système sur puce (SoC) multiphysique ; OSCI SystemC AMS ; VHDL-AMS ; Identification de modèles paramétriques ; Capteur d'image ; Modèle transactionnel (SystemC TLM).

Introduction et état de l'art

Le flot de conception de systèmes multi-domaines et à signaux mixtes de nos jours est réparti entre différentes méthodologies de conception supportées par divers outils de conception assistée par ordinateur (CAO) et d'automatisation de la conception électronique (EDA). La conception de systèmes hétérogènes est toujours un travail très manuel et pas normalisé comme l'est le flot de conception numérique. Une explication intuitive et partielle de ce retard de l'analogique sur le numérique est que les systèmes analogiques abordent une grande variété de phénomènes physiques qui doivent être interprétés et modélisés pour être compris et donc maîtrisés, tandis que les systèmes numériques sont des artefacts artificiels et donc plus facilement formalisables. Les outils pour la synthèse logique et le placement/routage aident à la conception des systèmes numériques alors que la conception hétérogène nécessite une approche multidisciplinaire. Une approche multidisciplinaire implique la définition de plusieurs formalismes de description (aussi appelés modèles de calcul (MoC)), ainsi que des méthodes d'analyse et de simulation. Celles ci sont pourvues par les fournisseurs d'outils de CAO/EDA et se composent de simulateurs spécialisés suivant les différentes disciplines et de langages de description à différents niveaux d'abstraction. Afin d'en faciliter l'utilisation, ces outils sont généralement intégrés dans un environnement de travail comprenant notamment une saisie de schéma, un "netlister", un visualisateur de courbes etc.

Par exemple, la conception d'un composant électromécanique (MEMS) tel qu'un accéléromètre à trois axes [Tan 08] exige :

- L'optimisation et la caractérisation du micro-résonateur mécanique et (séparément) la distribution du champ électrostatique de la structure en peigne, qui sert au contrôle et détection des mouvements de la structure flexible. Cela se fait avec l'aide d'une analyse aux éléments finis (par exemple ANSYSTM, CoventorTM).
- La simulation de l'ensemble du système au niveau du circuit, en tenant compte du couplage entre le domaine mécanique et électrostatique dans le transducteur MEMS et les échanges d'informations entre les circuits analogiques et numériques de contrôle et détection. Pour ce faire, des simulations comportementales à l'aide de langages de modélisation comme VHDL-AMS sont employés.
- L'implémentation (dessin des masques ou "layout") de la structure mécanique et des circuits électroniques. Ceci est réalisé avec l'aide d'outils de dessin des circuits intégrés.

D'une part, quand une **simulation précise** du système hétérogène est nécessaire, une co-simulation, définie comme une simulation effectuée par le lancement de deux noyaux de simulation différents, permet l'interfaçage de la variété d'outils spécifiques aux domaines et des simulateurs.

Toutefois, ce processus est loin d'être transparent et de réalisation simple. Ce n'est pas la priorité des fournisseurs d'outils de CAO de fournir un interfaçage facile pour d'autres

simulateurs, spécialement s'il s'agit d'un concurrent plus avancé dans d'autres domaines du génie ou si leur suite d'outils EDA dispose déjà d'une capacité similaire mais non suffisante. Les concepteurs sont obligés de combler le fossé entre les outils et les méthodologies en convertissant manuellement les modèles et en jonglant avec les différents outils. Bien entendu, ces bricolages sont source d'erreurs, limités à deux moteurs de simulation, et avec des résultats incertains dûs à un manque d'outils d'analyse.

D'autre part, la capacité de réaliser une **vérification globale du système** au début du flot de conception prend une extrême importance au vu de la complexité croissante des systèmes. Tant pour les composants numériques que pour les composants mixtes, des prototypes virtuels de ces composants permettront la vérification du système, tant en ouvrant la porte à de nombreuses possibilités telles que l'exploration d'architecture, l'estimation des performances, validation de IPs réutilisées (IP reuse), la vérification de l'interfaçage entre les domaines du RF, analogiques et numériques, la vérification précoce du développement du logiciel embarqué avec son débogage, la vérification de l'interopérabilité avec d'autres systèmes, et l'évaluation de l'impact de nouvelles générations de technologies en avance de phase. La tâche de la modélisation devient donc cruciale dans le flot de conception de systèmes sur puce (SoC) et les langages de modélisation doivent être choisis de manière appropriée, essentiellement en se basant sur le type de sous-systèmes envisagés, et le niveau de précision/abstraction souhaité.

Du côté du **numérique**, le flot de conception à approche descendante qui part d'une description "Register Transfer Level" (RTL) et va jusqu'aux portes logiques est bien établi et utilisé par la majorité des concepteurs du monde numérique. Ce flot utilise les langages de description de matériel VHDL [IEEE 09c] et Verilog [IEEE 04]. L'importance du logiciel embarqué dans les systèmes récents a conduit à la nécessité de modéliser le logiciel lui-même, donc pousser la recherche vers des langages de conception au niveau système. SystemC [IEEE 09a] et SystemVerilog [IEEE 09b] sont aujourd'hui largement utilisés pour ce niveau. SystemC est un standard IEEE (connu sous le nom de IEEE 1666-2005) depuis 2005. Basé sur le langage C++, il est promu par l'Open SystemC Initiative (OSCI) qui a fusionné en décembre 2011 avec ACCELLERA devenant *Accellera Systems Initiative* [Initiative 12]). SystemC permet de décrire des systèmes à un niveau plus abstrait (mais aussi au niveau RTL) en se basant sur un noyau piloté par des événements discrets, ceci donne la possibilité de modéliser à la fois le matériel et les composants logiciels. Pendant les dernières années, le langage C++ s'est affirmé comme la solution plus adoptée pour la programmation de processeurs embarqués. Par conséquence un tel langage de description du matériel basé sur le langage C++ permet d'avoir un environnement unifié de conception, c'est à dire du matériel et de son logiciel associé, ce qui n'est pas le cas pour SystemVerilog. En 2008 OSCI a également publié le standard de modélisation niveau transactionnel comme extension du noyau SystemC, TLM-2.0 [OSCI 09] visant à permettre l'interopérabilité des modèles en SystemC et leurs réutilisation au niveau transactionnel, en fournissant un cadre essentiel pour l'analyse de l'architecture au niveau système (ESL), mais aussi le développement du logiciel, l'analyse des performances du logiciel et la vérification du matériel.

En ce qui concerne la conception simultanée de sous systèmes à **signaux analogiques et mixtes**, comme les parties RF, les MEMS, les conversions de l'analogique au numérique (ADC) ou bien du numérique à l'analogique (DAC), des extensions de langages de modélisation existantes ont fait leur apparition. VHDL et Verilog ont produit des extensions pour l'AMS nommés VHDL-AMS (standard IEEE 1076-2008) [IEEE 07] et Verilog-AMS [Accellera 09]. Ces langages sont orientés pour la description de matériel et manquent de formalismes suffisants pour

décrire les systèmes AMS à un niveau comportemental ou fonctionnel. En outre, ils ne disposent pas de capacité de modélisation des logiciels embarqués et les simulateurs disponibles sont trop lents pour envisager de simuler la complexité d'un SoC hétérogène de pointe. D'autres outils de haut niveau comme Matlab/Simulink offrent une alternative pour la modélisation fonctionnelle et comportementale, mais ils manquent de lien direct avec le flot de conception du matériel et ils n'offrent aucun moyen de simuler le logiciel. Il a été par conséquent logique d'étudier et de préparer des extensions AMS du noyau SystemC puisque ce dernier est largement accepté et adopté comme langage polyvalent basé sur C++ par les concepteurs de systèmes, les ingénieurs du logiciel, et les concepteurs de matériel. Le groupe de travail AMS (AMSWG) de l'OSCI mis en place dès 2006 a concentré ses efforts sur la définition des paradigmes de modélisation, de la syntaxe et de la couche d'interface du noyau SystemC. Ces efforts se sont concrétisés avec la soumission pour la standardisation des extensions AMS de SystemC [OSCI 10, Vachoux 05] appelés SystemC AMS 1.0 (2010) auprès de l'OSCI. Ces extensions offrent maintenant la possibilité de modéliser des systèmes complexes hétérogènes à l'aide de différents modèles de calcul (MoC).

La modélisation comportementale des parties AMS, tels que les composants RF ou MEMS est de plus en plus d'actualité dans l'industrie dans le cadre du processus de conception de systèmes intégrés, car elle permet de simuler la totalité d'un système complexe et hétérogène. Les HDL standards pour signaux mixtes mentionnées ci-dessus sont mis en œuvre par plusieurs simulateurs commerciaux et open source. Un modèle comportemental permet de décrire le comportement d'un composant comme une fonction entre entrées et sorties qui intègre les principales fonctionnalités de l'implémentation réelle, mais qui n'exige pas une description complète de tous les détails de l'implémentation. Le but principal est de vérifier le fonctionnement correct de l'ensemble du système dans lequel le dispositif est intégré en un temps de simulation acceptable.

