Unified Multi-Level Design Environment for Mixed Signal Systems

Michel Vasilevski

UPMC
SORBONNE UNIVERSITÉS

Thèse de Doctorat de

l'Université Pierre et Marie Curie

École Doctorale Informatique, Télécommunications et Électronique (EDITE)

Présentée par :

Michel Vasilevski

Pour obtenir le grade de :

Docteur de l'Université Pierre et Marie Curie

Sujet de la thèse :

Environnement de Conception Multi-Niveaux Unifiée Appliqué aux Systèmes Mixtes

Date de soutenance : **Octobre 2012**

Le jury est composé de :

M. Ian O'Connor	Président du Jury
M. Alain Vachoux	Rapporteur
M. Gilles Jacquemod	Rapporteur
M. Serge Scotti	Examinateur
Mme. Marie-Minerve Louërat	Examinatrice
M. François Pêcheux	Examinateur
M. Alain Greiner	Co-directeur de Thèse
M. Hassan Aboushady	Directeur de Thèse

Unified Multi-Level Design Environment for Mixed Signal Systems.

A dissertation by:

Michel Vasilevski

Submitted to obtain the PhD degree from:

University of Pierre & Marie Curie (UPMC)

Defense date: **October 2012**

Committee in charge:

Mr. Ian O'Connor	Polytech'Nice-Sophia, France
Mr. Alain Vachoux	EPFL, Switzerland
Mr. Gilles Jacquemod	Lyon Institute of Nanotechnology, France
Mr. Serge Scotti	ST Microelectronics, France
Mrs. Marie-Minerve Louërat	UPMC, France
Mr. François Pêcheux	UPMC, France
Mr. Alain Greiner	UPMC, France
Mr. Hassan Aboushady	UPMC, France

Acknowledgements

I would like to express my gratitude to my adviser, Hassan Aboushady, for the guidance and support, to have put his trust in my work, and to have successfully supervised my work and the manuscript corrections. I thank the professors and Ph.D. students of the SoC department of the LIP6 laboratory in the University Paris 6, Pierre et Marie Curie, for their help at work. Especially, I thank Marie-Minerve Louërat, François Pêcheux and Hassan Aboushady for our collaboration in projects WASABI and BDREAMS. Especially, I thank Delaram Haghighitalab, Ramy Iskander and Farakh Javid for an important contribution to my work. Diomadson Rodrigues Belfort, Eldar Zianbetov, Andrii Dudka, Mootaz Allam, Raouf Khalil Ayad, Hussein Adel, Ahmed Ashry who helped in many ways in the course of my work. I would like to thank Diomadson Rodrigues Belfort, Isaac Maia Pessoa and Marcos Didonet del Fabro close friends, whom I could always count on. Finally, a special thank to Nicolas Beilleau for his grammar check and to Diomadson Rodrigues Belfort without whom the PDF version of the manuscript would not be in a standard PDF/A format.

Contents

List of Figures

List of Tables

Glossary

C_m	transcapacitance of a MOS transistor, $C_m = C_{dg} - C_{gd}$.
C_{bdmet}	RF model additional bulk-drain capacitance of a MOS transistor.
C_{bd}	bulk-drain capacitance of a MOS transistor.
C_{bg}	bulk-gate capacitance of a MOS transistor.
C_{bsmet}	RF model additional bulk-source capacitance of a MOS transistor.
C_{bs}	bulk-source capacitance of a MOS transistor.
C_{db}	drain-bulk capacitance of a MOS transistor.
C_{dg}	drain-gate capacitance of a MOS transistor.
C_{dsmet}	RF model additional drain-source capacitance of a MOS transistor.
C_{gb}	gate-bulk capacitance of a MOS transistor.
C_{gd}	gate-drain capacitance of a MOS transistor.
C_{gs}	gate-source capacitance of a MOS transistor.
C_{mb}	transcapacitance of a MOS transistor, $C_{mb} = C_{db} - C_{bd}$.
C_{mx}	transcapacitance of a MOS transistor, $C_{mx} = C_{bg} - C_{gb}$.
C_{sd}	source-drain capacitance of a MOS transistor.
I_{AB}	DC part of the current through a component.
I_{ab}	phasor of the current through a component.
$K_{2_{g_m g_{ds}}}$	2^{nd}-order nonlinearity coefficient in the two-dimensional power series expansion of the function describing the nonlinearity of the current controlled by v_{gs} and v_{ds}.

K_{2g_m}	2^{nd}-order nonlinearity coefficient in the power series expansion of the function describing the nonlinearity of the current controlled by v_{gs}.
$K_{2g_{ds}}$	2^{nd}-order nonlinearity coefficient in the power series expansion of the function describing the nonlinearity of the current controlled by v_{ds}.
$K_{3_{2g_m g_{ds}}}$	3^{nd}-order nonlinearity coefficient in the two-dimensional power series expansion of the function describing the nonlinearity of the current controlled by two orders of v_{gs} and one of v_{ds}.
$K_{3_{2g_{ds} g_m}}$	3^{nd}-order nonlinearity coefficient in the two-dimensional power series expansion of the function describing the nonlinearity of the current controlled by one order of v_{gs} and two of v_{ds}.
K_{3g_m}	3^{nd}-order nonlinearity coefficient in the power series expansion of the function describing the nonlinearity of the current controlled by v_{gs}.
$K_{3g_{ds}}$	3^{nd}-order nonlinearity coefficient in the power series expansion of the function describing the nonlinearity of the current controlled by v_{ds}.
V_{AB}	DC part of the difference between the voltage at node A and B.
V_{ab}	phasor of the difference between the voltage at node A and B.
g_m	transconductance of a MOS transistor.
g_{ds}	output conductance of a MOS transistor.
g_{mb}	bulk transconductance of a MOS transistor.
i_{AB}	total current through a component. Hence $i_{AB} = I_{AB} + i_{ab}$.
i_{ab}	AC part of the current through a component.
r_d	RF model drain resistance of a MOS transistor.
r_g	RF model gate resistance of a MOS transistor.
r_s	RF model source resistance of a MOS transistor.
r_{sti}	RF model bulk resistance of a MOS transistor.
v_{AB}	total of the difference between the voltage at node A and B. Hence $v_{AB} = V_{AB} + v_{AB}$.
v_{ab}	AC part of the difference between the voltage at node A and B.

1

Introduction

Nowadays, System-on-Chips containing digital, analog and RF blocks are very common and are present in almost all electronic devices. This fact brings the designers of analog and digital domains to work together to apply a complete analog/digital Mixed-Signal design.

A system-level model containing both the analog and digital parts is very important to determine the circuit specifications and to validate the desired system performance. Tools available to model such systems are currently lacking: Matlab, a widely used high-level simulator is not compatible with the integrated circuit design flow. VHDL-AMS, a language for analog mixed-signal circuit description is very time-consuming for large systems. Recently, an AMS extension of SystemC, called SystemC AMS, has filled this gap of Mixed Signal modeling in a system level.

At the circuit level, the conventional analog design methodologies have been mainly based on the analog designer approximated calculations and computer-aided simulations for tuning. The design time of analog and RF blocks is very dependent on the designer's experience. When the CMOS process or the specifications are changed a complete redesign is necessary.

Moreover, it is very difficult to optimize the overall design of a complex mixed-signal circuit because the design and simulation environments used for the digital blocks are very different from those used for the analog and RF blocks. The following points summarize the contributions realized in this work.

- The first implementation of a fairly complex Mixed Signal model in SystemC-AMS: a Wireless Sensor Network node.
- Refined models for a generic and configurable approach of system-level modeling.

- An accurate linear and nonlinear circuit performance evaluation tool for system-level refined models back-annotation and circuit-level optimized design.
- An optimized circuit design methodology based on an accurate sizing tool and the circuit performance evaluation tool.
- A unified Mixed Signal design environment with a very strong interaction between system-level simulation and optimized circuit-level design.

1.1 Outline

The thesis follows the outline illustrated in Fig. 1.1.

Chapter 2 discusses the state of the art in the context of the thesis.

Chapter 3 presents the SystemC AMS language applied for mixed signal high-level modeling.

Chapter 4 presents the system-level refined behavioral modeling of Mixed Signal systems.

Chapter 5 presents the performance evaluation procedure.

Chapter 6 presents the systematic circuit-level design and optimization. The proposed methodology is illustrated with two case studies: a GmC integrator design and a Low Noise Amplifier design.

Chapter 7 presents the unified multi-level design environment based on system-level model refinement and optimized circuit-level design. The environment is illustrated with two case studies: a GmC integrator design for a $\Sigma\Delta$ ADC and a Low Noise Amplifier design for a ZigBee RF transceiver. Both ADC and RF transceiver are crucial components of a Wireless Sensor Network node.

Chapter 8 concludes the thesis and presents the future work.

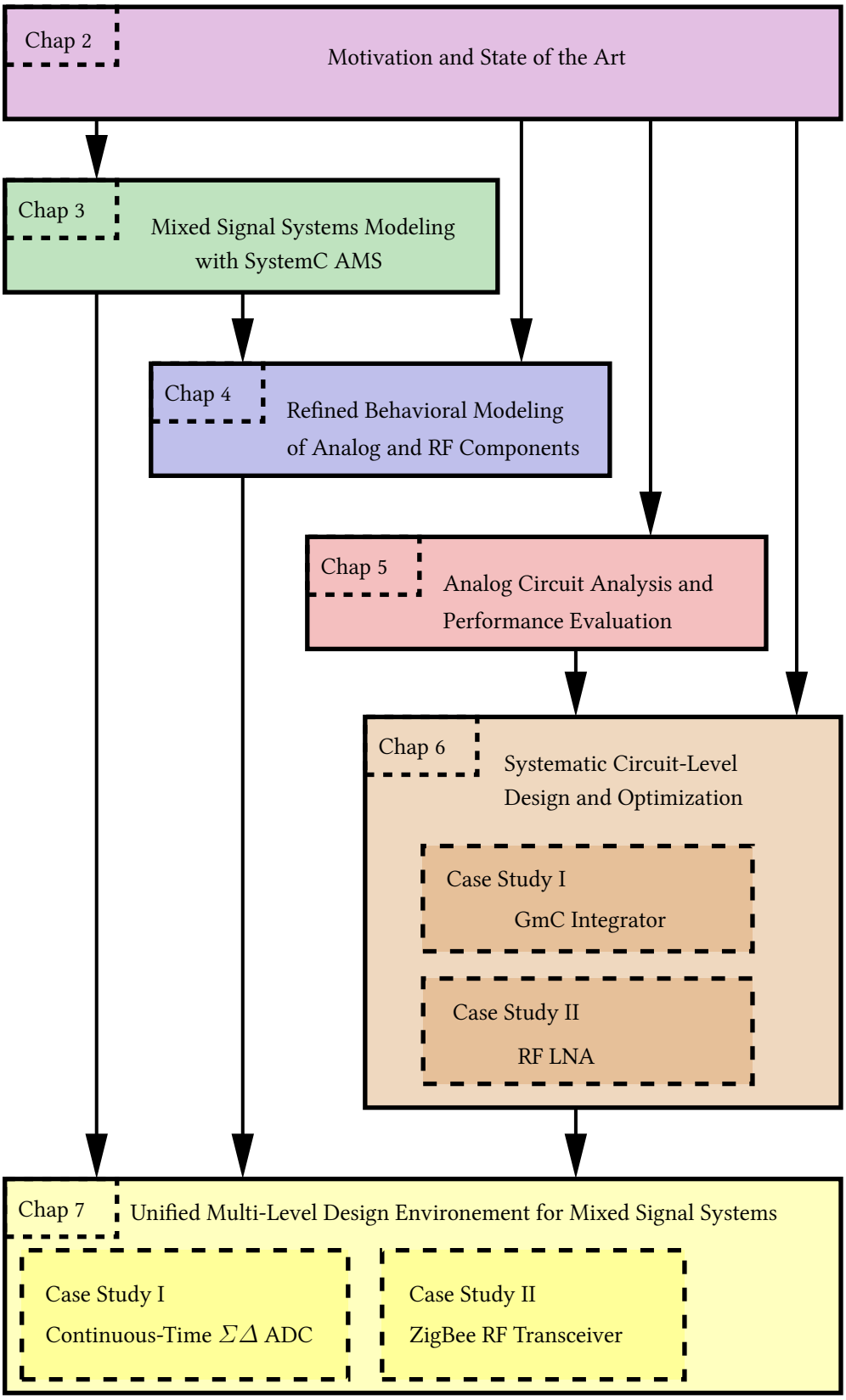

Fig. 1.1: Graphical outline of the thesis.

2

Motivation and State of the Art

2.1 Introduction

To illustrate the importance of this work, the following chapter reviews the state of the art in the field of CAD tools for Mixed Signal systems design in both system and circuit levels. In the first section, we present the context and a global motivation for having implemented a unified multi-level design environment for mixed-signal systems. In the second section, we present the state of the art in the fields of interest. In the third section, we present the major contributions brought by our work.

2.2 Unified Multi-Level Design Environment for Mixed Signal Systems

Mixed Signal systems design involves complex models, as it manages a several number of components. At this level of complexity, circuit-level simulations have to be used in moderation and global validations can only be done at the system level. With the increasing popularity of wireless communication systems, RF circuit design became very important. At those high frequencies and when using submicronic technologies, the design methods based on simplified models, cannot be used. Until recently, such complex systems were validated by hardware/FPGA [Pena07].

A conventional design flow operates a top-down design and a bottom-up verification [Gielen00]. When the performance of a level of abstraction does not match to the specifications, a redesign is operated at this level of abstraction, changing the topology or modifying the parameters involved in the degrees of freedom of the design.

A more innovative design flow, presented in Fig. 2.1, manages a performance/specification import/export between the levels of abstraction [Carloni02, Rabaey06]. It permits to benefit from the lower level informations, allowing precise and fast simulations. In this kind of flow, refining system-level models is important to be able to back-annotate the circuit-level performance [Rutenbar07].

Furthermore, a platform-based methodology has been presented in [Ferrari99, Sangiovanni-Vincentelli04], replacing the traditionally called top-down and bottom-up designs by a meet-in-the-middle flow. The illustrated design examples for a platform-based design always processed the performance evaluation by simulation [Carloni02, Nuzzo05]. The platform-based design is an interesting approach but rarely applied to a complex system as it supposes to have a collection of architectures or topologies for each component.