Il est habituel de considérer le processus de conception d'un produit technologique comme décrit par le modèle en forme de "V" dans la Figure 1. Le côté gauche du "V" montre une approche de conception descendante à partir de l'analyse du problème jusqu'à la mise en œuvre en passant par les spécifications du système. L'autre côté décrit la vérification comme ascendante, depuis le test après fabrication jusqu'à l'insertion dans le système final, en passant par l'extraction des caractéristiques réelles (parasites, délais ...), la rétro annotation des modèles, les résimulations.

La **modélisation comportementale** permet de simplifier considérablement le processus de conception dans les deux côtés du modèle en "V". Cependant, un mauvais choix de description ou de précision pour un modèle donné peut également conduire à une simplification excessive, et invalider l'ensemble du processus, par exemple en présence de phénomènes fortement non linéaires.

Lors de la conception descendante, le but n'est pas de concevoir dans l'acception classique du terme, qui à partir des spécifications du système, permet de descendre à la synthèse automatisée et à l'implémentation du système, bien que certaines études soient en cours dans ce domaine [der Plas 02]. L'intention est bien de profiter de la modélisation du comportement des composants numériques et AMS du système afin d'aider les concepteurs à effectuer l'exploration architecturale et de décliner les spécifications de performance de haut niveau sur les différents sous blocs tout en préservant le compromis entre les performances et les coûts de la mise en œuvre (par exemple la vitesse ou la consommation).

Lors de la **vérification ascendante** les mêmes modèles comportementaux peuvent être réutilisés après rétro annotation. La précision du modèle est améliorée en l'enrichissant de détails provenant de l'implémentation de bas niveau tels que l'extraction des effets parasites ou des délais après implémentation ou bien d'une caractérisation effectuée à travers des mesures sur un prototype physique. Evidemment, les détails se réduisent lorsque le niveau d'abstraction remonte progressivement.

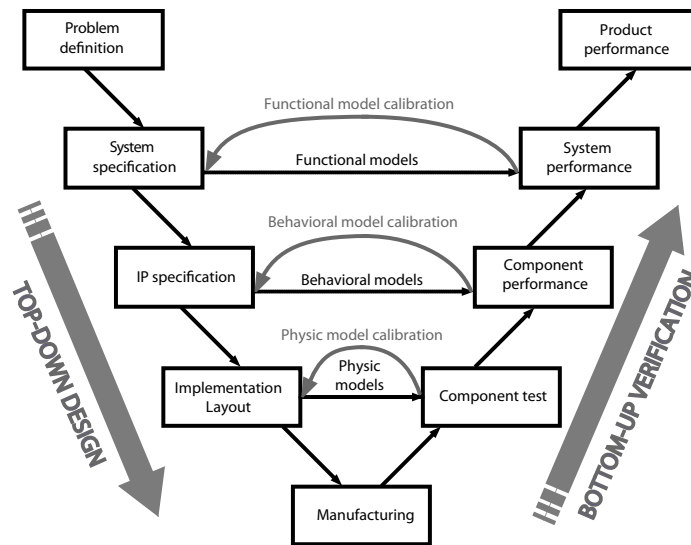


Figure 1.: Modèle en “V” du processus de conception d’un produit technologique.

1. Contributions à la recherche

Lorsqu’on traite de la diversité des natures et des domaines de la physique impliqués dans un SoC hétérogène, il est difficile de comprendre comment les comportements analogiques peuvent être saisis et modélisés pour représenter le fonctionnement des sous-systèmes/composants ciblés. Les formalismes/paradigmes de modélisation disponibles changent selon le langage de modélisation ou l’outil. Différentes techniques ont été étudiées pour différents types de modélisation comportementale et répertoriées en fonction de leur adaptation aux besoins. Les composants analogiques et à signaux mixtes (AMS) peuvent être connus avec différentes connaissances. Il est généralement admis de les regrouper en deux cas comme suit :

- La structure du dispositif est connue et les lois de la physique qui définissent son fonctionnement sont maîtrisées. Une équation analytique d’entrées/sorties, éventuellement impliquant des variables d’état, peut donc être déterminée. C’est ce qu’on appelle “Modélisation des connaissances” (branche violette à gauche dans la Figure 2). La possibilité de disposer d’une connaissance exhaustive du système est aujourd’hui de moins en moins probable en raison de la complexité croissante et de l’interaction non négligeable des dispositifs entre eux.
- Les lois de la physique qui gouvernent le comportement du dispositif ne sont pas *a priori* connues et seules les données expérimentales sont généralement plus facilement disponibles.

Le dispositif est soit physiquement disponible (branche verte de droite dans la Figure 2) (boîte noire, livrée sous la forme d'un prototype ou d'un modèle de l'IP précompilé et simulable), soit une description fine de celui-ci est connue (branche bleu au centre de la Figure 2). Cette description fine peut être issue d'une analyse structurale effectuée par des simulateurs/outils *ad hoc* (FEM par exemple) qui sont spécifiques au domaine. Dans les deux cas, un modèle mathématique doit être construit pour décrire le comportement du dispositif afin d'effectuer des simulations.

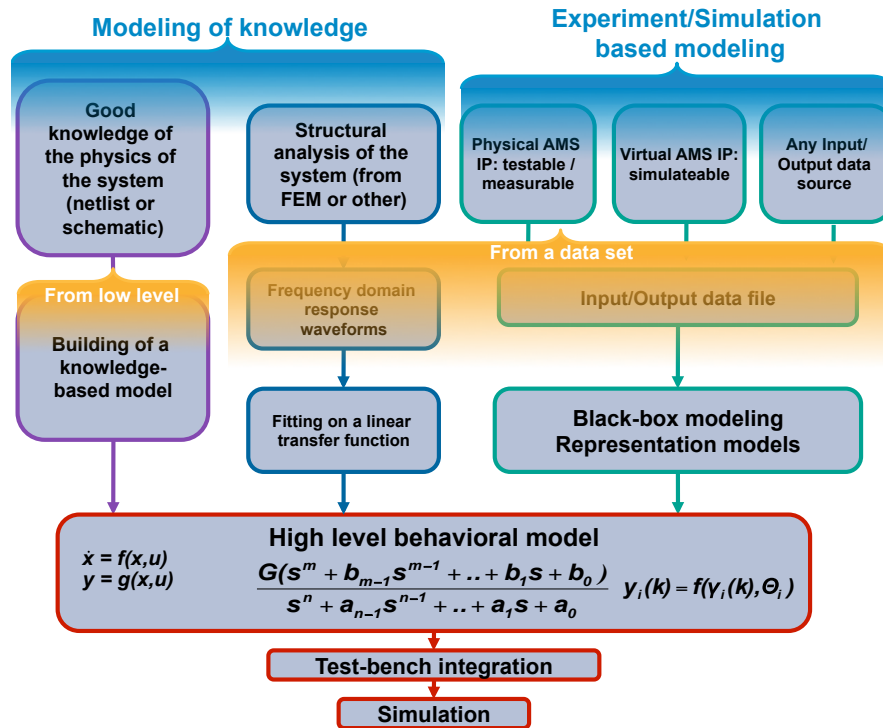


Figure 2.: Types de modélisation comportementale pour des dispositifs hétérogènes et à signaux mixtes.

Différentes techniques de modélisation ont été analysées et automatisées pour obtenir des modèles comportementaux à différents niveaux d'abstraction et de complexité. Un accent particulier a été mis sur l'utilisation de l'extension AMS de SystemC. Ces techniques de modélisation à ordre réduit sont étudiées et mises en œuvre pour la conception descendante et la vérification ascendante, et sont basées sur le raffinement successif des modèles. Différentes études de cas et des modèles de calcul offerts par SystemC AMS (mais aussi Verilog-A et VHDL-AMS), sont montrées pour prouver l'efficacité de la méthodologie proposée. L'interfaçage analogique/numérique est également présenté, notamment pour des applications ciblées, et des solutions spécifiques mises en œuvre pour le standard SystemC TLM-2.0.

En particulier, profitant de la nature industrielle de la thèse, la méthodologie pour les composants AMS est appliquée à des modèles raffinés d'un capteur vidéo CMOS produit par STMicroelectronics. Le capteur AMS lui-même communique en utilisant des protocoles de communication au niveau transactionnel, propriété de STMicroelectronics. Il est montré et prouvé comment la méthodologie aide à réduire le temps de mise sur le marché par anticipation

sur les phases de développement (notamment le logiciel embarqué) et par amélioration de la fiabilité des produits commercialisé (déverminage du logiciel).

2. Structure de la thèse

Le premier chapitre introduit le cadre et les motivations de ce travail. Le deuxième chapitre donne un aperçu de l'environnement SystemC basé sur C++ avec ses extensions AMS et TLM. Le troisième chapitre décrit ma contribution à l'étude des techniques de modélisation pour l'abstraction de modèles à haut niveau à partir du bas niveau, aussi bien pour la conception que pour le test. Le chapitre quatre montre une application de la technique d'*identification de système* à un amplificateur faible bruit (LNA) afin d'en optimiser la consommation au travers d'une boucle de contrôle fermée. Ce chapitre montre également l'application de la modélisation comportementale à la conception d'un capteur chimique basé sur des ondes acoustiques de surface (SAW). Le chapitre cinq traite de l'étude d'un capteur d'image CMOS industriel. Le chapitre six conclut mon travail et propose des perspectives de développements futurs.