In our work, we followed the multi-level design flow, presented in Fig. 2.1, exchanging the results of system-level simulations: "the specifications", and the results of circuit-level performance evaluation: "the performance". Optimizing this flow implied making choices for each level of abstraction. These choices are discussed, in the state of the art of mixed-signal design and simulation tools in Section 2.3.

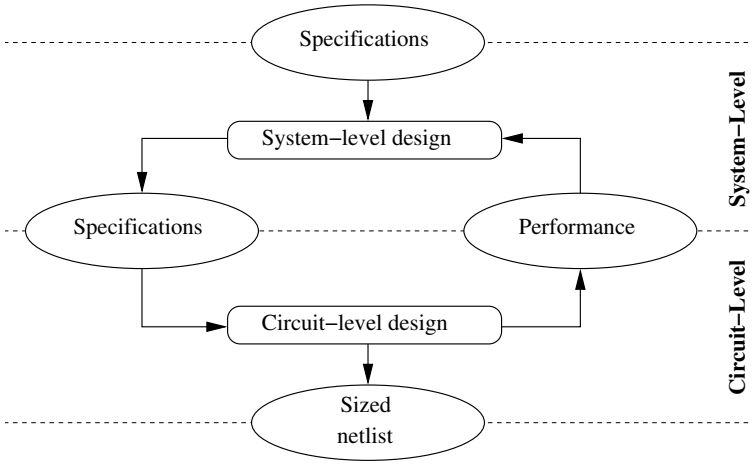

Fig. 2.1: The multi-level design flow with both top-down and bottom-up processes.

2.3 State of the art of Mixed-Signal Design and Simulation tools

2.3.1 Mixed Signal Systems Modeling and simulation

As the first stage of a complete multi-level design flow, the system-level modeling state of the art is presented in this section. For the past 20 years, hardware description languages have been widely used to model and simulate systems belonging to various engineering fields, from digital and analog electronics to mechanics, RF and even battery cell chemistry. The system-level description has been classified into levels of abstraction [Vachoux97]: functional level for signal flow diagrams described by mathematical equations and behavioral level for block diagrams described by DAE (Differential-Algebraic Equations) or s-domain transfer functions. Electronic Design Automation (EDA) industry recently proposed coherent modeling and simulation frameworks that allow the description of systems from different disciplines and for the description of interactions between these systems. These frameworks use VHDL-AMS [Christen99, Ashenden02, Normark04, Pecheux05] and Verilog-AMS [Frey00, Pecheux05] as effective backbones for modeling the behavioral level. However, when dealing with Mixed Signal systems, like a Wireless Sensor Networks (WSN) containing dozens of nodes, and with a carrier frequency of a few gigahertz, these frameworks rapidly show their limitations in terms of simulation speed. Up to now, the only way to validate the communication between WSN nodes was to run test benches with hardware component [Pena07].

Recently, SystemC AMS [sysb, Vachoux04], an AMS extension to the widely used SystemC, has been proposed for efficient high-level modeling and simulation of Mixed Signal systems. As illustrated in Fig. 2.2, the idea behind SystemC AMS is to fill a missing "architectural" level of abstraction in Mixed Signal systems. The language has the advantage of being a simple C++ library, and therefore to inherit the experience of 30 years of contribution and optimization. In Fig. 2.3, [Mahne11] presented the levels of abstraction covered by the mostly used languages in SoC design. The segment representing SystemC AMS language begins in a very high level of abstraction as it is based on C++, and covers additional levels as the language implements a complete library for system-level simulation.

When our work started, the state of the art around SystemC AMS was limited to some simple implementations [sysc]. Some applications were presented using SystemC AMS, for example, mod-

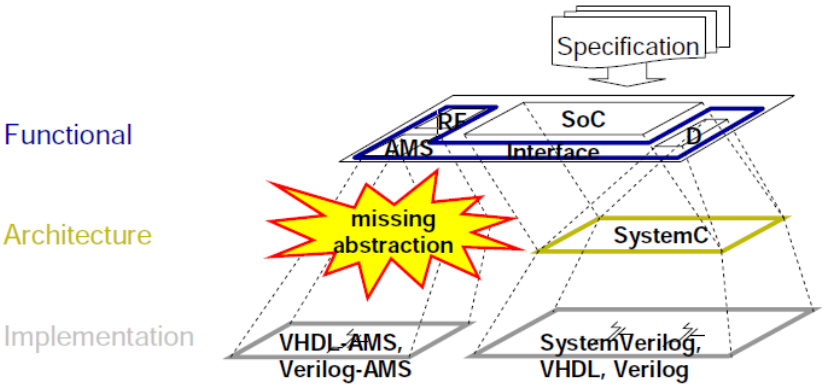

Fig. 2.2: Missing AMS modeling language at architecture level [sysb].

eling acceleration sensor arrays [Markert06] or for modeling a wired communication system [Einwich05]. Fairly complex systems were developed in these case studies but they lacked refined models for the analog blocks taking into account circuit non-idealities.

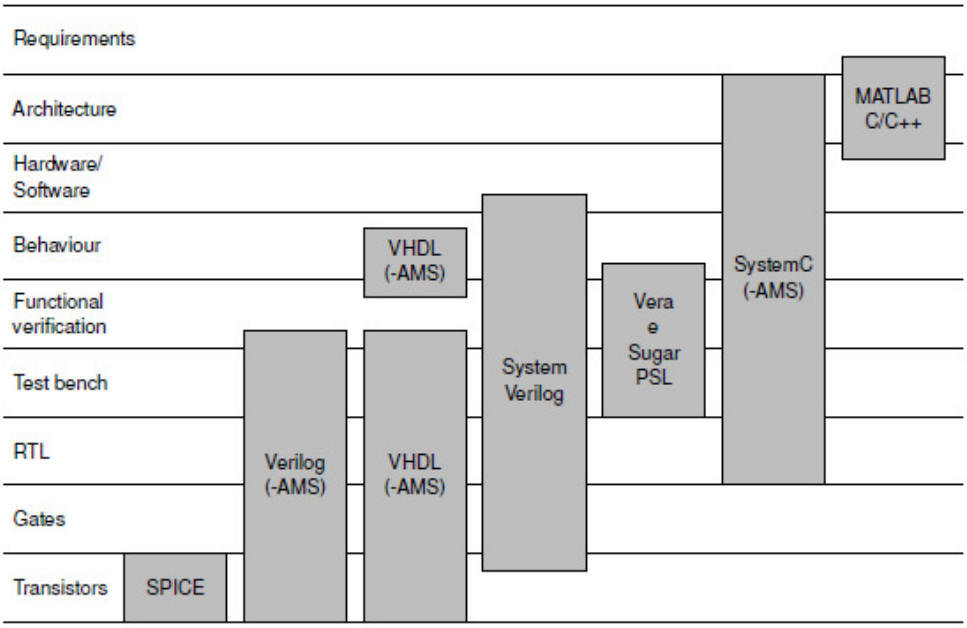

Fig. 2.3: Usage of modeling and verification languages in SoC design process [Mahne11].

2.3.2 Systematic Circuit Analysis and Design

As the second stage of a complete multi-level design flow, the circuit-level design state of the art is presented in this section. From the circuit designers point of view, the Electronics Design Automation EDA is becoming very important, as the analog circuit complexity is growing with the new technologies and high frequency applications. Two main categories subdivide the analog circuit design [Gielen00, Daems03]: knowledge and optimization-based.

- Knowledge-based:

  As shown in Fig. 2.4, the designer experience is captured as a design plan, the equations are simplified but the approximation is absolutely conscious and controlled. Many tools implemented a knowledge-based approach, IDAC [Degrauwe87] permitted a fast execution but lacked of flexibility as the set of topologies was fixed. BLADES [El-Turky89] introduced artificial intelligence. OASYS [Harjani89], PAD [Stefanovic05] introduced hierarchical design to manage more complex designs. In general this approach suffered from the large effort to introduce new topologies and from the precision of the equation implementing the knowledge.

  The characteristic of the knowledge-based approach is fast synthesis with low precision.

  To solve the problem of precision OCEANE [Iskander07] embedded the standard transistor models for knowledge-based design of operational amplifiers, and CAIRO+ [Iskander07] made it more flexible to be able to introduce new topologies implementing a C++ library for the designer. When a technology was based on a new model, OCEANE and CAIRO+ had to be upgraded with a complex implementation of the model inversion. To be absolutely technology independent, CHAMS [Javid09] implemented a C++ library for hierarchical synthesis with a simulation-based sizing instead of a model-based (equation-based) sizing implemented in OCEANE and CAIRO+. Although those sizing tools are based on equations or simulation for the sizing of each transistor, the entire circuit is mostly designed in a knowledge-based methodology.
- Optimization-based:

  As shown in Fig. 2.5, the methodology consists of optimization loops that converge on a design to achieve the constraints that describe the required performance. This approach is composed of two sub-categories: simulation and equation-based.

– Simulation-based:

A simulator is used to extract performance, a loop-based design algorithm is exploited to reach the specifications [Daems03, Phelps00].

From the time of the first implementations of an electrical simulator [Nagel71] [Nagel75], the simulation-based approach has been developed, as in AOP [Hachtel75], APLSTAP [Hachtel80], DELIGHT.SPICE [Nye88]. OAC [Onodera90] was dedicated to operational amplifiers. These first tools was not scaled to large circuits because of the time spent on simulation for each loop.

FRIDGE [Medeiro94] implemented annealing algorithm to make a fast convergence to the desired performance. And finally, MAELSTROM [Krasnicki99] and ANACONDA [Phelps00] introduced parallelization to speed up the simulation-based optimization.

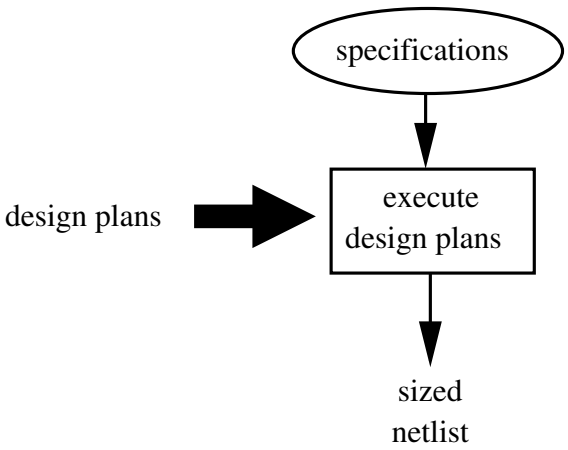

Fig. 2.4: The knowledge-based approach using procedural design plans [Gielen00].

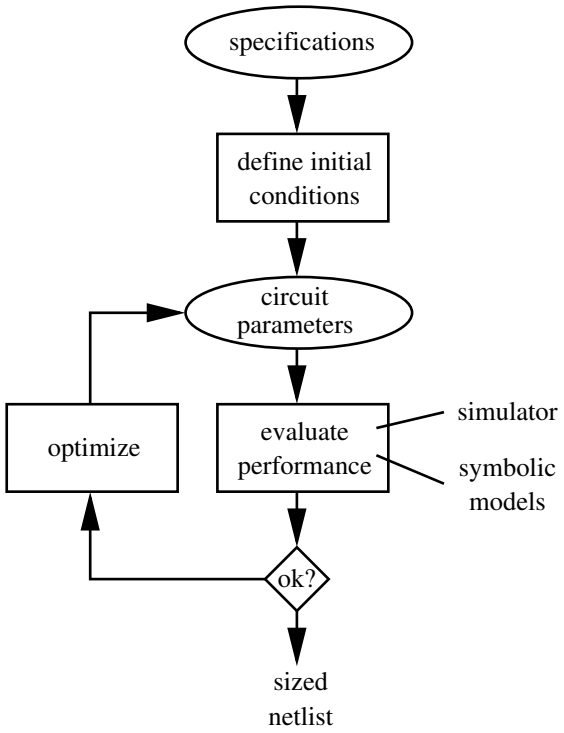

Fig. 2.5: The optimization-based approach [Gielen00].

However, the problem of this approach is the optimization speed, in another hand, it is the most independent with respect to the targeted technology and specifications. The characteristic of this approach is high precision with a long synthesis time.

– Equation-based:

The performance is described by equations, a loop-based design algorithm is exploited to reach the specifications. At first, the approach was using a performance description with equations derived by hand, as in OPASYN [Koh90] and STAIC [Harvey92].

DONALD [Swings91] introduced an automatically generated performance but with simplified equations.

As the performance can be generated with CAD tools, this allows flexibility and portability [Ochotta96, del Mar Hershenson98]. A lot of work proposed symbolic analysis [Gielen94, Aksin05] to generate accurately the performance equations. This method suffered from the growing complexity of circuits and needed to be combined with circuit reduction algorithms [Tlelo-Cuautle07, Gielen07]. At first, ISAAC [Gielen89] and ASAP [Fernandez91] was operating symbolic simplifications by extracting the dominant contributions with a user-defined tolerance. But, as symbolic generation is very complex, those tools were limited to a dozen of transistors. At second, the tools SYMBA [Wambacq95] and RAINIER [Yu96] implemented symbolic simplifications before generation (SBG) to enable a large scaled circuit symbolic analysis. Furthermore, equations developed for circuit analysis are usually based on the linear small-signal model with simplified models for non-linearity [Martens05].

[Tulunay08] presented a design flow with an equation-based optimization procedure for a LNA design. The evaluation of linear and nonlinear performance is done by generating symbolic equations. However, the approximations that have been done for linear and nonlinear performance have resulted in significant difference between estimated and simulated/measured results. These differences can give incorrect directions during the optimization process. The first reason of those differences is the estimation of small-signal parameters, as reported in [Tulunay08]. A second reason comes from the use of a simplified Volterra nonlinearity model that supposes the only contribution of v_{GS} to the nonlinearity of i_{DS} in a MOS transistor [Tulunay06].

The characteristics of the equation-based approach combine fast synthesis with a controlled precision.