Environnement de modélisation et simulation SystemC

Implémenté comme une extension du langage C++, **SystemC** (IEEE 1666 de décembre 2005) fournit une bibliothèque de classes pour la modélisation, dont le modèle d'exécution est adapté à la simulation de SoCs et, potentiellement, de tout système matériel numérique complexe, avec ou sans processeurs et logiciel. La modélisation SystemC couvre les concepts de comportements concurrents, de temps du système simulé et de types de données adaptés à la description du matériel, ainsi que la modélisation de structures hiérarchiques. Les modèles en SystemC sont simulables sur station de travail de type PC ou autre, avec des simulateurs open source ou commerciaux qui augmentent encore la productivité.

3. SystemC AMS et ses modèles de calcul

Les extensions AMS de SystemC ajoutent d'autres méthodes d'abstraction pour la modélisation à haut niveau de composants AMS au noyau SystemC. Comme le montre la Figure 3, les niveaux d'abstraction font la distinction entre des comportements en temps discret et en temps continu et des comportements conservatifs et non conservatifs.

Les extensions AMS de SystemC définissent également les formalismes de modélisation nécessaires pour soutenir la modélisation comportementale à différents niveaux d'abstraction (Figure 3). Ces formalismes de modélisation sont mis en œuvre en utilisant différents modèles de calcul (MoC) : flux de données temporisé (Timed Data Flow : TDF), flux de signal linéaire (Linear Signal Flow : LSF), et réseau électrique linéaire (Electrical Linear Network : ELN). Les trois MoCs sont ci-après brièvement introduits, de plus amples détails peuvent être trouvés dans [Barnasconi 10] :

- **Flux de données temporisé (TDF)** : la sémantique d'exécution du MoC TDF introduit une modélisation à temps discret, de plus le TDF évite le surcoût en temps de simulation donné par l'ordonnancement dynamique imposée par le noyau à événements discrets de SystemC. La simulation est ainsi accélérée en définissant une planification d'exécution statique (static schedule). Cette planification est calculée avant le début de la simulation, et permet d'exécuter les fonctions de traitement de chaque module TDF (méthode **processing**) selon la direction du flux de données. Les échantillons en temps discret des signaux se propagent à travers les modules TDF. Les données de ces signaux peuvent être de n'importe quel type C++. Si, par exemple, un type de valeur tel que le réel en double est utilisé, le signal TDF peut être associé à une tension ou un courant. Les valeurs complexes peuvent être utilisées pour représenter un signal de bande de base équivalente dans le cas du RF.

Model of Computation (MoC)	Imposed abstractions			
	Behavior	Structure	Communication	Time/Frequency
Timed Data Flow (TDF)	Algorithmic descriptions, transfer functions	Functional blocks (non-conservative system)	Sequence of samples of arbitrary type	(Over)sampling, baseband modeling
Linear Signal Flow (LSF)	Linear functional descriptions	Structural representation of linear equations (non-conservative system)	Directed signals, continuous value	No abstraction (continuous time)
Electrical Linear Networks (ELN)	Macro modeling with linear primitives and ideal switches	Simplified network (conservative system)	No abstraction (physical quantities)	No abstraction (continuous time)

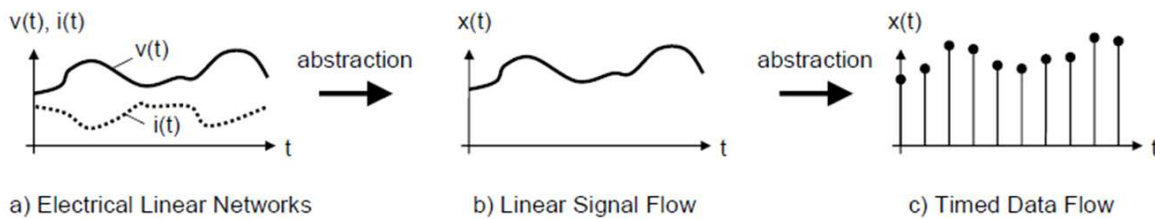


Figure 3.: Abstractions par rapport aux modèles de calcul de SystemC AMS.

- **Flux de signal linéaire (LSF)** : le formalisme LSF prend en charge la modélisation de comportements en temps continu en offrant un ensemble de modules primitifs tels que l'additionneur, le multiplicateur, l'intégrateur ou le module de retard. Un modèle LSF est une connexion de ces modules de base liées par des signaux à valeurs réelles qui peuvent représenter des quantités réelles. Un modèle LSF définit un système d'équations linéaires qui est résolu par un solveur linéaire d'équations différentielles algébriques (DAE).
- **Réseau électrique linéaire (ELN)** : la modélisation de réseaux électriques est supportée par l'instanciation d'éléments de base prédéfinis tels que des résistances ou de condensateurs, qui sont utilisés comme de macro modèles pour décrire les relations en temps continu entre les tensions et les courants. Un ensemble restreint de primitives linéaires et des commutateurs est disponible pour modéliser le comportement conservatif de l'énergie électrique.

4. SystemC TLM

TLM (Transaction Level Modeling), bâti sur SystemC, standardise la modélisation à un niveau d'abstraction plus élevé que le RTL, en fournissant des modèles de communication de structures de données complètes ("transactions") entre blocs, IPs ou circuits complexes, au lieu des signaux habituels. Le standard TLM de l'OSCI [OSCI 09] permet ainsi à des concepteurs matériels ou développeurs logiciels de modéliser facilement, et simuler à grande vitesse, des blocs, IPs, circuits SoC, ou cartes, à l'aide de modules SystemC communiquant par des transactions.

Contribution à la modélisation de haut niveau de systèmes hétérogènes

5. Introduction et regroupement à partir des connaissances de départ

Une enquête sur les techniques de modélisation de circuits qui abstraient des modèles d'ordre réduit ou de haut niveau à partir d'une description de bas niveau est donnée dans cette section.

L'approche que nous utilisons est liée aux scénarios de modélisation que nous voulons aborder. Les scénarios sont présentés ci-après et classés par le type et niveaux de **connaissances de départ** du système. Deux catégories de systèmes sont d'abord identifiées :

- Les lois de la physique qui définissent le comportement du dispositif sont connues (voir le regroupement bleu “modeling of knowledge” en Figure 2).
 - **Modélisation des connaissances.** La structure du dispositif est connue et les lois de la physique qui régissent le comportement du dispositif sont maîtrisées. La représentation au niveau transistor d'un système analogique est considérée comme faisant partie de cette catégorie, puisque les lois qui régissent les dispositifs ont été exhaustivement caractérisées par le fabricant de silicium du point de vue du procédé technologique (branche violette en Figure 2).
 - **Connaissances structurelles.** Si le composant ne peut pas être décomposé en blocs élémentaires reliés entre eux pour former une netlist ou d'un schéma au niveau transistor, l'analyse de l'appareil cible est généralement effectuée au moyen d'outils permettant de considérer la structure du dispositif dans tous ses éléments. Ceci est particulièrement vrai dans le cas des microsystèmes électromécaniques (MEMS), où une analyse structurelle du système peut être effectuée par le biais d'outils spécifiques basés sur l'analyse aux éléments finis (FEA). L'analyse par ce type d'outils fournit généralement des courbes de réponse des entrées-sorties dans le domaine fréquentiel et une linéarisation est considérée autour d'un point de repos. À côté des courbes de réponse fréquentielles, des réponses non linéaires peuvent aussi bien être obtenues tels que l'atténuation de l'amplitude de vibration d'une micro poutre mécanique. Comme conséquence, une description précise dans le domaine fréquentiel est obtenue à partir d'une analyse structurelle effectuée par des simulateurs/outils spécifiques au domaine (FEM par exemple), ce flot est représenté par la branche bleu en Figure 2.
- Les lois de la physique qui définissent le comportement de l'appareil ne sont pas connues *a priori*.

- **Modélisation basée sur des expériences ou des simulations.** Le dispositif est disponible et des données expérimentales ou de simulation sont disponibles. Le dispositif est disponible soit sous la forme d'un prototype matériel du IP (des expériences peuvent ainsi être faites), ou sous la forme d'un modèle précompilé simulable, dans les deux cas, il est considéré comme une boîte noire avec des entrées contrôlables et des sorties observables (voir regroupement bleu "Experiment/Simulation-based modeling" en Figure 2).

Les trois cas introduits ont été triés par le type de source et de connaissance de départ de l'utilisateur/concepteur. Une deuxième phase consiste en analyser les dispositifs et profiter de la description disponible afin de construire un modèle comportemental. A cet effet, les trois flots montrés sur la Figure 2 doivent être différemment regroupés. Dans tous les cas (ci-après rementionnés et retriés) un **modèle mathématique** doit être construit pour décrire le comportement du composant afin d'en effectuer des simulations.