Table 2.1 summarizes the advantages of each design automation method. As it has been mentioned, the knowledge-based technique is fast but inaccurate, the simulation-based optimization technique is slow but permits to obtain the best accuracy. Finally, the equation-based optimization can be seen as a trade-off, because the speed is better than simulation-based and, if the performance evaluation equations are precise, it can be as accurate as a simulation. We specified an additional line on Table 2.1 to compare the ability to implement each design technique into a unified process. This is important to prevent time consuming round trips, between different languages/softwares, for optimization procedure calls. The simulation-based solution is more difficult to implement into a unified environment as it would suppose to implement the environment with the simulator language. Whereas, the equation-based solution simply uses equations for the performance that can be implemented easily into each kind of design environment. Following the observations, the equation-based technique has been chosen for the implementation of the design flow in our work.

Table 2.1: Comparison between the different analog design automation techniques.

	Knowledge based	Optimization based	
		Simulation based	Equation based
Precision	X	✓✓	✓
Speed	✓✓	X	✓
Homogeneity	✓✓	X	✓✓

2.4 Major Contributions

2.4.1 Mixed Signal Systems Modeling with SystemC AMS

The observation of state of the art in the area of system-level modeling, reveals a lack of high-level modeling of complex mixed signal systems. In this work, we present in Chapter 3 a design platform for the physical layer of a WSN node, consisting of an ADC, an RF transceiver and a

digital microcontroller. For fast RF simulations, baseband-equivalent modeling has been introduced, with emphasis on the genericity of the implementation. This is the first implementation of such a sophisticated model as it encloses analog, digital and RF models.

When this work [Vasilevski07b, Vasilevski07a] was published, it presented the first complex SystemC AMS model of a WSN node. Since then, the community of SystemC AMS users has been growing rapidly as the list of publications [sysc] suggests. More than 40 publications have referenced our work that served as a base for more advanced models in different domains: seismic vibrations [Leveque10], biological analyzes [Habib10], RF fading channel [Massouri10], energy harvesting [Hormann11] [Caluwaerts08]. In terms of complexity of a whole system model, [Beserra11] presents an interesting and evolved WSN simulation platform by combining SystemC AMS with a SystemC Network Simulation Library (SCNSL) [Fummi08]. A massive multi-nodes network simulation is presented in [Zhou11]. In terms of speed, a ratio greater than 100 in simulation time from a VHDL-AMS model to a SystemC AMS model was revealed in [Cenni11b]. In terms of interoperability, the works of [Cenni11a] presents a co-simulation with SystemC TLM Model of Computation. Finally, in terms of flexibility of the language, [Maehne09] presented an extension to support dimensional analysis providing a verification of the specifications dimensional consistency.

2.4.2 Refined Behavioral Modeling of Analog and RF Components

In the state of the art section, we pointed a lack of complex systems model that implemented non-idealities. Refined models, taking into account circuit non-idealities of the analog and RF blocks are also presented in this work (Chapter 4), with a special care for genericity of models. Thus, the implemented SystemC AMS model refinement consists of three kind of block: the analog component, the RF component and the sine wave source component. An other paper [Vasilevski08b, Vasilevski08a] presented this extended work with model refinement and simulation improvement.

2.4.3 Analog and RF Circuit Analysis and Performance Evaluation

We implemented a performance evaluation procedure that is motivated by two main points illustrated in Fig. 2.6: back-annotation of system-level refined models and circuit-level optimized biasing and circuit sizing. Each analyzed topology is associated to a symbolical admittance ma-

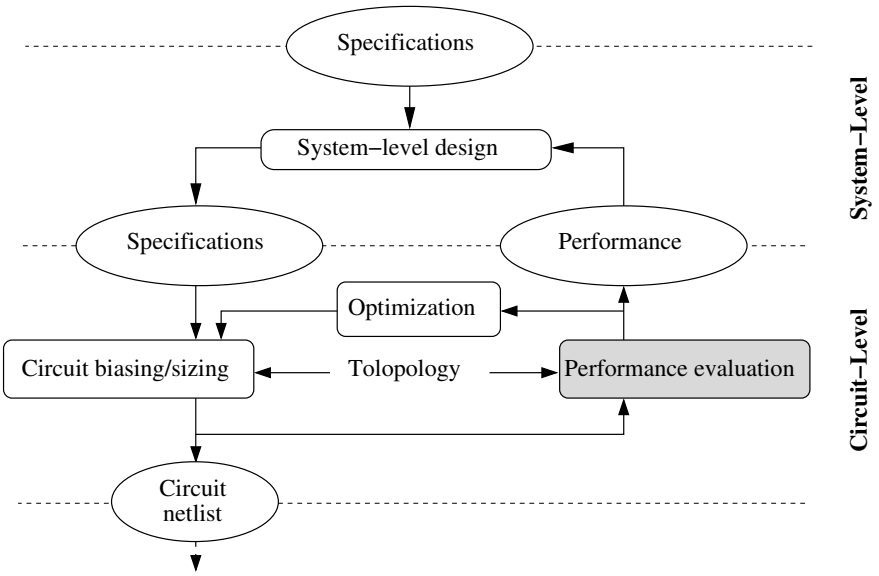

Fig. 2.6: The detailed multi-level design flow with both top-down and bottom-up processes.

trix built from the complete small-signal model of the circuit. This way the linear performance matches perfectly with a transistor-level simulation result. The performance evaluation procedure also supports nonlinearity thanks to the use of Volterra series and an accurate implementation is incorporated into the procedure. To maintain the homogeneity of the flow the matrices of performance are produced in a C++ format.

2.4.4 Systematic Circuit-Level Design and Optimization of Analog and RF Circuits

We have chosen an equation-based methodology following Fig. 2.7, but we considered that the control on the precision is too much dependent on the specifications and the technology. By experience, the approximation that is validated for one configuration could be invalid for another. That's why, the effort was done on describing precisely the performance to make it as generic as possible the synthesis procedure. Furthermore, comparing to a simulation-based methodology, the speed of our approach is better as we avoided heterogeneous round trips between different softwares, as the entire environment is C++ based, and since we avoided transient simulation for nonlinear performance evaluation.

As described in Fig. 2.7, additionally to the previously described performance evaluation procedure, transistor-level circuit biasing/sizing and optimization procedures have been implemented

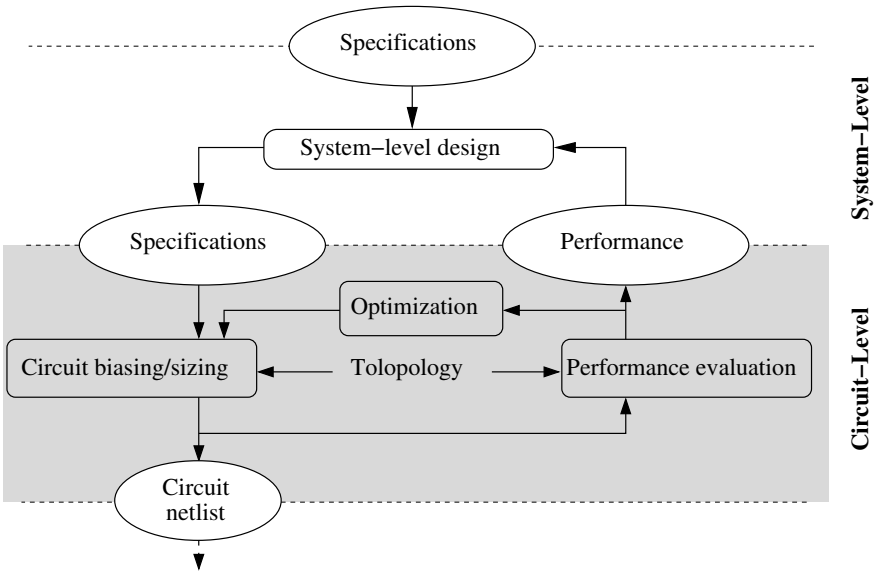

Fig. 2.7: Unified multi-level design environment for mixed signal systems.

to complete the equation-based circuit design flow. C++ based CAIRO+ and CHAMS have been used for precise sizing and small-signal parameters extraction. The performance is calculated numerically to allow a complex circuit to be analyzed very fast. Thus, the contribution in this field consisted of implementing a circuit-level equation-based methodology that provides a high speed of execution thanks to a unified C++ implementation.

The methodology has been applied to two different design examples: a GmC integrator for a $\Sigma\Delta$ ADC design and a RF Low Noise Amplifier for a ZigBee RF receiver design. The examples have been chosen as they are both mixed-signal components that constitute a Wireless Sensor Network Node.

2.4.5 Unified Multi-Level Design Environment for Mixed Signal Systems

Our approach is inspired by the idea of building a communication between the abstraction levels as described in Fig. 2.1. We propose a multi-level design flow and we apply this approach to a Wireless Sensor Network (WSN) node design. By incorporating the circuit-level evaluated performance into the system-level models, we are able to run fast but also accurate simulations. The systematic design flow is entirely implemented in a unified C++ environment thanks to the use of C++ libraries for each stage: SystemC AMS, CHAMS, GiNaC. By describing the entire flow in a C++ environment, the design procedures are packaged in a generic analog IP. To demonstrate the methodology,

two case studies are presented, the continuous-time $\Sigma\Delta$ ADC and the Zigbee RF transceiver they constitute the mixed-signal components of the Wireless Sensor Network node.

2.5 Conclusion

In this chapter, we have presented the systematic design flow implementing the most innovative methodology, namely a unified multi-level design flow. As a multi-level design flow manages a link between both system and circuit level abstractions, we have presented the state of the art in the field of each level. In the system level, we have identified the lack of an abstraction level for fast mixed-signal simulations. In the circuit level, we reviewed the advantages of each systematic analog circuit design technique. In the last section, we presented the major contributions of this work organized in the same order as the chapters of the thesis.

3

Mixed Signal Systems Modeling with SystemC AMS

3.1 Introduction

This chapter presents mixed signal systems modeling using SystemC AMS. In Section 3.2, we introduce the SystemC AMS Models of Computation (MoC) with emphasis on Timed Data Flow (TDF) MoC. In Section 3.3, a Wireless Sensor Network node model is presented as an example of a complex Mixed Signal system implemented with SystemC AMS. Finally, a technique is presented in Section 3.4 to accelerate RF simulation with SystemC AMS.

3.2 SystemC AMS

3.2.1 Models of Computation

This section summarizes the concepts behind AMS description and the details of each SystemC AMS Model of Computation (MoC) covered in the SystemC AMS User's Guide [sysb]. SystemC AMS 1.0 is the Open SystemC Initiative (OSCI) standard [sysa] for Analog/Mixed-Signal (AMS) modeling. It is an extension of the SystemC standard (IEEE std. 1666-2005) based on C++ language (ISO/CEI 14882:1998) that is widely employed [sysd] for complex digital systems description. Because of the possibility of describing the behavior with all the functionalities offered by C++, the level of abstraction can be very high. The designer controls the abstraction by managing the granularity and the complexity of the models, and by choosing the Model of Computation. A Model of Computation (MoC) is a set of rules that define the behavior of components and the interaction between them. The choice of a MoC is related to the model abstraction and the use cases. Fig. 3.1,

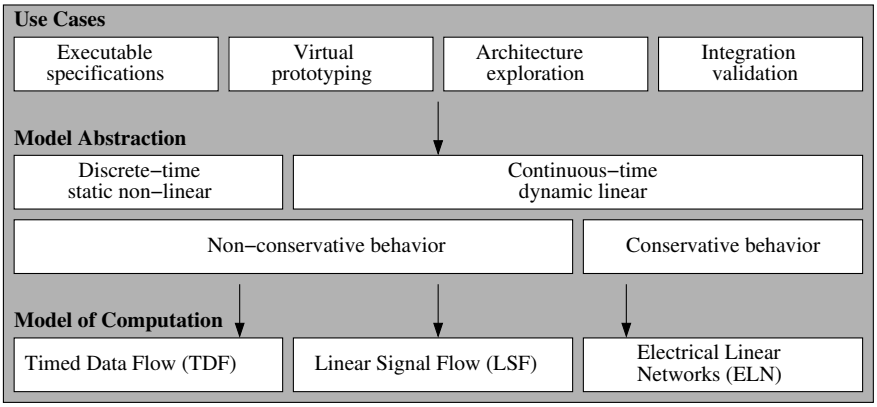

Fig. 3.1: SystemC AMS from Use Cases to Models of Computation [sysb].

extracted from the SystemC AMS User's Guide [sysb], depicts the supported SystemC AMS MoC related to their model abstraction and the use cases.

Fig. 3.2, extracted from the SystemC AMS User's Guide [sysb], shows the layered architecture of SystemC AMS on top of SystemC. This architecture permits to model different domains, their mapping to various Models of Computation (MoC) and their interaction. In this way, different parts of a complex heterogeneous system can be modeled and simulated using the optimal methodology and solving algorithm. For system engineering, this approach permits a higher modeling efficiency and an order of magnitude faster simulation for AMS modeling [Barnasconi10]. The available SystemC AMS prototype provides three dedicated Models of Computation.

The first MoC permits the description of conservative Electrical Linear Networks (ELN). It is implemented with a continuous-time solver applying a Modified Nodal Analysis (MNA) to determine the voltages and currents. With this MoC, the designer simply declares the netlist of interconnected linear elements like resistors, capacitors, inductors, current/voltage sources, in a SystemC module.

The second MoC is Linear Signal Flow (LSF). It is implemented with a non-conservative continuous-time solver. The main difference with respect to ELN is that LSF does not operates on flows and potentials but the signal is abstracted without guaranteeing energy conservation. It allows to interconnect some predefined primitives like adders, subtractors, differentiators, integrators.

The third MoC is multi-rate Timed Data Flow (TDF), which can be used to describe analog non-conservative behaviors. It is implemented with a discrete-time solver with a constant time step. In TDF, the analog behavior is described within module functions, which communicate directly via

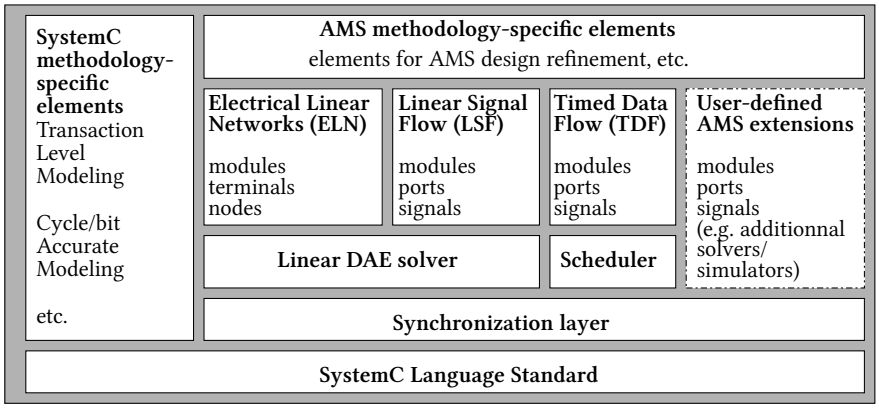

Fig. 3.2: SystemC AMS layered architecture [sysb].

input and output ports. TDF is especially well suited for high complexity systems as it operates at a very high level of abstraction. In this work, we have chosen to model the Wireless Sensor Network node with a TDF MoC because of its high level of abstraction and because of its ability to describe nonlinear behaviour in a discrete-time simulation.

3.2.2 Modeling using Timed Data Flow Model of Computation

In our application, the multi-rate TDF MoC is of particular interest, whereby continuous-time behavior of a subpart of an analog block is embedded into a data flow module processing method. As this MoC runs discrete-time non-conservative simulations, the behavior is expressed by the transient flow of samples. One sample is described by a value and associated to an absolute time. Modeling in SystemC AMS TDF is describing the behavior that produces each sample and managing the simulation settings. Understanding both concepts is important to get a good idea of what can be modeled and how is it modeled.

At first, let's describe the behavior of a SystemC AMS simulation. The designer has the freedom of the complexity and the granularity of the behavior, and he can benefit from the ability of mixed-signal systems description. The complexity because all the standard C++ functions are available, acquiring the experience of more than 30 years. Moreover, external C++ libraries can be included, allowing the use of shared experience, for example using a FFT computation. The granularity because of the ability of hierarchical description that allows the control of the level of details of a component, as illustrated in Fig. 3.3. A higher level of details allows to follow the intermediate sig-

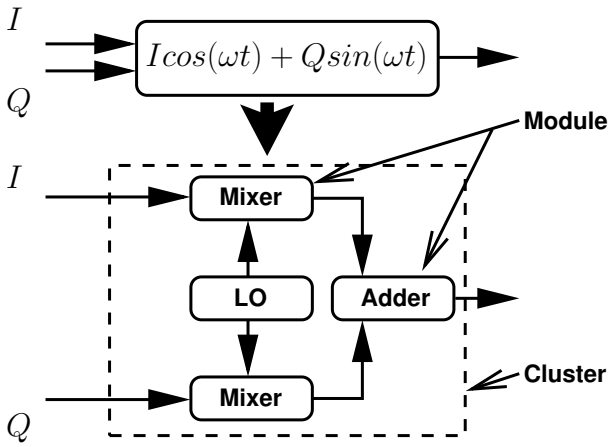

Fig. 3.3: Behavior description: user defined level of hierarchy.

nals, to analyze the effect of non-ideal components to the global behavior. The TDF components are called modules and the list of interconnected TDF modules is called a cluster. The cluster simulation plan is elaborated by a static scheduler thanks to the simulation settings. Finally, about the ability of mixed-signal systems description, the designer can easily connect TDF modules to Discrete Event (DE) SystemC digital modules. Actually, SystemC AMS is capable of time signal conversion between a constant step as in SystemC AMS TDF MoC and variable step as the SystemC DE MoC. This makes possible complex simulations with digital and analog components involving complete description of a microcontroller like the Wireless Sensor Network node that will be presented in Section 3.3.

Secondly, about managing the simulation settings, actually, they are all user-defined. In one hand, the designer has to verify the coherence of the simulation settings. It is a hard work of balancing between precision and simulation speed. It implies that the designer knows exactly the frequency of the application at each stage. In an other hand, the designer is supposed to understand

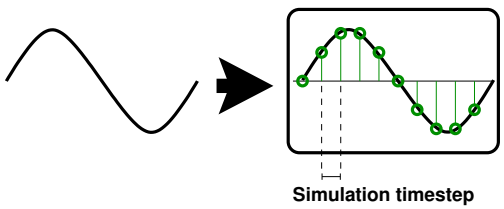

Fig. 3.4: Simulation settings: user defined simulation timestep.

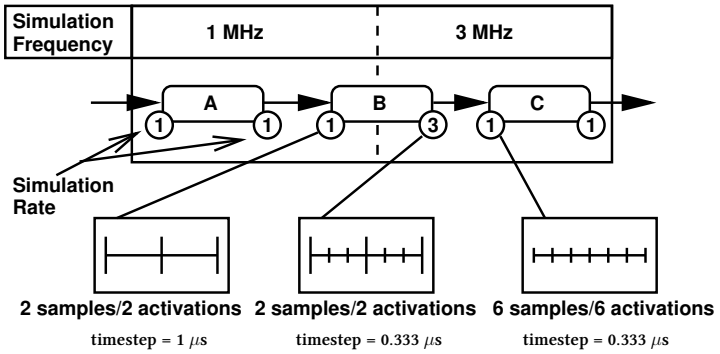

Fig. 3.5: Simulation settings: user defined simulation rate.

each stage of his own application. By managing himself the simulation settings, all is in control, the designer can use the knowledge to increase the speed of simulation.

There are three main simulation settings in SystemC AMS TDF: simulation timestep, simulation rate, sample delay. Each sample is generated in a constant user-defined time distance called simulation timestep as depicted in Fig. 3.4. In case of a behavior that describes both low-frequency and high-frequency signals in different subparts, a simulation rate can be attached to the input or output of the blocks to increase or decrease the samples data rate. The rate value set at input of a module corresponds to the number of available samples that can be read in one process activation. The rate value set at output of a module corresponds to the number of necessary samples that have to be written in one process activation. As presented in Fig. 3.5, a rate of 1 at the input of module B implies that in one process activation, one sample is available to be read from module B input. A rate of 3 at the output of module B implies that 3 samples have to be written in one process activation. Finally, rate 1 at input of module C implies that 1 sample is available at each process activation. This way, when 2 activations are processed in modules A and B, 6 activations are processed in module C. The process activations are automatically and statically scheduled during the elaboration procedure, before the simulation starts. For the example Fig. 3.5, the static scheduler can build an activation sequence "ABCCC". The designer needs to set one simulation timestep per clusters (in a chosen module definition) and to maintain the coherence between the simulated signal and the applied sampling at each module thanks to the simulation rate setting. With those informations, the scheduler will be able to propagate the timestep setting to all the connected modules of the cluster. A last user-defined simulation setting is delay. It can be related to the chosen behavior but also to a

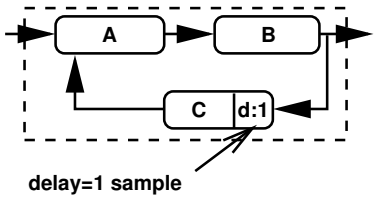

Fig. 3.6: Simulation settings: a delay for feedback looped systems simulation.

modeling constraint. As depicted in Fig. 3.6, a feedback loop needs to be delayed to make possible the static scheduling.

Appendix A presents TDF Model of Computation through the example of modeling an RF transmitter with its testbench. It illustrates practically the essential concepts of SystemC AMS TDF modeling and introduces the language syntax.

3.3 Wireless Sensor Network Node Model

As an example of mixed-signal systems modeling, we have chosen a Wireless Sensor Network (WSN) node. This section describes the ideal implementation of the WSN components that was used to run a first validation process.

A wireless sensor network is a network of autonomous devices that uses sensors to monitor environmental conditions such as temperature, sound, motion and contaminants at distributed geographical locations. As shown in Fig. 3.7, a typical node contains an ADC for analog to digital conversion, a microcontroller executing the embedded application (data processing) and the RF transceiver for wireless communication.

The behavior of the implemented WSN node consists of simply propagating acquired data to RF communication device. The ADC which will be detailed in Section 3.3.1 converts analog measures read from the input to 8-bit digital values. An ATMEGA128 [atm] microcontroller, presented in Section 3.3.2, reads the 8-bit value from an input port and executes instructions to serialize the read data. The serial bitstream is sent from a 1 bit port of the microcontroller to the RF transceiver, Section 3.3.3. The signal is finally sent using a QPSK modulation, this is the RF output of the node.

The following Sections 3.3.1 to 3.3.3 present the details of the chosen architecture and the SystemC AMS implementation. The implementation is described with listing examples to point the

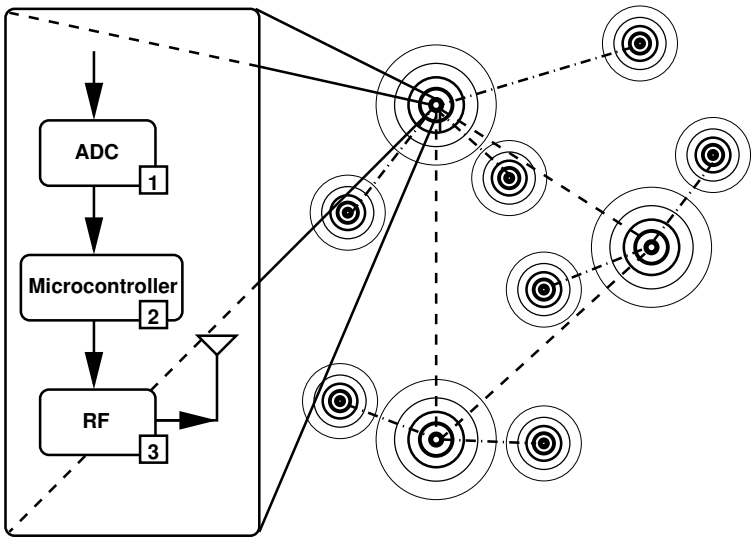

Fig. 3.7: A wireless sensor network node.

behavior description and with block schematics to visualize the hierarchy and the simulation settings of the whole component.

3.3.1 ADC Model

The ADC shown in Fig. 3.8 is a 1-bit second order $\Sigma\Delta$ modulator with an oversampling rate (OSR) of 64 and with delayed return-to-zero feedback [Aboushady02] and a decimator using third order FIR filters [Aboushady01a].

The implementation is presented in Fig. 3.9 depicting the whole ADC cluster, the $\Sigma\Delta$ modulator contains two integrator/subtractor modules, a DAC, one 1-bit ADC operating as a sampler and 1-bit quantifier, and a delay module to resolve the simulation scheduling issue as described in Section 3.2. As it has been said in Section 3.2, the timestep setting has to be called once in a cluster. We have chosen to set the timestep in the first module to locate easily this function. However, the value of the timestep: T_{simu} is provided by the top-level to maintain the genericity of the component. The analog modulator is simulated with a certain amount of samples related to the modulator sample time: T_s. The relation is $T_{simu} = T_s/simrate$, where $simrate$ is the simulation rate that is specified by the top-level. As the signal is becoming digital at the output of the modulator, the samples are simply bit values duplicated "$simrate$" times. As the ADC contains an analog part, the

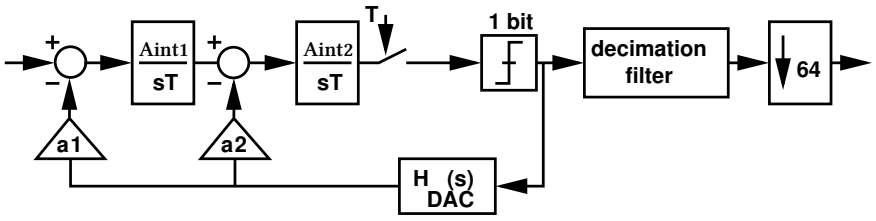

Fig. 3.8: Second order lowpass $\Sigma\Delta$ continuous-time modulator.

modulator and a digital part, the decimator, we introduced a rate transition module downsampling the redundant samples by the simulation rate. Finally, the decimator is a digital signal processing component, we have chosen to describe it with SystemC AMS like an analog component. This way, we are able to trace the intermediate signal like an analog component, with a constant timestep, to process an FFT in order to validate the behavior. By using a SystemC model, a clock would wake up the module behavior, with SystemC AMS the module is automatically woke up by the scheduler.

Listing 3.1 presents the $\Sigma\Delta$ integrator-subtractor ideal model source code. We also model the subtraction of the feedback signal from the input signal, applying a gain a_i to the feedback signal. The integration transfer function is modeled using the Laplace Transfer Function (Lines 8, 9 and 20). As a general transfer function is described by a numerator and a denominator:

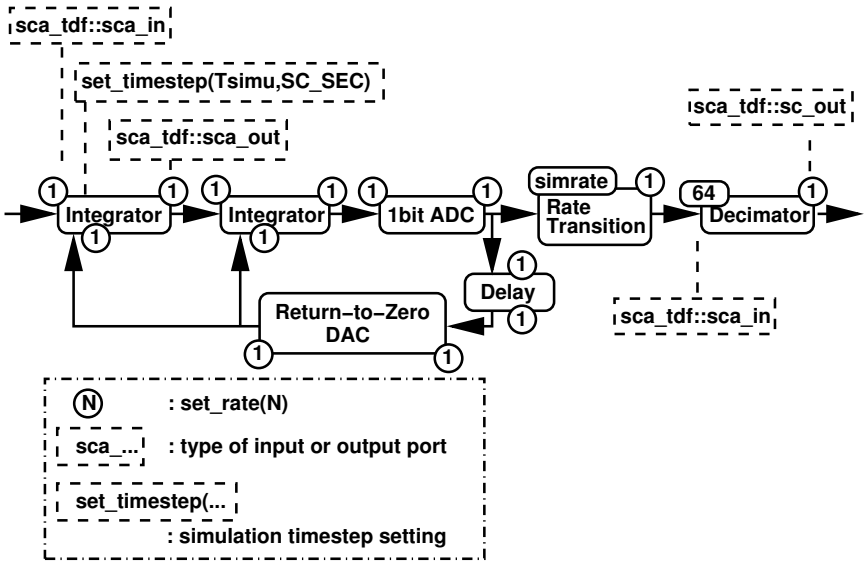

Fig. 3.9: ADC SystemC AMS TDF model with detailed simulation time step, input/output rates and type of interfaces.