- Une description du système représentant sa netlist ou son schéma est disponible (voir regroupement orange "From low level" de la Figure 2).
 - **Modélisation des connaissances.** La connaissance complète du système permet de déterminer les équations analytiques d'entrée/sortie dans le domaine temporel, impliquant éventuellement des variables d'état. Néanmoins, la possibilité d'avoir une connaissance exhaustive du système est aujourd'hui de moins en moins probable en raison de la complexité croissante et de l'interaction entre les dispositifs. En outre, si le modèle analytique est trop complexe en terme d'effort de calcul, sa complexité peut être réduite par le biais de techniques de réduction de l'ordre du modèle (MOR) connues dans la littérature. Une enquête est disponible dans [Mantooth 03] montrant différentes techniques de modélisation comportementale de circuits analogiques pour des systèmes linéaires invariants dans le temps (LTI), linéaires variants dans le temps (LTV)/linéaires périodiquement variants dans le temps (LPTV) et non linéaires (NL).
- Un fichier de données décrivant la relation entrée/sortie est disponible (voir regroupement orange "From a data set" en Figure 2).
 - **Connaissances structurelles.** Une analyse structurelle peut conduire à une courbe de réponse d'entrée-sortie dans le domaine fréquentiel composée d'un ensemble de points, donc sans une forme analytique. Cet ensemble de données pourrait être la seule information dont on dispose pour modéliser le système ou un modèle de faible complexité pourrait être requis. Dans les deux cas, une approximation de l'ensemble de données peut être effectuée à l'aide d'un ajustement d'une équation au degré de complexité souhaité et couvrant la plage des fréquences d'intérêt.
 - **Modélisation basée sur des expériences ou des simulations.** La possibilité de contrôler la/les entrée/entrées est un degré de liberté suffisant à permettre d'identifier le système et d'associer une équation mathématique, à condition de pouvoir récupérer sa sortie. Les techniques d'identification de systèmes utilisent des méthodes statistiques pour construire des modèles mathématiques paramétriques de systèmes dynamiques à partir de données mesurées. L'identification de systèmes est particulièrement utilisée dans le domaine du contrôle automatique et de l'ingénierie

de traitement du signal. Le principe de base est de stimuler à la fois le modèle et le système réel et de minimiser l'erreur entre les sorties en identifiant les paramètres d'une équation de départ (structure du modèle).

Nos contributions à la recherche adressent tous les trois axes avec des approches différentes. Toutes les approches ont pour but d'identifier automatiquement les données source disponibles et de définir une méthodologie pour créer facilement des modèles comportementaux (décrites en utilisant un langage AHDL) à différents niveaux de détail et de lancer la simulation globale (dans un *banc de test virtuel*). Selon le niveau d'abstraction requis pour le modèle, les trois flots de modélisation peuvent être réglés pour le niveau souhaité, un compromis entre la précision du modèle et sa rapidité/complexité doit être assumé et trouvé. Plus précisément et de manière générale, le fonctionnement du dispositif analogique est conditionné par l'environnement opérationnel dans lequel il est inséré. Il est essentiel que la méthodologie de modélisation se concentre sur deux aspects essentiels, qui sont : la précision du modèle dans sa **région de fonctionnement** et la **fonctionnalité** particulière que le modèle est destiné à capter et à reproduire.

Dans les sections suivantes, les trois axes sont abordés séparément, et les travaux effectués pour chaque axe sont détaillés ainsi que les études de cas correspondantes qui valident les flots et les concepts.

6. Contribution à l'élaboration de modèles de haut niveau basés sur la connaissance à partir d'une netlist

L'outil que nous proposons permet d'accéder à l'information sur la consommation de puissance pour des modèles décrits au niveau comportemental. Pour le moment, l'outil est limité à des composants analogiques linéaires, même si la démarche est générique, et vise à une modélisation basée sur SystemC AMS. Une première possibilité est de simuler le circuit à l'aide du MoC ELN. Ce MoC offre la possibilité de modéliser et de simuler des sous-systèmes analogiques linéaires en gardant l'aspect conservatif de l'énergie. Il est possible d'accéder à la valeur instantanée du courant d'alimentation et de le remonter jusqu'au niveau d'abstraction plus élevé offert par le MoC TDF à l'aide de convertisseurs de signaux d'ELN à TDF. Cette solution permet d'atteindre un niveau élevé de détails, mais la simulation serait lourde en temps de calcul, en particulier dans le cas de systèmes très complexes. Pour ces raisons, un niveau d'abstraction plus élevé que celui offert par le MoC ELN est nécessaire. D'autre part, concernant les outils de modélisation et de simulation AMS ne donne aucune information sur la consommation des appareils à un haut niveau d'abstraction (SystemC AMS TDF MoC). Cette information est disponible uniquement lorsque un choix d'implémentation est fait, et un modèle de bas niveau de description est disponible (modèle ELN).

L'idée proposée est d'extraire automatiquement à la fois un modèle comportemental et l'information pertinente la consommation d'énergie à partir d'une description de bas niveau (netlist) d'un circuit linéaire. Ces informations sont ainsi remontées dans un modèle de haut niveau afin que les informations de puissance soient propagées lors de la simulation. Ainsi, à chaque cycle de simulation, il sera possible d'extraire les courants d'alimentation électrique associés à chaque composante du système et d'utiliser le résultat afin d'estimer la consommation d'énergie du système global.

7. Contribution à l'élaboration de modèles basés sur la connaissance à partir de données de simulation

7.1. Macro modélisation de composants statiques pour la simulation en SystemC AMS

En ce qui concerne les blocs statiques, où la sortie au moment “t” ne dépend que de la valeur des entrées en même temps “t” comme dans la fonction $Y(t) = f(X(t))$ où “f” a une forme arbitraire, souvent non-linéaire. Par conséquent, il n’y a pas de mémoire impliqué. La fonction “f” pourrait avoir des fortes non-linéarités et dans certains cas, l’expression analytique de “f” n’est pas connue, donc seulement un ensemble de points est disponible ou une représentation graphique. Dans ces cas, la fonction “f” peut être approximé par une équation analytique simple obtenue au moyen de différentes méthodes comme l’interpolation, l’approximation par ajustement (fitting), le développement de Taylor. La Figure 4 montre le flux utilisé pour des composants analogiques statiques non linéaires, ce qui pourrait également être appliqué au cas trivial des comportements statiques linéaires. La dernière colonne de droite fait référence explicitement à un modèle décrit en SystemC AMS.

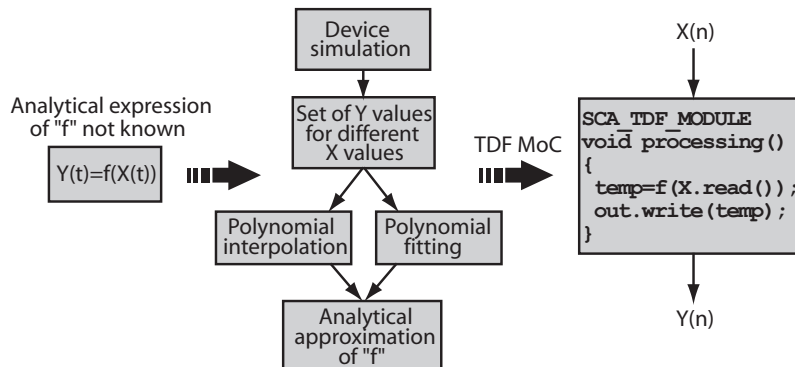


Figure 4.: Macro modélisation de composants non linéaires statiques en SystemC AMS.

7.2. Macro modélisation de composants dynamiques pour la simulation en SystemC AMS

Pour les blocs dynamiques, les valeurs des sorties dépendent des valeurs des entrées à l’instant courant, mais aussi à aux instants précédents. Ce type de comportement peut être modélisé en SystemC AMS en utilisant le MoC TDF ou le MoC ELN. La modélisation des non linéarités de blocs dynamiques n’est possible qu’en utilisant une linéarisation autour d’un point de fonctionnement. Le MoC ELN est en outre limité à l’utilisation de blocs de construction de base disponibles dans les bibliothèques. D’autre part, le MoC TDF peut être utilisé pour représenter tout comportement linéaire en termes de leur fonction de transfert ou de leur représentation d’état. L’approche illustré dans la Figure 5 et utilise le MoC TDF. Si la fonction de transfert de Laplace du dispositif est connue, celle-ci peut être instanciée en SystemC AMS à l’aide d’une représentation polynomiale ou sous forme de pôles et zéros.

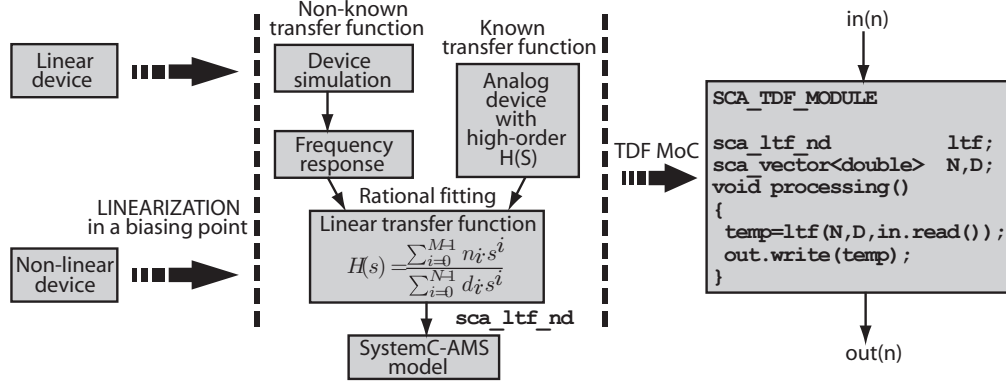


Figure 5.: Macro modélisation de composants dynamiques en SystemC AMS au moyen d'une approximation linéaire.

8. Contribution à l'élaboration de modèles boîte noire à partir de données empiriques

Une extension des bibliothèques SystemC AMS a été développée. Cette bibliothèque permet d'automatiser le flot de notre scénario de modélisation, à partir d'un fichier de données d'ondes d'entrée/sortie échantillonnées jusqu'à la construction et instanciation d'un modèle SystemC AMS adapté du système identifié.

Figure 6 montre en trois axes la méthodologie à différents niveaux de détails. Sur la gauche se trouve le concept, nous partons de la simulation d'un modèle de bas niveau jusqu'à la construction et instanciation d'un modèle AutoRégressive à variable exogène (ARX). Au centre de la figure se trouve un scénario possible dans lequel les résultats de formes d'ondes d'une simulation transitoire au niveau du transistor sont utilisés pour obtenir le vecteur de paramètres Θ du modèle ARX. Sur la droite le flot est mieux expliqué avec des valeurs.

9. Application des techniques d'identification de système à la commande en boucle fermée pour l'optimisation de consommation de puissance. Cas d'étude d'un LNA

Une nouvelle approche pour réduire la consommation d'énergie dans les dispositifs RF est ici décrite. La méthode est basée sur l'adaptation de la tension d'alimentation par le biais d'une stratégie de contrôle logique. Cette stratégie s'appuie sur des capteurs embarqués, sur l'identification de paramètres en temps réel et sur l'estimation des performances. Des économies d'énergie importantes ont été démontrées au niveau transistor pour un LNA RF avec différents modes de performance. L'algorithme de contrôle peut garantir les spécifications requises pour chaque mode de performance malgré les écarts paramétriques en raison du processus de fabrication ou du mécanisme de vieillissement.

Dans le cas d'étude d'un amplificateur faible bruit (LNA) des simulations en Verilog-A/Cadence-Spectre ont permis de valider l'économie d'énergie apportée par la méthodologie à

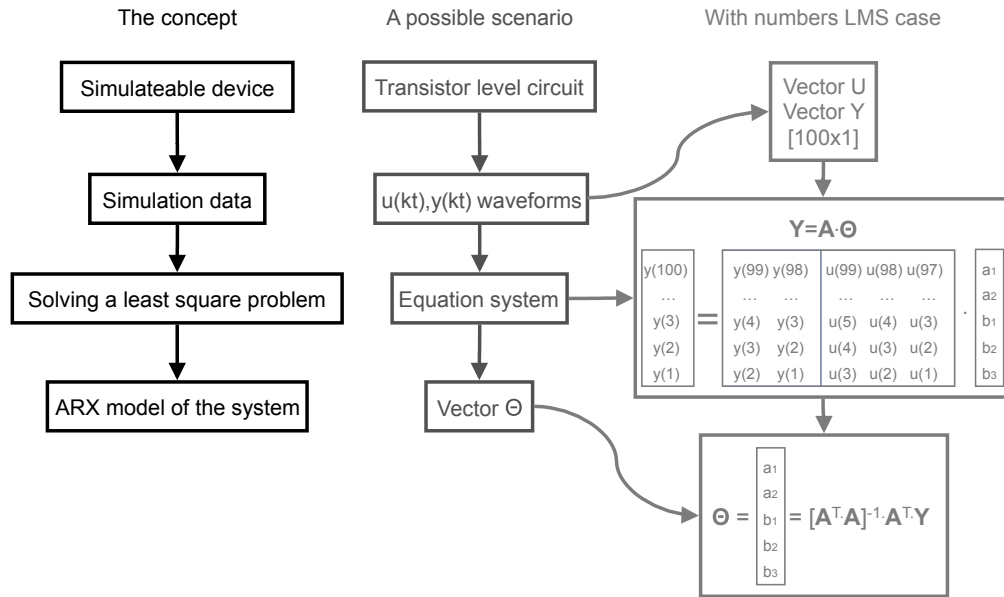


Figure 6.: Le concept de l'extension de SystemC AMS pour l'identification de systèmes.

bas niveau (niveau transistor pour le LNA). D'un point de vue plus abstrait, si la description matérielle de la partie numérique du système doit être simulée, une solution prometteuse serait de décrire la CUC analogique au moyen des MoCs de SystemC AMS et de mapper les algorithmes numériques de contrôle et identification récursive sur un modèle SystemC d'un simulateur de jeu d'instructions (Instruction Set Simulator : ISS). Une telle solution permettrait de décrire avec précision le temps d'exécution de l'algorithme en fonction de la quantité de ressources matérielles (mémoires ou fréquence d'horloge du processeur). Cela permettrait une exploration d'architecture au niveau système afin de mettre en œuvre la méthodologie de commande adaptative présentée.

10. Cas d'étude de la conception de l'interface microélectronique pour un capteur chimique SAW

La méthodologie pour la *construction d'un macro-modèle à partir des réponses en fréquence* (branche bleue) est appliquée à un capteur chimique basé sur des ondes acoustiques de surface (SAW). Le capteur a été modélisé, simulé avec son interface microélectronique de frontend et fabriqué. Deux niveaux de modélisation de l'interface microélectronique du capteur basée sur une boucle à verrouillage de phase (PLL) ont été analysés. Le premier consiste en une simulation Verilog-A/Cadence-Spectre, cette simulation a permis la conception au niveau du transistor et du layout de la PLL, et la fabrication de puces de test. Pourtant ce niveau d'abstraction ne permet pas d'envisager la simulation d'un nœud complet d'un réseau de capteurs chimiques sans fils (WSN). Le deuxième niveau de consiste en modéliser l'interface du capteur chimique à l'aide de SystemC AMS/SystemC. Les modèles ont été réalisés à partir des résultats de la simulation de la réalisation au niveau du transistor montré dans nos précédents travaux [Cenni 09a]. Il a été montré qu'une modélisation SystemC AMS/SystemC pourrait reproduire assez précisément

les simulations au niveau transistor si on choisi une valeur assez petite du pas de simulation temporel. Le concept clé est que l'analyse au niveau transistor conduit à la conception et fabrication de la puce d'interface du capteur chimique tandis que l'analyse SystemC AMS vise plutôt à une étude de faisabilité au niveau du système.

11. Conclusions

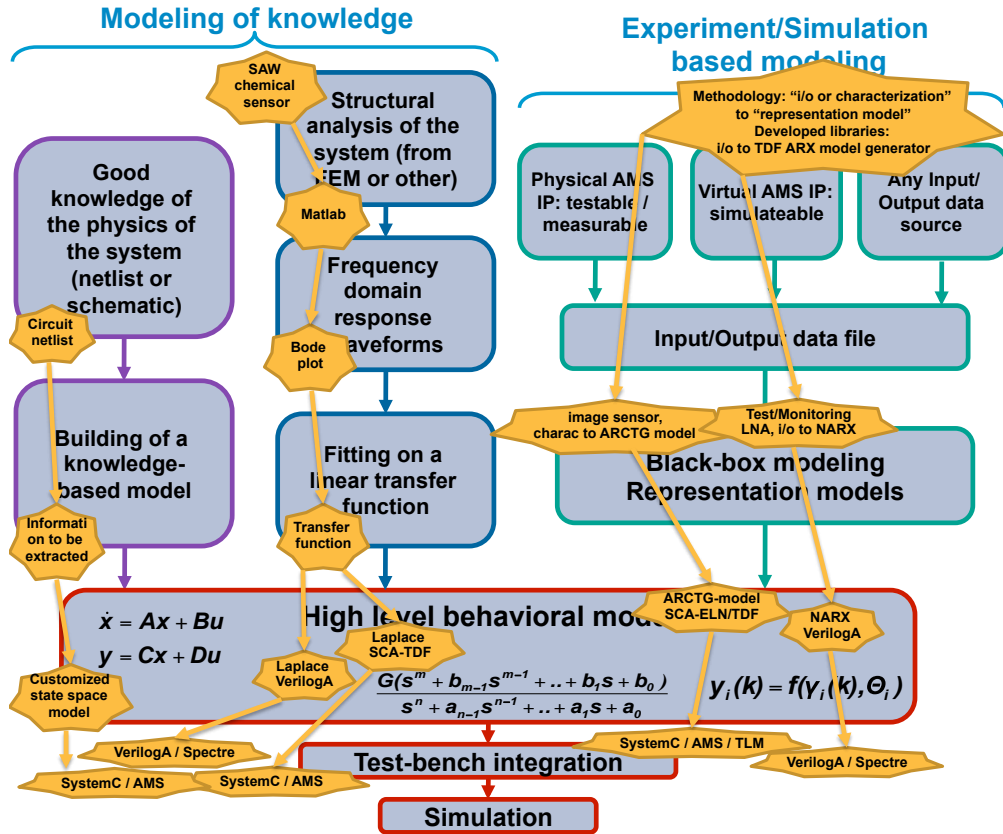


Figure 7.: Méthodes de modélisation comportementales et cas d'études.