Listing 3.1: The $\Sigma\Delta$ integrator description in SystemC AMS TDF.

```
1  #ifndef INTEGRATOR_H
2  #define INTEGRATOR_H
3
4  SCA_TDF_MODULE(integrator){
5    sca_tdf::sca_in<double> in1,in2; // Inputs declaration
6    sca_tdf::sca_out<double> out; // Output declaration
7
8    sca_vector<double> NUM,DEN; // Numerator and denominator declaration
9    sca_tdf::sca_ltf_nd ltf1; // Laplace transform function structure declaration
10   double ai; // Feedback gain
11
12   integrator(sc_core::sc_module_name,double Aint,double ai,double fs) // Constructor of the module
13       :in1("in1"),in2("in2"),out("out"){ // Aint, ai in linear dimension, fs in Hz.
14     DEN (0) = 0.0;
15     DEN (1) = 1.0;
16     NUM (0) = Aint*fs; // Aint is the DC gain, fs the sampling frequency of the SD modulator
17     ai=this->ai; // ai is the SD feedback gain
18   }
19   void processing(){ // Behavior of the module
20     out.write(ltf1(NUM,DEN,in1.read()-ai*in2.read()));
21   }
22 };
23 #endif
```

$$H(s) = \frac{a_0 + a_1 s + ... + a_n s^n}{b_0 + b_1 s + ... + b_n s^n} \tag{3.1}$$

The ideal integrator transfer function is: $\frac{A_{int}}{sT}$. The numerator and denominator coefficient (Lines 14 to 16) are defined in the constructor of the module to be sure that the values will be specified.

3.3.2 Microcontroller Model

The microcontroller is an ATMEGA128 [atm], an AVR family device from ATMEL. It is a RISC microcontroller with 16-bit wide instructions and a flash program memory of 128 Kbytes.

The microcontroller is implemented as an instruction set simulator in a Bit Cycle Accurate (BCA) SystemC model. This pure SystemC model is described by the Discrete Event MoC, the simulation step is variable and automatically computed by an algorithm in the simulator. As said

in Section 3.2, TDF to DE signal converter are available and has been used to connect this DE component to the TDF components.

The model can read a binary file generated by a C compiled for ATMEGA128 source code like if it was uploaded to a hardware microcontroller. This way, it has been possible to develop embedded software to process the behavior of the communication protocol. The presented work focused on the physical layer and did not investigate deeply in the direction of embedded software implementation, the only work of embedded software is to serialize the acquired data to send it to the RF transmitter. However, [Leveque10] presented a similar environment with the same analog models but with a more complex digital platform with SoCLib [soc] components (MIPS, RAM, Interconnect, ...) embedding the implementation of TDMA (Time division multiple access).

3.3.3 RF Transceiver Model

The RF transceiver [Normark04, Ravatin02] of the WSN node uses QPSK (Quadrature Phase-Shift Keying) modulation (Fig. 3.10) with a f_c carrier frequency and a $f_b = \frac{1}{T_b}$ data frequency. QPSK transmission is based on Eq. (3.2) to (3.4).

$$\begin{cases} I(t) = \{-1, 1\} \\ Q(t) = \{-1, 1\} \end{cases} \tag{3.2}$$

$$s(t) = I(t)\sqrt{\frac{1}{T_b}}\cos(2\pi f_c t) + Q(t)\sqrt{\frac{1}{T_b}}\sin(2\pi f_c t) \tag{3.3}$$

$$\begin{cases} \int_0^{2T_b} s(t)\sqrt{\frac{1}{T_b}} cos(2\pi f_c t)\,\mathrm{d}t = I(t) \\ \int_0^{2T_b} s(t)\sqrt{\frac{1}{T_b}} sin(2\pi f_c t)\,\mathrm{d}t = Q(t) \end{cases} \tag{3.4}$$

As presented in Eq. (3.2), the input data is a bitstream encoded in -1 and 1 values. Eq. (3.3) represents the ideal transmitted signal, Eq. (3.4) represents the reconstructed odd and even bitstreams in the receiver. Because the phase distance between cosine and sine waves is $\pi/2$, and because the distance between a sine or cosine wave and its opposite (multiplied by -1) is π, QPSK can modulate 4 symbols.

The QPSK transmitter (Fig. 3.10) consists of an encoder, a demultiplexer, two mixers, and a signal adder. The encoder shifts the voltage levels of the input bitstream according to the user-defined

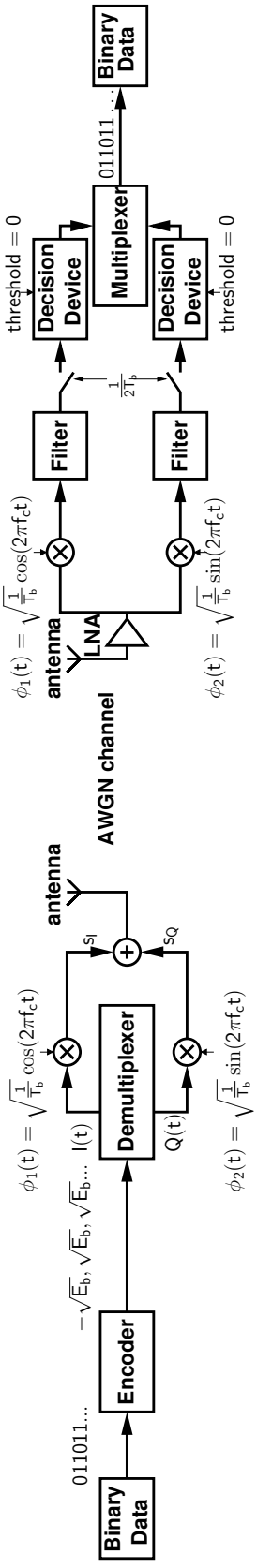

Fig. 3.10: QPSK RF transceiver.

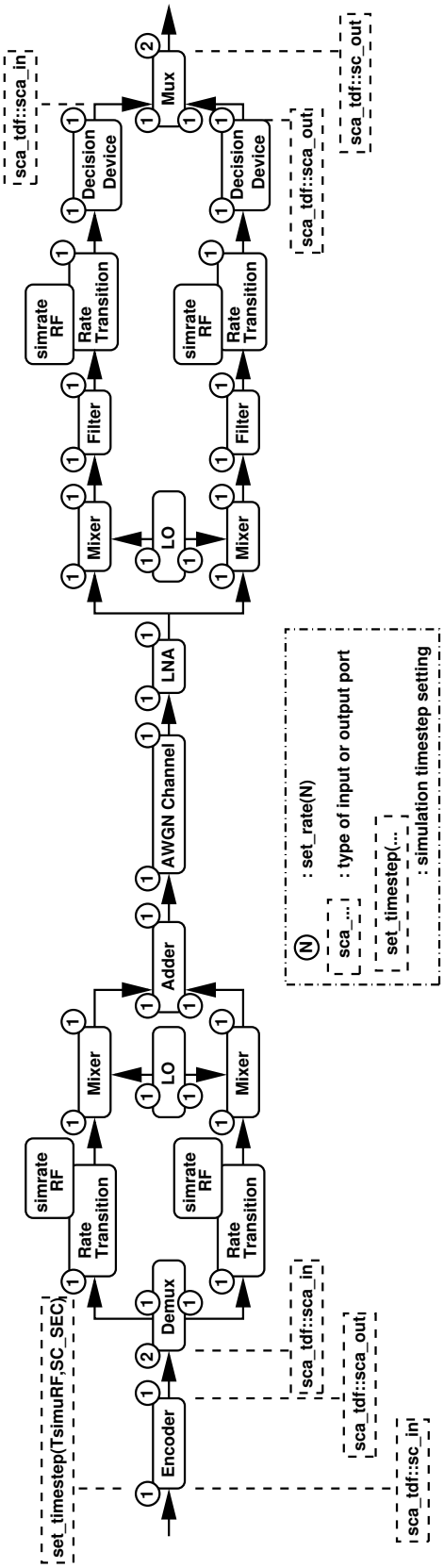

Fig. 3.11: QPSK RF transceiver SystemC AMS TDF model with detailed simulation time step, input/output rates and type of interfaces.

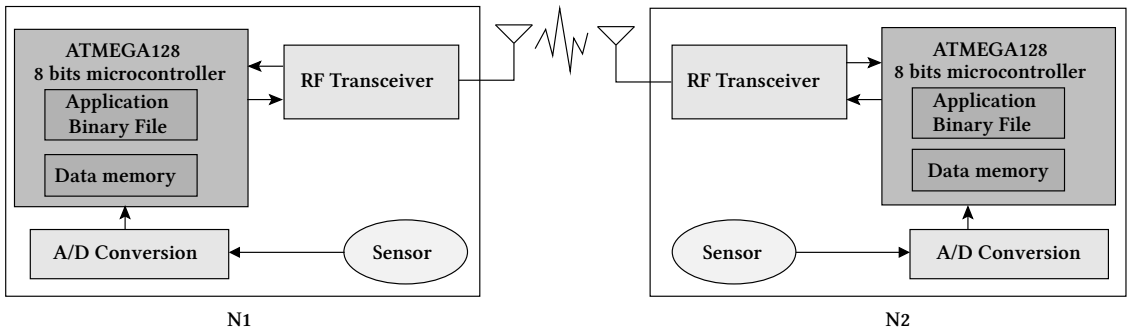

Fig. 3.12: A Wireless Sensor Network consisting of two nodes N1 and N2.

energy specification. Multiplied by a cosine or sine signal in the mixer, signals s_I and s_Q are combined in the adder, and the sum produces the QPSK-modulated signal. AWGN (Additive White Gaussian Noise) is taken into account in the RF communication channel model. From the QPSK receiver (Fig. 3.10) viewpoint, the signal is mixed with a cosine and sine wave and each component is integrated over the time. The integration of cosine and sine mixed components provides respectively the I and Q components informations. A decision device digitizes positive values to symbol '1' and negative values to symbol '0'. The signal is finally multiplexed and reconstructed.

The implementation is detailed in Fig. 3.11 as a block schematic representing the whole transceiver cluster. The simulation timestep has been specified in the first module of the cluster with a value T_{simuRF} specified by the top-level. The transmitter uses a demultiplexer (**Demux** module) to distribute even and odd bits to I and Q (In-phase and Quadrature paths). Two input samples of this module produce one output sample to the I path and one output sample to the Q path. Therefore, the **Demux** input rate is set to 2, and its outputs to 1. Next, the simulation timestep had to be adapted to RF signals of **LO** module thanks to simulation rate settings, we introduced a **Rate_Transition** module with a $simrate_{RF}$ parameter specified by the top-level. In the receiver, the opposite distribution of simulation rates is implemented: the $simrate_{RF}$ parameter is used to set the downsampling rate and the multiplexer has a rate of 2 at the output. As depicted, the transmitter input and the receiver output are digital signals (**sca_tdf::sc_in** and **sca_tdf::sc_out** types).

3.3.4 Simulation Results

For simulation purposes, in a first step, each component is validated separately, then the testbench presented in Fig. 3.12 is simulated. The configuration of the components implies to set some parameters. In the ADC, the bandwidth is set at 50 kHz, the gain values of $\Sigma\Delta$ feedback loop are set to a_1=2 and a_2=7/6. In the RF component, the carrier frequency is set to 2.4 GHz.

Considering the simulation settings, we consider that 10 samples per sine wave period are sufficient to represent correctly the analog signals. In the ADC, the simulation timestep, T_{simu}, is 10 times the modulator sampling frequency f_s:

$$f_s = 2 * OSR * BW = 6.4MHz \tag{3.5}$$

$$T_{simu} = 1/(10 * f_s) = 1.5625e^{-7}s \tag{3.6}$$

Where BW and OSR are the input signal bandwidth and the $\Sigma\Delta$ ADC oversampling ratio respectively. In the microcontroller, the input and output bitstream flowing at $f_b = 2.4MHz$ for digital communication with the RF block. As the signals at the input of the **Encoder** module and the output of the **Mux** module are 1-bit digital, they can be represented in SystemC AMS TDF by 1 sample per bit. As the bitstream of the microcontroller is synchronized with the input and output bitstream of the RF component, the simulation timestep located in the **Encoder** module is set a $T_{simuRF} = 1/f_b = 4.1667e^{-7}s$. We needed a simulation frequency expressed in Eq. (3.7) to represent the RF modulated signal.

$$f_{simuRF} = 24GHz \tag{3.7}$$

As the signal timestep is multiplied by 2 after the **Demux** module, a simulation rate has been set in the **Rate_Transition** modules following Eq. (3.8).

$$\begin{aligned} f_{simuRF} &= 10 * f_c = 10 * 1000 * f_b \\ simrate_{RF} &= 20000 \end{aligned} \tag{3.8}$$

We verify the accuracy of the simulation results with the following well known tests: For the ADC, a spectral analysis is used to compute the output Signal-to-Noise Ratio (SNR). We can observe

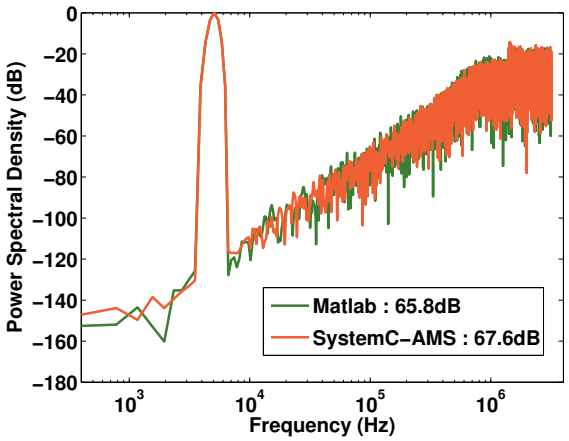

Fig. 3.13: $\Sigma\Delta$ modulator output spectrum. (BW=50kHz, OSR=64, N=16*1024 pts).