La Figure 7 montre les cas d'étude abordés à l'intérieur du synoptique qui décrit les flots de modélisation. Dans le cadre de la *modélisation des connaissances* la méthodologie pour la *construction d'un macro-modèle à partir des réponses en fréquence* (branche bleue) est appliquée à un capteur chimique basé sur des ondes acoustiques de surface (SAW). Le capteur a été modélisé, simulé avec son interface microélectronique de frontend et fabriqué. Dans le cadre de la modélisation des connaissances à partir de la *netlist* d'un circuit, la procédure pour l'extraction d'une représentation d'état personnalisée a été étudiée et mise en œuvre pour l'environnement SystemC AMS, cela permet de générer un signal contenant la puissance instantanée consommée à chaque instant de la simulation.

Dans le scénario de la *modélisation basée sur des simulations*, une bibliothèque d'extension a été développée pour la construction de modèles AutoRégressifs à variable eXogène (ARX). La méthodologie de modélisation à boîte noire est appliquée à l'étude de cas d'un amplificateur

faible bruit (LNA). Sur l'extrême droite de la Figure 7 un modèle comportemental d'un LNA est obtenue par une identification du système par des *données de simulation* non seulement à fin de simuler le LNA dans un émetteur-récepteur RF complet, mais aussi pour surveiller et contrôler de manière adaptative les performances du LNA. Les concepts des statistiques sont ici exploités à des fins de surveillance et contrôle de performances et pour des buts de test du dispositif.

Dans le scénario de la *modélisation basée sur des expériences*, un capteur d'image CMOS industriel a été modélisé. Des modèles comportementaux de ce dernier ont été mis au point par des *données expérimentales* obtenues à travers une caractérisation opto-électrique de la réponse à la lumière du capteur d'image.

Cas d'étude industriel : capteur d'images CMOS

12. Motivation

Dans ce chapitre, un capteur d'image CMOS (CIS) sur la base la plateforme d'acquisition vidéo est modélisé en utilisant l'environnement SystemC. Un système d'acquisition d'image est composé de trois blocs principaux : un capteur d'image suivi par un processeur de d'images (ISP) et une unité centrale de traitement (CPU).

Ce travail se concentre sur le développement d'un modèle fiable et précis du comportement d'un CIS de STMicroelectronics dans le but de simuler son fonctionnement au sein de son environnement. Aujourd'hui, le développement de logiciels embarqués ne démarre que lorsque la CIS est physiquement disponible. Le logiciel est débogué et validé au moyen d'ensembles d'images émises par un capteur réel. Si des retards sont accumulés au cours du processus de conception il n'y aura pas beaucoup de temps pour le développement de logiciels, ceci débouchera sur un logiciel non-fiable ou dans un délai de mise sur le marché.

Puisque l'ISP doit également être en mesure de contrôler les paramètres du capteur, l'interopérabilité entre le capteur et l'ISP est normalement contrôlée en connectant une carte de test contenant un circuit logique programmable (FPGA) et une vraie puce de test du capteur avec un PC. Sur ce dernier tourne un modèle du matériel de la plateforme numérique avec le processeur cible (Instruction Set Simulator : ISS). Un prototype virtuel de l'ensemble du système permettrait d'éviter, à des stades préliminaires de conception, l'utilisation de cartes pour une première validation du système. Cela permettrait également de valider les algorithmes de l'ISP à travers des simulations et de les régler pour déterminer leurs conditions de travail limite. En ce qui concerne un CIS, des signaux électriques analogiques et numériques sont impliqués aussi bien que des quantités analogiques optiques.

13. Modélisation du CIS

Le modèle SystemC AMS est composée d'un cluster TDF, comme le montre la Figure 8. Trois pas de temps de TDF sont présents dans le modèle : le temps de trame (modules TDF exécutés à chaque temps de trame), le temps de pixel (modules TDF exécutés à chaque temps de pixel) et le temps de ligne pour l'exécution des convertisseurs analogique/numérique (ADC). Les trames du flux vidéo sont transmises d'un bloc à l'autre de la chaîne. Les images peuvent être générés de trois façons dans le modèle (voir Figure 8) :

- *IIB* : la premier émule la capture d'un objet en mouvement sur un fond fixe (appelée *constructeur de l'image d'entrée* (IIB). L'IIB est composé de deux sous-modules, le module

constructeur du fond (BGB) et le module *constructeur de l'objet* (OB). Le BGB construit le fond de l'image et l'OB dessine un objet sur le fond (une voiture par exemple). L'OB modélise également l'effet de la capture à volet roulant (Electronic Rolling Shutter : ERS) sur une voiture en marche.

- *DB* : la deuxième émule la situation d'un CIS inséré dans une boîte sombre avec des scénarios (charts) d'entrée et des conditions de lumière contrôlables, appelée *boîte noire* (DB). Le module *environnement* (ENV) fournit trois signaux de commande TDF au bloc *chargeur d'image* (IL). Les contrôles sont les suivants : quel chart doit être chargé entre les charts standardisés prédéfinies (scénarios), quelle est la couleur et l'intensité de la lumière qui illumine le chart. Le DB émule le fonctionnement d'une boîte noire contenant un ensemble de diodes électroluminescentes (LED) rouges, verts, bleus et blancs contrôlés par des signaux modulés à largeur d'impulsion (PWM).
- *Chargeur de fichier* : le troisième permet le chargement direct de fichiers d'image de référence.

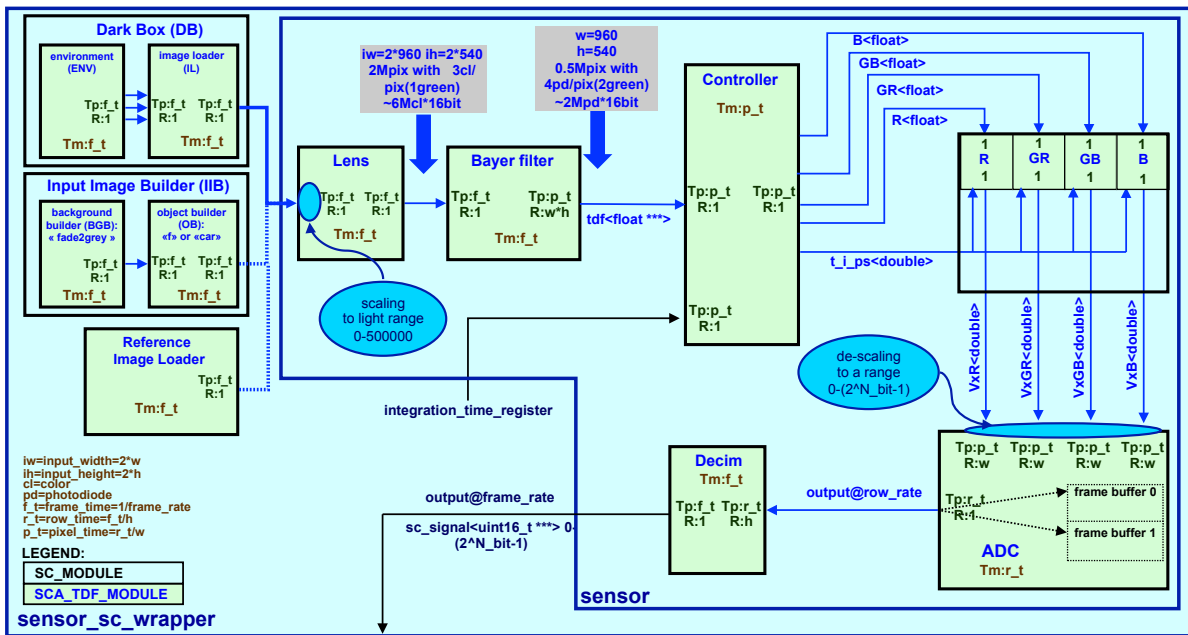


Figure 8.: Synoptique du modèle en SystemC AMS TDF du capteur d'images.

L'image délivrée par le DB ou IIB ou le chargeur de fichiers (Figure 8) est codée sur un nombre de bits paramétrique et envoyée au module émulant la *lentille*. L'image est envoyée au module émulant le *filtre Bayer* (BF) qui, à son tour, envoie les données à la partie intérieure du modèle, où le pas de temps TDF est réduit et l'ensemble de la matrice de pixels est balayée par une instance du module *pixel*. Le *contrôleur* pilote le *pixel* avec les signaux lumineux en provenance du *filtre Bayer*. Le *pixel* contient quatre *photodiodes* pilotées par les quatre signaux lumineux et par la valeur du temps d'intégration. Les signaux lumineux sont mis à jour, au début de chaque décharge. La valeur de tension à la fin du processus de décharge V_x est donnée en sortie par chaque *photodiode* suivant l'équation modélisant la décharge. La tension de colonne V_x est ensuite envoyée au banc de CAN (module ADC) et échantillonnées.

14. Comparaison de performances entre les modèles du CIS

Le Tableau 1 évalue les performances des différents modèles de CIS développés. Puisque les tailles des trames simulées sont différentes, les temps de simulation pour une trame ont été relativisés au pixel afin d'effectuer une comparaison. Le temps de simulation pour un pixel est calculé en divisant le temps de simulation pour une trame par le nombre de pixels de la trame. Le modèle VHDL-AMS est celui de référence pour calculer le gain en temps de simulation obtenu par les autres modèles.

Table 1.: Comparaison entre les vitesses de simulation des différents modèles.

Model	Size	Sim time for 1 frame	Time step	Sim Time for 1 pixel	Speed-up ratio	Speed-up ratio
VHDL-AMS reference	2x496	2h 30mn	Maximum time step = 5ns	Ref = 9s		1
ELN for photo diodes only+TDF	48x48	10mn	TDF time step =0.5 μ s	0.26s	1	x35
TDF array of pixels instantiated	48x48	2mn 40s	TDF time step =0.5 μ s	69ms	x3.7	x131
TDF one pixel sweeping the array	640x480	10s	50 TDF steps per pixel→ waveform	32.6 μ s	x7 949	x278 220
TDF one pixel sweeping the array	1920x1080	7s	1 TDF step per pixel→ no waveforms	3.4 μ s	x77 142	x2.7e6
↓ Enhancement to one "controller"	1920x1080	2s	1 TDF step per pixel→ no waveforms	0.96 μ s	x270 000	x9.2e6

Le modèle plus rapide est développé en SystemC AMS, ce modèle conduit à 2 secondes de temps de simulation pour l'acquisition d'une trame de 2 méga pixels (dernière ligne du Tableau 1). Ce gain est équivalent à un facteur d'accélération d'environ 5 ordres de grandeur par rapport au modèle écrit en SystemC AMS ELN et TDF, et de 7 ordres de grandeur par rapport au modèle VHDL-AMS, toutefois le niveau de précision est nécessairement réduite. Un temps de simulation rapide est crucial si le modèle du CIS est destiné à fonctionner dans la plate-forme globale SystemC TLM. Nous pouvons observer que le modèle ELN-TDF est équivalent au modèle VHDL-AMS en termes de niveau d'abstraction, toutefois celui-ci présente un gain d'environ 35 fois. Cela démontre la pertinence de SystemC AMS comme aide à la conception au niveau RTL. Le modèle SystemC AMS TDF plus rapide atteint un formidable facteur de speed-up, mais le niveau de précision est réduite dans le sens que sa simulation ne permet pas de tracer des formes d'onde des signaux qui sont, par contre, nécessaires pour le concepteur matériel/RTL. Cependant, cette modélisation de haut niveau peut permettre de prendre en

compte de nombreux aspects qui ne seraient pas possible de modéliser à l'aide VHDL-AMS en raison de leur poids de calcul. Notamment ces aspects sont : l'adsorption du filtre Bayer, la modélisation de la lentille et, évidemment, l'interaction avec la plate-forme SystemC TLM qui décrit le matériel numérique entourant le CIS. Ces raisons rendent le modèle approprié pour le développement/débogage du logiciel embarqué et pour la validation des algorithmes de l'ISP.

Conclusions

Conclusions

Dans cette thèse, j'ai travaillé sur la modélisation à un niveau comportemental de systèmes multi physiques, hétérogènes et à signaux mixtes.

Cette modélisation comportementale est destinée à fournir de nombreux bénéfices au flot de conception d'un système sur puce complexe contenant une partie analogique importante, principalement des capteurs et des actionneurs.

Pour effectuer une vérification complète du système il est proposé de composer les modèles comportementaux des IPs dans une plateforme virtuelle du système complet en utilisant un environnement unifié basé sur le langage C++. Plus précisément un grand nombre d'avantages peuvent être fournis par cette méthodologie, tels que l'exploration d'architecture, l'estimation des performances, validation de la réutilisation des IPs (IP reuse), la vérification du couplage entre les domaines du RF, analogique et numérique, la vérification précoce du développement de logiciels embarqués avec le débogage éventuel, la vérification de l'interopérabilité avec d'autres systèmes, et l'évaluation de l'impact d'autres bancs de test sur le composant cible.

En outre, la validation par le biais de prototypes virtuels permettra de tester la pertinence/conformité d'IPs pas encore disponibles à l'intérieur du système cible. Ceci permettrait de réduire la dépendance de la date de démarrage de la conception par rapport à la disponibilité d'un prototype matériel de l'IP afin d'anticiper le processus de conception de futurs produits et par conséquent, pourquoi pas, d'anticiper les avancées technologiques.

En ce qui concerne les techniques de modélisation, dans cette thèse trois flots sont présentés pour la construction de modèles analytiques et pour la réduction du degré de complexité de ces modèles.

Tout d'abord, une technique de modélisation comportementale en partant de la description sous forme de netlist, a été décrite et automatisée. Le but étant d'extraire des informations considérées d'importance par le concepteur sous la forme de représentation d'état.

Deuxièmement, les techniques basées sur l'ajustement analytique de réponse en fréquence sont explorées avec le but de réduire l'ordre du modèle ou bien pour identifier des modèles analytiques et donc simulables à partir de l'analyse effectuée avec d'autres outils de CAO spécifiques à l'application.

Enfin, des techniques d'identification de systèmes sont étudiées pour l'extraction de modèles boîte noire à partir de données obtenues de manière empirique, soit à partir de simulations de modèles précis ou à partir de données de mesure. Une bibliothèque preuve-du-concept a été mise en œuvre en utilisant SystemC AMS, celle-ci démontre l'applicabilité de la méthodologie.

Les techniques de modélisation comportementales ont été exploitées pour la conception et pour des buts de test et contrôle dans deux études de cas majeurs. D'un côté, la conception de

l'interface microélectronique d'un capteur chimique de vapeurs de mercure basé sur des ondes acoustiques de surface (SAW) a été faite. Cette méthode de conception exploite les techniques de modélisation à base d'approximations pour exécuter des simulations de bas niveau jusqu'au niveau layout. De l'autre côté, les techniques d'identification de systèmes sont appliquées à la modélisation d'un amplificateur faible bruit (LNA). Ce cas d'étude vise à une commande automatique du LNA par une estimation de ses performances à partir des paramètres du modèle. Cette boucle de rétroaction est destinée à la minimisation de la consommation d'énergie d'un émetteur-récepteur RF.

Bien que les outils développés pour l'extraction de modèle comportemental soient principalement orientés sur l'extension AMS du noyau de SystemC, la méthodologie peut être appliquée à d'autres langages de description matérielle analogiques (AHDL) tels que VHDL-AMS et Verilog-AMS.

Par la suite, le caractère industriel de la thèse a conduit à concentrer les efforts de modélisation sur un capteur d'image CMOS (CIS). Les modèles du capteur d'image sont extraits à différents niveaux en utilisant différents modèles de calcul de SystemC AMS. Cette modélisation à différents niveaux montre des gains en temps de simulation impressionnants.

L'intégration des modèles AMS du CIS dans différentes plates-formes virtuelles d'acquisition d'image en SystemC a été faite. Tout d'abord, l'intégration dans une plateforme SystemC TLM 2.0 preuve-du-concept a été achevée. Deuxièmement, le modèle SystemC AMS du CIS a été simulé dans une plateforme SystemC précise au bit. Cette plateforme est composée des nœuds d'un réseau de capteurs sans fils (WSN) pour une application automobile, cette application vise l'assistance au système de freinage des voitures afin d'éviter la collision. Par la suite, le modèle a été intégré dans deux différentes applications industrielles. Dans les deux cas l'interfaçage analogique/numérique entre les MoCs de SystemC AMS et les protocoles SystemC-TLM de STMicroelectronics a été validé.

Le client industriel ST-Ericsson est actuellement à la validation de la plateforme virtuelle TLM qui contient le modèle du CIS. Tant la partie analogique que la partie numérique sont ainsi décrites en utilisant l'environnement SystemC, respectivement AMS et TLM. La méthodologie de conception/vérification basée sur SystemC AMS est en cours d'adoption par ST-Ericsson dans le cas d'un SoC AMS pour les applications mobiles. L'applicabilité de la méthodologie pour le flot de conception industriel est prouvée et permettra un développement/débogage précoce du logiciel embarqué pour des SoCs AMS.

Considérations et perspectives

La naissance des langages de description matérielle analogiques et mixtes (VHDL-AMS et Verilog-AMS) a permis la validation de systèmes AMS et numériques avec de petites quantités de logiciel embarqué. Poussé par la complexité sans cesse croissante des systèmes sur puce, les outils/langages de modélisation exigent de faire abstraction à une vue de plus haut niveau qu'on peut définir comme "niveau système".

Du côté du numérique l'environnement SystemC offre une couverture de ce besoin au niveau système grâce à SystemC-TLM. Les extensions AMS de SystemC offrent maintenant les capacités de modélisation génériques de haut niveau même pour les parties AMS. D'autres modèles de calcul peuvent toujours être branchés sur l'architecture de SystemC, mais une définition de l'interface normalisée pour le branchement doit être fournie.