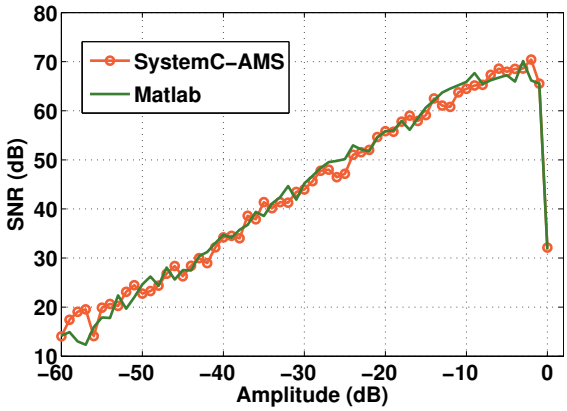

Fig. 3.14: $\Sigma\Delta$ modulator output SNR with respect to the sine input amplitude.

in Fig. 3.13, the output spectrum of the $\Sigma\Delta$, it validates the ADC behavior by comparing with a Matlab/Simulink similar model simulation result. Fig. 3.14 shows the $\Sigma\Delta$ ADC SNR versus the input amplitude. It can be shown that SystemC AMS and Matlab/Simulink models give very similar results.

In order to validate the RF transmission, we visualize the constellation and the Bit-Error Rate with respect to the signal-to-channel noise ratio. In Fig. 3.15, we show the constellation of an ideal QPSK transmission. Referring to Fig. 3.11, Bit-Error rate is the number of erroneous received bits divided by the number of transmitted bits. The transmission of up to $1e^6$ bits needs to be simulated

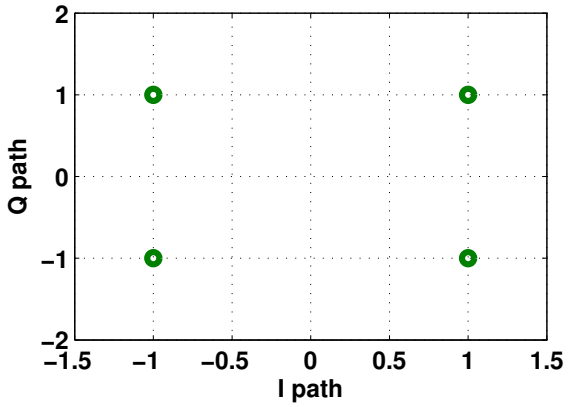

Fig. 3.15: Constellation of symbols received from an ideal QPSK transmission.

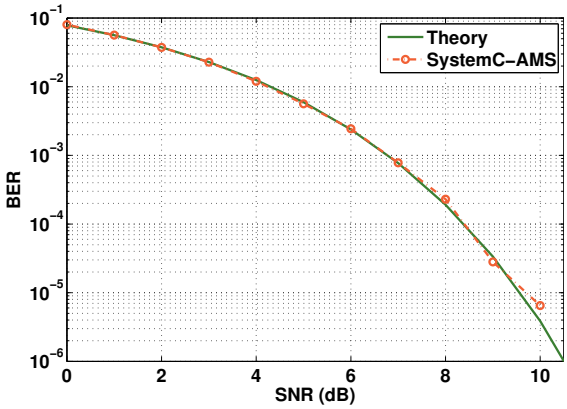

Fig. 3.16: Bit-Error Rate with respect to the SNR of a QPSK transmission through an AWGN channel.

to obtain a good estimation of the BER. A theoretical BER is computed from the Additive White Gaussian Noise (AWGN) characteristics and is compared with simulation results. Fig. 3.16 demonstrates a good matching between simulation and theoretical results of BER analysis produced by Eq. (3.9).

$$BER = \frac{1}{2} erfc(\frac{\sqrt{E_b}}{N_0}) = \frac{1}{\sqrt{\pi}} \int_{\sqrt{\frac{E_b}{N_0}}}^{\infty} e^{-z^2} dz \tag{3.9}$$

After the validation of the model behavior, we analyzed the simulation speed. We timed Matlab/Simulink vs. SystemC AMS simulations (Table 3.1), using equivalent models on both sides with the same parametrization and equivalent simulation settings. The performance is a bit better with SystemC AMS, but as the abstraction levels are similar, this is an expected result. However, a stronger argument is about the ability of SystemC AMS to perform an analog-mixed signal simulation. In fact, a simulation of communication between 2 nodes could not be performed with Matlab/Simulink as we did not have such a complex instruction set Bit Cycle Accurate (BCA) microcontroller model. This problem reveals the advantage of SystemC AMS simulation: we are able to simulate both digital and analog models simultaneously. Moreover, SystemC description language is widely used in digital part, the digital models of main components are already available [soc].

Table 3.1: ADC, RF transceiver and 2-node transmission SystemC AMS and Matlab models simulation speed results.

	Configuration	Simulation	Matlab	SystemC AMS
ADC	OSR=64 8 bits BW=50 kHz	1 ms 16*1024 pts	1.60 s	0.93 s
RF	f_c=2.4 GHz	416.67 μs 10^3 samples for digital part 10^7 samples for RF part	2 m 30	54 s
2-node transmission	Same settings	416.67 μs	–	3 m 1.65 s

3.4 Baseband Equivalent Modeling for Fast RF simulation

3.4.1 Baseband Equivalent Technique

The standard WSN simulation used so far shows that most of the time is spent in the simulation of the RF part. In fact, 24 Giga samples are needed to simulate 1 second of communication through the transceiver. To prevent the simulation time from becoming too prohibitive, the baseband equivalent representation can be used [Kundert99, Yee01, Chen05]. Considering an RF signal represented by:

$$x(t) = I_1 \cos(\omega_c t) + Q_1 \sin(\omega_c t) \tag{3.10}$$

Its baseband equivalent model is simply the terms: I_1 and Q_1. In order to have an accurate simulation of these two terms, we only need to oversample with respect to the data rate and not with respect to the carrier frequency. Therefore, this technique can significantly reduce the number of samples.

Two problems needed to be solved: How to propagate nonlinear effects between RF components, which cause harmonics in addition to the principal modulated signal. And how to manage adjacent frequencies, a problem met when simulating frequency offset or IIP_3 test. To solve the first problem, we decided to represent a modulated signal more accurately taking the DC and 2 harmonics into account (Eq. (3.11)), extending the baseband equivalent representation to (DC, I_1, I_2, I_3, Q_1, Q_2, Q_3).

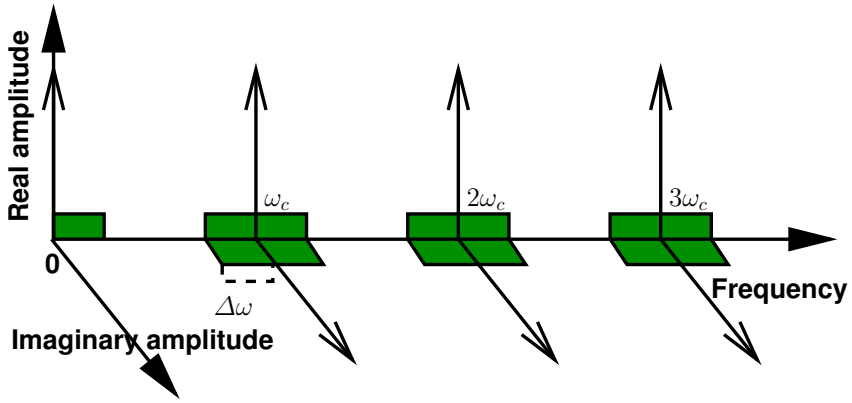

Fig. 3.17: Signal representation in baseband equivalent modeling, the segments delimited in green are the simulated bands of frequency when the signal is oversampled.

$$x(t) = DC + I_1\cos(\omega_c t) + I_2\cos(2\omega_c t) + I_3\cos(3\omega_c t) \\ +Q_1\sin(\omega_c t) + Q_2\sin(2\omega_c t) + Q_3\sin(3\omega_c t) \tag{3.11}$$

The second problem is about adjacent frequencies representation. According to Eq. (3.12),

$$\cos(\omega_c t + \Delta\omega t) = \cos(\Delta\omega t)\cos(\omega_c t) + \sin(\Delta\omega t)\sin(\omega_c t) \tag{3.12}$$

the offset $\Delta\omega$ applied to the signal frequency ω_c will be distributed to amplitudes I_1 and Q_1. In other words, when a signal

$$x(t) = Acos(\omega_c t)$$

is represented by ($DC = 0$, $I_1 = A$, $I_2 = 0$, $I_3 = 0$, $Q_1 = 0$, $Q_2 = 0$, $Q_3 = 0$),

$$x(t) = Acos((\omega_c + \Delta\omega)t)$$

will be represented by ($DC = 0$, $I_1 = Acos(\Delta\omega t)$, $I_2 = 0$, $I_3 = 0$, $Q_1 = Asin(\Delta\omega t)$, $Q_2 = 0$, $Q_3 = 0$).

That's why a high frequency signal with low frequency variations has to be oversampled to enlarge the band of each harmonic represented by baseband equivalent samples (Fig. 3.17). The gain of simulation speed is closely related to the bandwidth around the harmonics that is simulated, as it is represented in Fig. 3.17. With a minimal timestep, there is no possibility of representation of a frequency deviation. The maximal timestep is the value for which, the bandwidths meet each other in the middle ($\Delta\omega = \omega_c/2$. In this case, all the frequencies can be simulated from 0 to $3.5 * \omega_c$, but the simulation speed will be similar to a conventional non-baseband-equivalent simulation. The adapted timestep is determined by the kind of simulation considering or not some adjacent frequencies, or frequency deviations.

3.4.2 SystemC AMS Implementation

The implementation of baseband equivalent modeling is presented in Listing 3.2. The shift from scalar representation (*double* values) to vector (DC, I_1, I_2, I_3, Q_1, Q_2, Q_3) can be simply done with SystemC AMS, by taking advantage of C++. SystemC AMS allows to change the signal type that

Listing 3.2: Baseband equivalent implementation.

```
class BB{ // Baseband equivalent class definition
  public:
    double DC,I1,I2,I3,Q1,Q2,Q3; // The supported harmonics
    double w; // The current pulsation.
    ...
    BB operator* (double x) const{ // Multiplication operator
      BB z(DC*x,I1*x,I2*x,I3*x,Q1*x,Q2*x,Q3*x,w);
      return z;
    }
    BB operator* (BB x) const{
      BB z(
  DC*x.DC+I1*x.I1/2+I2*x.I2/2+I3*x.I3/2
              +Q1*x.Q1/2+Q2*x.Q2/2+Q3*x.Q3/2,
  ...
  Q3*x.DC+Q2*x.I1/2+Q1*x.I2/2
              +I2*x.Q1/2+I1*x.Q2/2+DC*x.Q3,
      w);
      return z;
    }
    BB operator+ (BB x) const{ // Addition operator
      BB z(
          DC+x.DC,
          I1+x.I1, I2+x.I2, I3+x.I3,
          Q1+x.Q1, Q2+x.Q2, Q3+x.Q3,
          w
          );
      return z;
    }
};
```

is transmitted via the ports. This type is, in C++ terms, a class defining the data structure and the associated operators. The type **double** is the conventional signal representation, each sample is the value of the oscillating signal, the operators "+" and "*" are floating point addition and multiplication. We implemented a class "BB" defining the baseband equivalent representation data structure and the associated operators "+" and "*", to be able to make every simple mathematical operation needed in RF modules.

The **BB** class is the baseband equivalent representation of a signal, each sample is a signal amplitudes vector. Operator "+" is a simple floating point addition, Eq. (3.13) illustrates this affirmation.

Table 3.2: ADC, RF transceiver and 2-node transmission SystemC AMS and Matlab models simulation speed results.

Configuration	Simulation	SystemC AMS conventional model	SystemC AMS baseband equivalent model
f_c=2.4 GHz	416.67 μs 10^3 samples for digital part	46.7 s	0.016 s
	416.67 ms 10^6 samples for digital part	40572.1 s = 11h16m12.1 s	5.9 s

$$Acos(\omega_c t) + Bcos(\omega_c t) = (A + B)cos(\omega_c t) \tag{3.13}$$

Operator "*" is a bit more complicated. To simplify the explaination, we present in Eq. (3.14) a first harmonic multiplication.

$$Acos(\omega_c t) * Bcos(\omega_c t) = \frac{AB}{2} + \frac{AB}{2}cos(2\omega_c t) \tag{3.14}$$

In fact, when multiplying two signals at the same frequency, the result is a sum of the second harmonic tone with the DC tone. For this example of one harmonic multiplication, the baseband equivalent representation is:

$$(0, A, 0, 0, 0, 0, 0) * (0, B, 0, 0, 0, 0, 0) = (\frac{AB}{2}, 0, \frac{AB}{2}, 0, 0, 0, 0) \tag{3.15}$$

We implemented the complete computation of the baseband equivalent multiplication operator for the three complex harmonics. Thus, rather than computing each module behavior in baseband equivalent form, the only modification to be done is to change the signal type of **sca_tdf::sca_in** and **sca_tdf::sca_out** module ports from **double** to **BB**. As shown in Listing 3.2, a class called **BB** has been defined implementing the vector and related operators.

In terms of simulation results, the baseband equivalent implementation has been used in the RF component of the WSN node model, achieving the same performance as the conventional model. The simulation speed has been clearly improved as suggests Table 3.2. While generating 10^6 samples required a 11h of simulation, with baseband equivalent modeling, it has been generated in 5.9 seconds.

3.5 Conclusion

The SystemC AMS language has been described in detail, revealing the ability of Mixed Signal modeling. Moreover, the language is particularly well suited for architectural modeling of complex mixed signal systems, as it has been used to describe a WSN node, containing a microcontroller, a $\Sigma\Delta$ ADC and an RF transceiver. Using SystemC AMS, it was possible to simulate the communication between two WSN nodes. It has also been shown that SystemC AMS can easily embed new models to speed up RF simulations with baseband equivalent technique.

4

Refined Behavioral Modeling of Analog and RF Components

4.1 Introduction

The choice of an abstraction level is always a compromise between the precision of the models and the simulation speed. In this chapter, we propose to incorporate the non-idealities of each component into the system-level models. This way, a fast simulation with more precise models can be operated. Such refined models are implemented to be user defined and as generic as possible.

We define three different categories of components with different model refinement characteristics.

- In Section 4.2, we define the low frequency "analog component" category and its model refinement characteristics.
- In Section 4.3, we define the high frequency "RF component" category and its model refinement characteristics.
- In Section 4.4, we present a third category, "sine wave source component" and its model refinement characteristics.

4.2 Model Refinement of Analog Components

A first refined model is for general low frequency analog components, typically for an amplifier or a filter. It is is described with model refinement incorporating a description of the Static Gain A_0, Poles, Zeros and Noise. Fig. 4.1 represents the refined model that is configured by used defined performance.

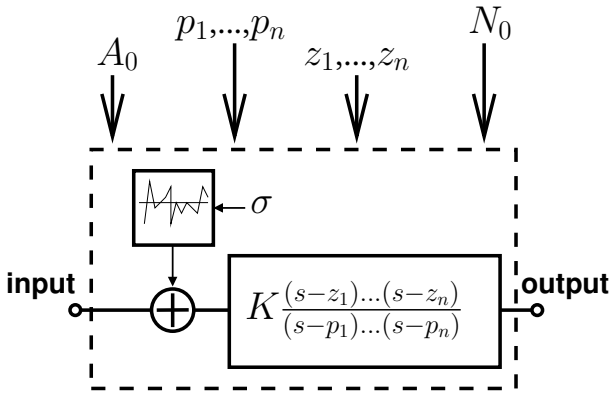

Fig. 4.1: Analog component refined model incorporating Static Gain, Poles, Zeros and Noise.

4.2.1 Gain

The gain is expressed by a transfer function built with the user defined parameters: A_0, Poles, Zeros. The parameter A_0 is translated to K as presented in Eq. (4.1), because of the need of expressing the transfer function as Eq. (4.2) for SystemC AMS implementation.

$$K = A_0 \frac{p_1 ... p_n}{z_1 ... z_n} \tag{4.1}$$

$$H(s) = K \frac{(s - z_1)...(s - z_n)}{(s - p_1)...(s - p_n)} \tag{4.2}$$

4.2.2 Noise

The noise is represented by N_0 that is the spectral density of the noise power. This parameter is translated to standard deviation: σ, as described in Eq. (4.3), that is the parameter that characterizes the amount of a normal random noise.

$$\sigma = \sqrt{\frac{N_0}{2} f_{simu}} \tag{4.3}$$

The equation distributes the spectral density N_0 into the whole band that is simulated: $fsimu$, because the noise power is supposed constant along the simulation bandwidth.

4.2.3 Implementation

The refined model of analog components is implemented as expressed in Listing 4.1. The constructor, Lines 11 to 21, is declared getting, as input parameter, the described gain and noise perfor-

Listing 4.1: Refined analog component model in Systemc AMS TDF.

```
1 #ifndef ANALOG_COMPONENT_H
2 #define ANALOG_COMPONENT_H
3 SCA_TDF_MODULE(analog_component){
4   sca_tdf::sca_in< double >in; // Input declaration
5   sca_tdf::sca_out< double >out; // Output declaration
6
7   sca_util::sca_vector <sca_complex>Z,P; // Zeros and poles declaration
8   double K;
9   sca_tdf::sca_ltf_zp ltf1;
10  double sigma, N0, A0;
11  analog_component(sc_core::sc_module_name,
12      const sca_vector<sca_complex> &Z, // Zeros in rad
13      const sca_vector<sca_complex> &P, // Poles in rad
14      double A0,                        // DC Gain in dBV
15      double N0)                        // Added Noise in V^2/Hz
16      :in("in"),out("out"){
17    this->A0=A0;
18    this->Z=Z;
19    this->P=P;
20    this->N0=N0;
21  }
22  void initialize(){
23    sigma=sqrt((N0/2)/get_timestep().to_seconds()); // The standard deviation
24    K=pow(10.0,A0/20); // Constant multiplicator
25    for(unsigned int i=0;i<P.length();i++)
26      K*=abs(P(i));
27    for(unsigned int i=0;i<Z.length();i++)
28      K/=abs(Z(i));
29  }
30  void processing(){
31    out.write(ltf1(Z, P, in.read()+sigma*randn(), K));
32  }
33 };
34 #endif
```

mance. It translates these parameters to be ready to be processed at each timestep in the function **processing()** (Line 31). As the σ parameter needs the timestep value, it is described in the function **innitialize()** in Lines 22 to 29. This SystemC AMS standard function is automatically called before

the simulation and when the elaboration is finished. At this point, the simulation timestep is known into each module of the cluster, it can be extracted with **get_timestep().to_seconds()**.

4.2.4 Results

Fig. 4.2 presents the comparison of the AC simulation between an ideal model and a 2 poles/2 zeros refined model of the integrator designed for a $\Sigma\Delta$ modulator with a $50kHz$ of bandwidth. As it will be described in details in Section 6.3, one potential problem of integrator design is the presence of a pole close to the transition frequency f_T, the frequency for which the gain is 0 dB. In the results presented in Fig. 4.2, we have plot the ideal and the 2 poles/2 zeros refined model of integrator frequency response. This way, the performance is precisely described in a system level allowing a fast simulation.

The validation of the implementation has been done by comparing the AC simulation of Matlab/Simulink versus SystemC AMS. The result perfectly matched.

For the noise validation, a signal with constant value 0 is connected at the input and the transfer function is configured to be $H(s) = 1$. The sum of output samples is processed, the noise behavior is validated as this sum equals to the variance $var = \sigma^2$. This way, we could reconstruct the performance N_0 that has been specified.

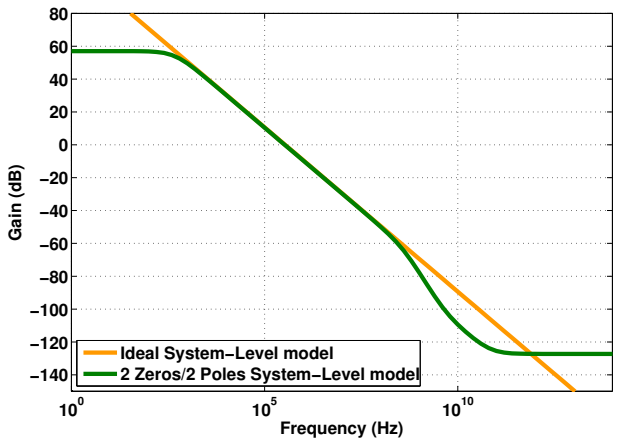

Fig. 4.2: Frequency response of a $\Sigma\Delta$ GmC integrator designed for BW=50 kHz, comparison between the ideal model and a 2 poles/2 zeros model.

4.3 Model Refinement of RF Components

The second refined model describes precisely the RF components at a system level. The RF components are typically an RF Mixer, a LNA, a PA. The RF refined model incorporates Power Gain, Noise Figure (NF), Third-order Input Intercept Point (IIP_3), Input/Output Resistances (R_i,R_o). Fig. 4.3 represents this generic RF component for which the performance and specifications can be configured by the user.

4.3.1 Gain

The first performance is the Power Gain, expressed in dB. The refined model operates a translation of this value into a voltage gain following Eq. (4.4), supposing a load/output and source/input resistances perfect matching.

$$a_1 = \sqrt{\frac{4G_pR_o}{R_i}} \tag{4.4}$$

4.3.2 Noise

The RF performance that characterizes the noise is Noise Figure, expressed in dB. Following Eq. (4.5), we could extract the added input noise to the component expressed as spectral density N_0 (Eq. (4.6).

$$NF = \frac{SNR_{IN}}{SNR_{OUT}} \tag{4.5}$$

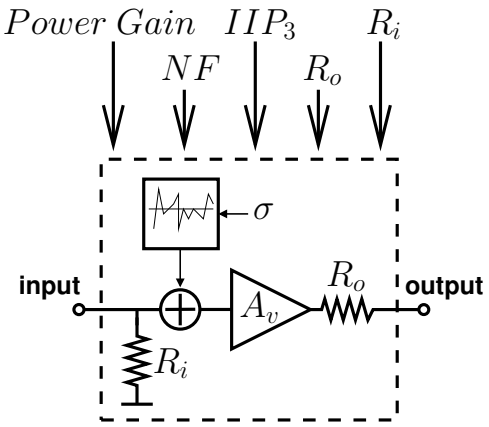

Fig. 4.3: RF component refined model incorporating Power Gain, NF, IIP_3 and Input/Output Resistance.

$$N_0 = KT(NF - 1) \tag{4.6}$$

This value is the same as in previous section, so we could use Eq. (4.3) to extract the σ parameter.

4.3.3 Nonlinearity

The nonlinearity is described by the IIP_3 parameter, expressed in dBm. Eq. (4.7) permitted to extract the a_3 parameter that characterizes the nonlinearity into the model.

$$a_3 = \frac{4a_1}{3IIP_3^2} \tag{4.7}$$

The behavior of a nonlinear gain is implemented by Eq. (4.8)

$$v_{out} = a_1 v_{in} - a_3 v_{in}^3 \tag{4.8}$$

4.3.4 Implementation

The implementation of the RF generic component is presented in Listing 4.2. At first, the constructor Lines 11 to 24 make the translation of the performance into parameters for the behavior description. Secondly, the **initialize** function, automatically called before the simulation, at the end of elaboration process, gets the simulation timestep to compute the σ parameter. Finally, the **processing()** function, automatically called at each timestep, for samples generation, incorporates the parametrized non-ideal behavior.

4.3.5 Results

The result of the implementation of the generic RF component is the validation of the refined behavior. This way, we verified the performance after a system-level simulation by comparing with the defined performance into a refined RF model. Fig. 4.4 plots the output spectrum of an RF component during a two-tone test. It images the non-ideality of an RF component by revealing the nonlinearity effect, as the intermodulation products appeared, and the effect of noise, as we can visualize a noise floor. Fig. 4.5 plots the amplitude of fundamental harmonic and 3^{rd} order intermodulation product (IM_3) in relation to the input amplitude.

Listing 4.2: Refined RF component model in SystemC AMS TDF.

```
#ifndef RF_COMPONENT_H
#define RF_COMPONENT_H

SCA_TDF_MODULE (rf_component){
  sca_tdf::sca_in < double >in; // Input declaration
  sca_tdf::sca_out < double >out; // Output declaration

  double nf, gain_power_db, iip3, rin, rout;
  double a1, a3, sigma;

  rf_component(sc_core::sc_module_name,
      double gain_power_db, // Power Gain in dB
      double iip3,          // IIP3 in dBm
      double nf,            // Noise Figure in dB
      double rin,           // Input resistance in Ohm
      double rout)          // Output resistance in Ohm
      :in("in"),out("out"){
    this->rin=rin;
    this->rout=rout;
    this->gain_power_db=gain_power_db;
    this->iip3=iip3;
    this->nf=nf;
    srand (time(NULL));
  }
  void initialize(){
    double f  = pow(10,nf/10);
    double N0 = 4*(f-1)*K*T*50;
    sigma=sqrt((N0/2)/get_timestep().to_seconds());
    double gain_power=pow(10,gain_power_db/10);
    a1 = sqrt(4*gain_power*rout/rin);
    double AIP3=undbm(iip3);
    a3 = a1/(3*pow(AIP3,2)/4);
  }
  void processing (){
    double input = in.read()+sigma*randn();
    out.write (a1*input-a3*pow(input,3));
  }
};
```

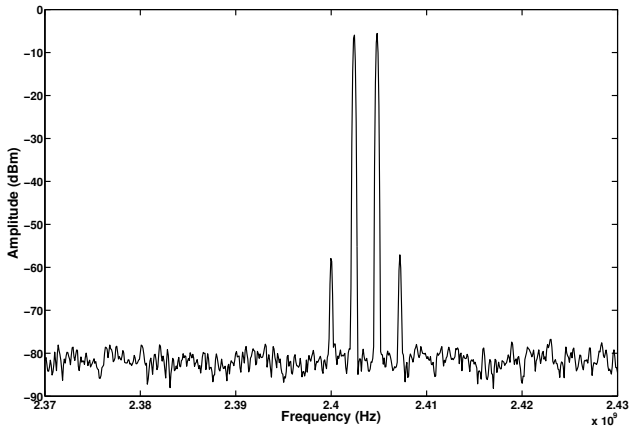

Fig. 4.4: Spectrum of a characterized low noise amplifier output.

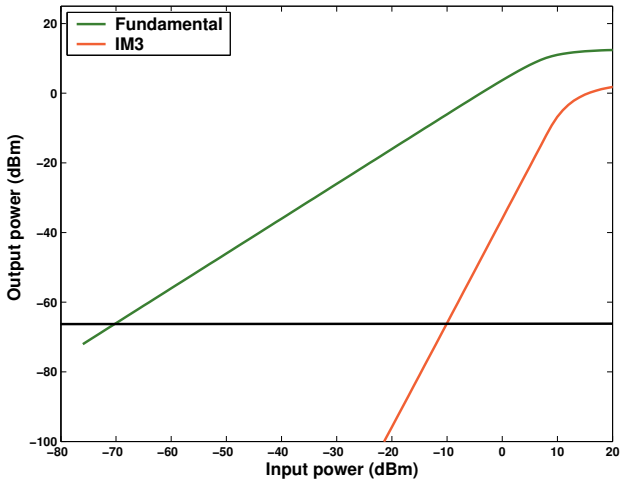

Fig. 4.5: Characterized LNA nonlinearity and noise effect analysis in respect to input power.

4.4 Model Refinement of Sine Wave Source Component

The last refined component that has been implemented is sine wave source model. Fig. 4.6 presents the configurable refined sine wave source component, incorporating the non-idealities of this component into the model: DC Offset, Frequency Offset and Phase Mismatch. The model is implemented with a sine and a cosine output to incorporate mismatch non-idealities.