Le chef d'équipe en charge du prototypage virtuel en SystemC-TLM de ST-Ericsson a déjà déclaré ce qui suit : “Le développement/débogue du logiciel commence normalement une fois que les prototypes matériels sont disponibles et juste deux mois sont alloués pour le débogage avant que le produit soit vendu. La méthodologie basée sur TLM met à disposition des prototypes virtuels de la plateforme environ 9 mois avant la disponibilité de prototypes matériels. Ces prototypes virtuels sont adaptés pour le développement/débogue du logiciel embarqué. Un tel développement/débogue peut donc commencer avant la disponibilité du prototype matériel. Cela assure une plus grande fiabilité du logiciel, d'où une faible probabilité de bouges, et jusqu'à deux mois de gain du temps de mise sur le marché dans le cas idéal optimiste” (voir Figure 9).

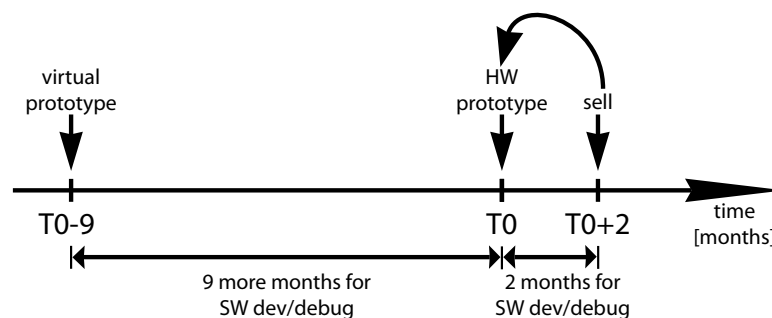


Figure 9.: Gain de temps et de fiabilité de SoCs AMS.

Un flot de conception entièrement automatisé de systèmes AMS à partir des spécifications à l'exploration d'architecture au moyen de simulations au niveau du système, jusqu'au layout (en passant par la synthèse de l'RTL) est encore assez loin d'être mature. Toutefois une validation du système grâce à une simulation de l'AMS/RF/numérique HW et SW dans un environnement C++ unifié est désormais une réalité.

Afin d'améliorer la réutilisation des IPs pour accélérer la phase de conception, la réutilisation des modèles n'est pas suffisante. En ce qui concerne le numériques, l'aspect de l'emballage (packaging) des IPs est couvert par le standard IEEE 1685-2009 IP-XACT (développé par le consortium de *Accellera Systems Initiative*). En ce qui concerne le côté AMS, le consortium du projet européen CATRENE MEDEA+ appelé *Beyond-Design Refinement of Embedded Analog and Mixed-Signal Systems* (Beyond-DREAMS) [Beyond-DREAMS 11] a soumis une proposition pour les extensions AMS de IP-XACT afin de gérer les spécifications pour les ports analogiques quand un IP AMS doit être emballé.

Dans la continuation du projet européen Beyond-DREAMS un autre projet européen appelé Heterogeneous-INCEPTION est en cour de démarrage. Ce projet visera à renforcer les capacités des extensions de SystemC AMS pour la modélisation de plates-formes multi capteurs/actionneurs pour l'optimisation de la consommation d'énergie. Du point de vue technique, il est prévu par la communauté SystemC AMS d'ajouter le support pour l'analyse aux éléments finis, des extensions pour les comportements non-linéaires, un pas de simulation TDF dynamique et la possibilité de simuler une description type “graphe de liaisons” (bond-graph). La polyvalence de l'environnement SystemC ouvre la voie à de futures applications et champs de recherche sur l'hétérogénéité.

Fabio Cenni

Date of birth : 09/08/1983

Place of birth : Faenza

Nationality : Italian

Email : fabio.cenni2@gmail.com



Education Path and Job Experiences

Mar'12-present Electronic CAD Engineer for Analog and Mixed-Signal systems verification methodology at STMicroelectronics, Crolles, France.

Mar'09-Feb'12 Industrial Ph.D. on Nano-Electronics and Nano-Technologies at TIMA Laboratory (University of Grenoble) and STMicroelectronics, Grenoble, France. Entitled : "High-Level Modeling of Heterogeneous Systems, Analog/Digital Interfacing". Modeling and simulation languages for Analogue and Mixed-Signal (AMS) systems. Focus on : SystemC AMS (also VHDL-AMS, Verilog-AMS).

Sep'08-Feb'09 Research fellowship at TIMA Laboratory, Grenoble, France.

May'08-Aug'08 Developer of C firmware for control/monitoring microcontroller-based printed circuit boards of Uninterruptible Power Sources (UPSs) and Static Transfer Switches (STSs) at Linari Enzo s.r.l, Forlì, Italy.

Sep'06-Mar'08 M.Sc. Electronics Engineering specialization on "Elaboration of the information", University of Bologna, Italy. 110/110 *cum laude*. Master thesis at TIMA Laboratory with Micrel Lab. (DEIS, Bologna) entitled "Microelectronics frontend interface architecture for a Surface Acoustic Wave based chemical sensor".

Sep'03-Aug'06 B.Sc. Electronics Engineering, University of Bologna, Italy.

Publications

Book Chapters

- 1 R. Khereddine, L. Abdallah, E. Simeu, S. Mir, F. Cenni, "Adaptive Logical Control of RF LNA performances for efficient energy consumption." book chapter of *VLSI-SoC 2010 Springer book : Forward-Looking Trends in IC and System Design*, pp. 43-68, Feb. 2012.

Peer-reviewed International Journals

- 2 F. Cenni, S. Scotti, E. Simeu, "A SystemC AMS / TLM platform for CMOS video sensors," *International Journal of Real-Time Image Processing (IJRTIP)*, Springer Ed., Special Issue on Design and Architectures for Real-Time Image Processing in Embedded Systems, (**submitted**, selected from DASIP 2011).
- 3 F. Cenni, S. Scotti, E. Simeu, "Behavioral modeling of a CMOS video sensor platform using SystemC AMS / TLM," *Design Automation for Embedded Systems Journal (DAES)*, Springer Ed., Special Issue on Languages, Models and Model Based Design for Embedded Systems, (**submitted**, selected from FDL 2011).

- 4 L. Bousquet, F. Cenni, E. Simeu, "Inclusion of Power Consumption Information in High-Level Modeling of Linear Analog Blocks," *International Journal Of Low Power Electronics (JOLPE)*, Publisher : American Scientific Publishers, vol.7, no.4, pp. 541-551(11), Dec. 2011.
- 5 F. Cenni, J. Cazalbou, S. Mir, L. Rufer, "Design of a SAW-based chemical sensor with its microelectronics front-end interface." *Microelectronics Journal*, Ed. Elsevier, vol.41, no.11, pp.723-732, Nov. 2010.

Peer-reviewed Conference Papers

- 6 A. L  v  que, F. P  cheux, M-M. Lou  rat, A. Aboushady, F. Cenni, S. Scotti, A. Massouri, and L. Clavier "Holistic Modelling of Heterogeneous Embedded Systems with High Multi-Discipline Feedback : Application to a Precollision Mitigation Braking System," *Design, Automation and Test conference in Europe (DATE 2012)*, Dresden, Germany, March 2012, pp. 739-744.
- 7 F. Cenni, S. Scotti, E. Simeu, "A SystemC AMS / TLM platform for CMOS video sensors," *IEEE Conference on Design and Architectures for Signal and Image Processing (DASIP 2011)*, Tampere, Finland, November 2011, pp. 164-169.
- 8 F. Cenni, S. Scotti, E. Simeu, "SystemC-AMS behavioral modeling of a CMOS video sensor," *19th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC 2011)*, Hong Kong, China, October 2011, pp. 380-385.
- 9 F. Cenni, S. Scotti, E. Simeu, "Behavioral modeling of a CMOS video sensor platform using SystemC AMS / TLM," *IEEE Forum for Design Languages (FDL 2011)*, Oldenburg, Germany, September 2011, pp. 1-6.
- 10 L. Bousquet, F. Cenni, E. Simeu, "SystemC-AMS High-level Modeling of Linear Analog Blocks with Power Consumption Information," *12th IEEE Latin-American Test Workshop (LATW 2011)*, Porto de Galinhas, Brazil, March 2011, pp. 1-6.
- 11 R. Khereddine, L. Abdallah, E. Simeu, S. Mir, F. Cenni, "Adaptive Logical Control of RF LNA performances for efficient energy consumption," Best paper award in *18th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC 2010)*, Madrid, Spain, September 2010, pp. 161-166.
- 12 F. Cenni, E. Simeu, S. Mir, "Macro-modeling of analog blocks for SystemC-AMS simulation : A chemical sensor case-study," *17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC 2009)*, Florian  polis, Brazil, October 2009, pp. 211-214.
- 13 F. Cenni, S. Mir, L. Rufer, "Behavioral modeling and simulation of a chemical sensor with its microelectronics front-end interface," *3rd IEEE International Workshop on Advances in Sensors and Interfaces (IWASI 2009)*, Trani, Italy, June 2009, pp. 92-97.

Other Publications

- 14 F. Cenni, "SystemC-AMS model of a CMOS video sensor," *OSCI SystemC AMS Day 2011 : Industry Adoption of the SystemC AMS Standard*, Dresden, Germany, May 2011, pp. 42-49.