4.4.1 Non-idealities

DC Offset is a constant value that is added to the oscillating signal as suggests Eq. (4.9)

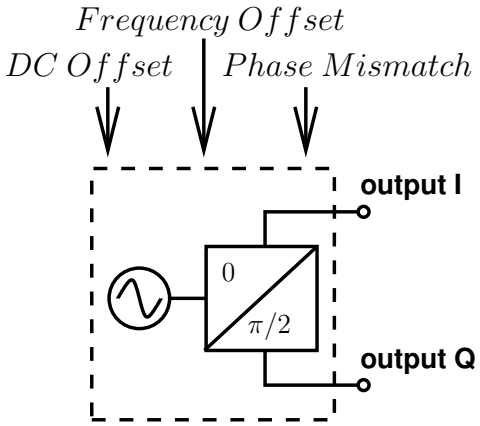

Fig. 4.6: Sine wave source component refined model incorporating DC Offset, Frequency Offset and Phase Mismatch.

$$v_{out} = DC_{Offset} + cos(2\pi f_c t) \tag{4.9}$$

Frequency Offset expressed by Δf in Eq. (4.10), is a deviation of the oscillation frequency.

$$v_{out} = cos(2\pi(f_c + \Delta f)t) \tag{4.10}$$

Phase Mismatch expressed by Φ in Eq. (4.11) represents the mismatch of the phase between both in-phase and in-quadrature signals.

$$\begin{cases} v_{outI} = cos(2\pi f_c t + \Phi/2) \\ v_{outQ} = sin(2\pi f_c t - \Phi/2) \end{cases} \tag{4.11}$$

4.4.2 Implementation

The SystemC AMS TDF implementation, presented in Listing 4.3, declares the constructor, Lines 11 to 18, that gets the user-defined performance and specifications. Next, some functions Lines 19, 22 and 25 give the ability to the user to modify dynamically the non-ideality parameters. Finally, Lines 28 to 33 present the **processing()** function that operates the behavior on each timestep.

4.4.3 Results

The sine wave source refined model is useful for analyzing the non-ideality of a Local Oscillator (LO) in an RF transmission. The refined model has been used in the QPSK transceiver presented

Listing 4.3: Refined sine wave source component model in SystemC AMS TDF.

```
#ifndef SWS_COMPONENT_H
#define SWS_COMPONENT_H
SCA_TDF_MODULE(sws_component){
  sca_tdf::sca_out< double >outI;
  sca_tdf::sca_out< double >outQ;

  double phase_mismatch;
  double frequency_offset;
  double dc_offset;
  double fc;
  sws_component(sc_core::sc_module_name,
      double fc, double dc_offset, double frequency_offset, double phase_mismatch)
      :outI("outI"),outQ("outQ"){
    this->fc=fc;                              // Carrier frequency in Hz
    this->dc_offset=dc_offset;                // DC Offset
    this->frequency_offset=frequency_offset;  // Frequency Offset
    this->phase_mismatch=phase_mismatch;      // Phase Mismatch
  }
  void set_phase_mismatch(double ph_m){
    phase_mismatch=ph_m;
  }
  void set_frequency_offset(double f_o){
    frequency_offset=f_o;
  }
  void set_dc_offset(double dc_o){
    dc_offset=dc_o;
  }
  void processing(){
    outI.write(dc_offset+cos(2*M_PI*(fc+frequency_offset)*
          get_time().to_seconds()+phase_mismatch/2.0));
    outQ.write(dc_offset+sin(2*M_PI*(fc+frequency_offset)*
          get_time().to_seconds()-phase_mismatch/2.0));
  }};
#endif
```

in Fig. 3.10. We present in Fig. 4.7, the constellation of a non-ideal QPSK transmission. The ideal constellation has been presented in Fig. 3.15. Referring to Fig. 3.11, the axis from Fig. 4.7 are the In-phase and Quadrature signal at the output of the sampler. The constellation permits to visualize the deviation of the amplitude in relation to the analyzed non-ideality. When the deviation crosses one axis, the resulting data will contain errors. The number of errors can be observed with the Bit-Error rate.

We can also observe the effect of non-idealities on the Bit-Error rate of the overall QPSK transmission. Fig. 4.8 shows a comparison between a transmission with an ideal receiver and a transmis-

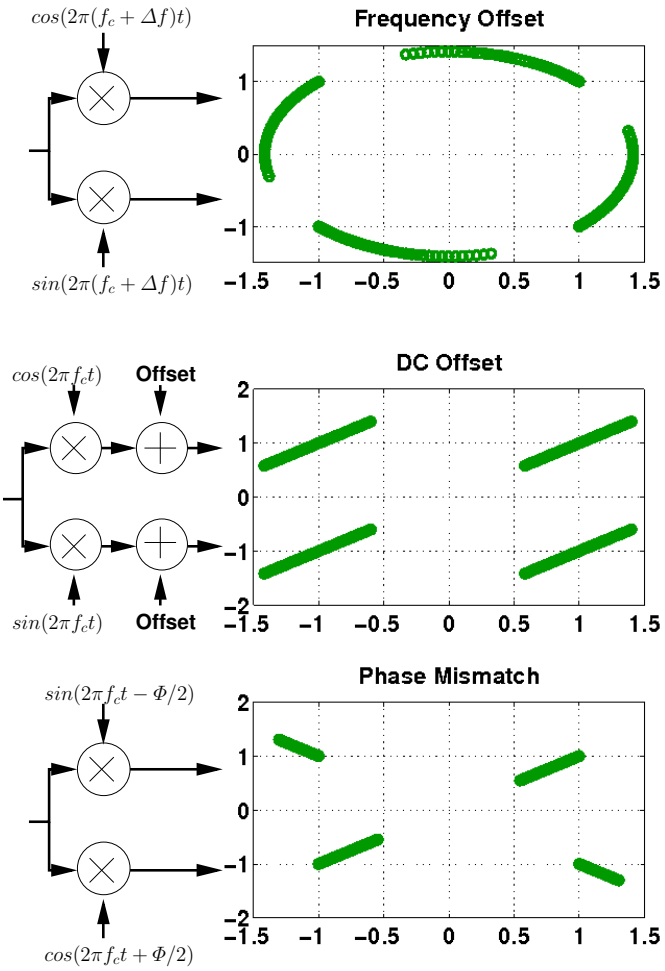

Fig. 4.7: Constellation of symbols received from QPSK transmission with a DC Offset, Frequency Offset, and Phase Mismatch.

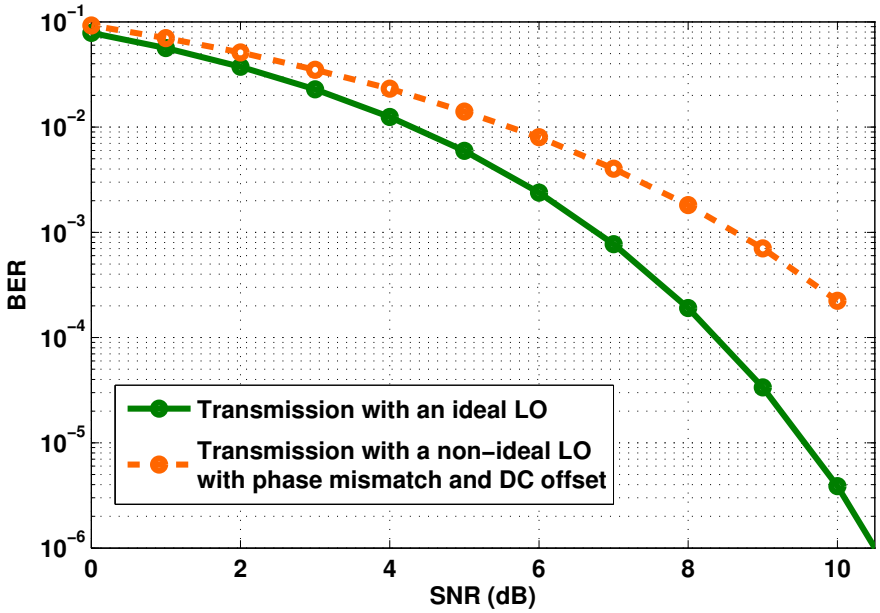

Fig. 4.8: Bit-Error Rate with respect to the SNR of a QPSK transmission with a non-ideal Local Oscillator through an AWGN channel.

sion with a non-ideal receiver. The non-ideality is located into the Local Oscillator as it operates with a DC offset and a phase mismatch.

4.5 Conclusion

In this chapter, the system-level models of analog and RF components have been refined to incorporate their non ideal behavior. We have defined three categories of analog and RF components depending on their non-idealities and their model refinement characteristics. Special attention has been given to ensure the genericity of the models. The proposed model refinement method has been applied to the QPSK transceiver model presented in Chapter 3 and the effect of circuit non-idealities on the overall system performance has been shown.

5

Analog and RF Circuit Analysis and Performance Evaluation

5.1 Introduction

In this chapter, we present our proposed methodology for the automatic evaluation of the linear and nonlinear performance of analog and RF circuits. In Section 5.2, we start with our motivation for having a dedicated performance evaluation tool. In Section 5.4, we present our linear performance evaluation method based on modified nodal analysis. In Section 5.5, we present our non-linear performance evaluation method based on Volterra series.

5.2 Motivation

As it has been observed in the previous chapter, making system-level simulations with models refined by the performance is useful for fast and accurate validation. The next step is to back-annotate those refined models with the results of circuit-level performance evaluation. As illustrated in Fig. 5.1, the proposed multi-level design flow with both top-down and bottom-up approaches requires communication between the levels of abstraction through the specifications and the performance.

The circuit-level performance is usually extracted by circuit-level simulation [Phelps00,Daems03], making the flow heterogeneous in terms of software and description languages. Therefore, a first motivation of implementing a systematic performance evaluation in a seamless environment with the system-level models is to embed the procedure in a unified flow and to benefit from simulation time improvement.

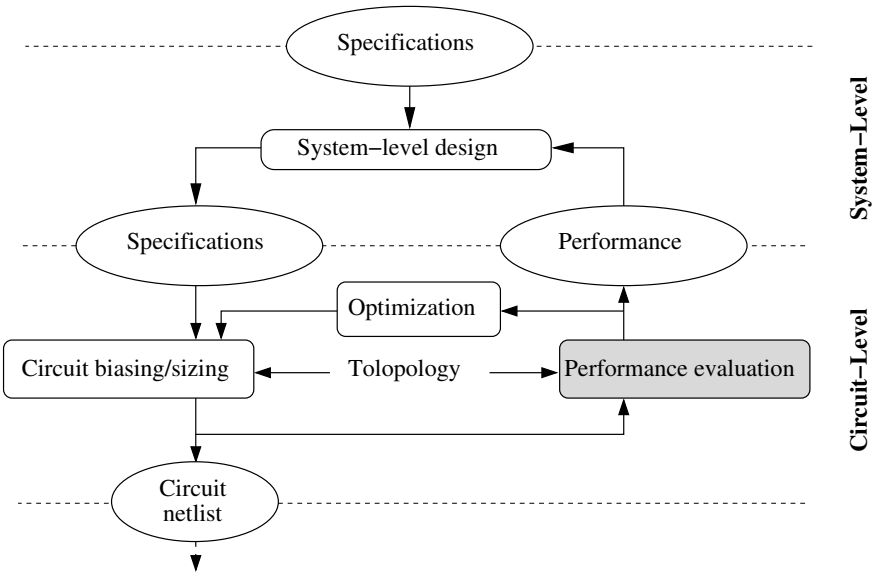

Fig. 5.1: Unified multi-level design environment for mixed signal systems.

The second motivation is to open the possibility to implement a systematic circuit design and optimization flow, in which the optimization will be strongly related to the performance evaluation. As the optimization will be implemented with multiple round-trips between circuit design and performance evaluation, it will benefit likewise from optimization time improvement in a seamless environment. The circuit design and optimization flow will be described in detail later in Chapter 6.

5.3 Modified Nodal Analysis

5.3.1 The MNA library based on Maxima

Modified Nodal Analysis (MNA) is a well known way to perform a small-signal analysis of a circuit [Ho75]. It manages a matrix of admittances, where each line is a Kirchoff's current law applied to one node and each column is the contribution of a potential voltage to the currents. In this work, we propose to build automatically the symbolic admittance matrix of a circuit and to provide it for the multi-level design flow.

Maxima is a computer algebra system (CAS) under GNU General Public License (GPL) [max]. It has been used for building the symbolic performance matrices. A set of circuit design functions have been implemented to make the linear performance evaluation more intuitive. The full docu-

mentation of the MNA library for Maxima is provided in Appendix B, the following paragraph and Table 5.1 resumes the implemented functions.

The designer can find the primitives for building a matrix of admittances like **addAdmittance** to add an admittance to the circuit, **addVCCS** to add a voltage controlled current source. These primitives are dedicated to the small-signal models of integrated components. Some higher-level functions are calling the primitives to build the complete small-signal model of a component: **addTransistor**, **addRFTransistor**, **addInductanceModelIDNW**, **addInductanceModelLVI**. Also, some performance evaluation functions are implemented to update the admittance matrix with the required input voltages and eventual short circuits to prepare the matrix for a performance evaluation: **setVin**, **setIin**, **shortCircuit**. Then, when the matrix can be solved in a symbolical representa-

Table 5.1: Some of the elements of the implemented MNA Maxima library sorted into categories.

Primitives	addAdmittance
	addVCCS
Macro functions for IC models	addTransistor
	addRFTransistor
	addInductanceModelIDNW
	addInductanceModelLVI
Inputs	setVin
	setIin
	shortCircuit
Outputs	getVout
	getIout
	getINoise
	getVNoise
Macro functions for performance evaluation	getVGain
	getIGain
	getOImpedance
	getIImpedance

tion, the designer can use functions to get the voltage of a node, a current through a component or the output noise current/voltage: **getVout**, **getIout**, **getINoise**, **getVNoise**. Finally, some top-level functions use the previous functions to compute in one command, the performance: **getVGain**, **getIGain**, **getOImpedance**, **getIImpedance**.

When the designer lets too much unknown variables to the symbolic representation, the performance cannot be extracted in a reasonable time. That's why the methodology is designed to generate the symbolic matrix with Maxima but solving this matrix in a procedure that can annotate the values of the small-signal parameters into the matrices of performance. This procedure will be detailed in the next chapter as it will expose all the procedures for circuit design.

5.3.2 Task Scheduling for a Systematic Circuit Analysis and Performance Evaluation

The use of the implemented Maxima library for generating matrix of performance follows a schedule that is presented in Fig. 5.2. The following paragraphs are describing each one of those scheduled tasks.

The first task is about integrating small-signal models of the targeted technology. This implies to describe the small-signal model with the primitives of the MNA library. The library already

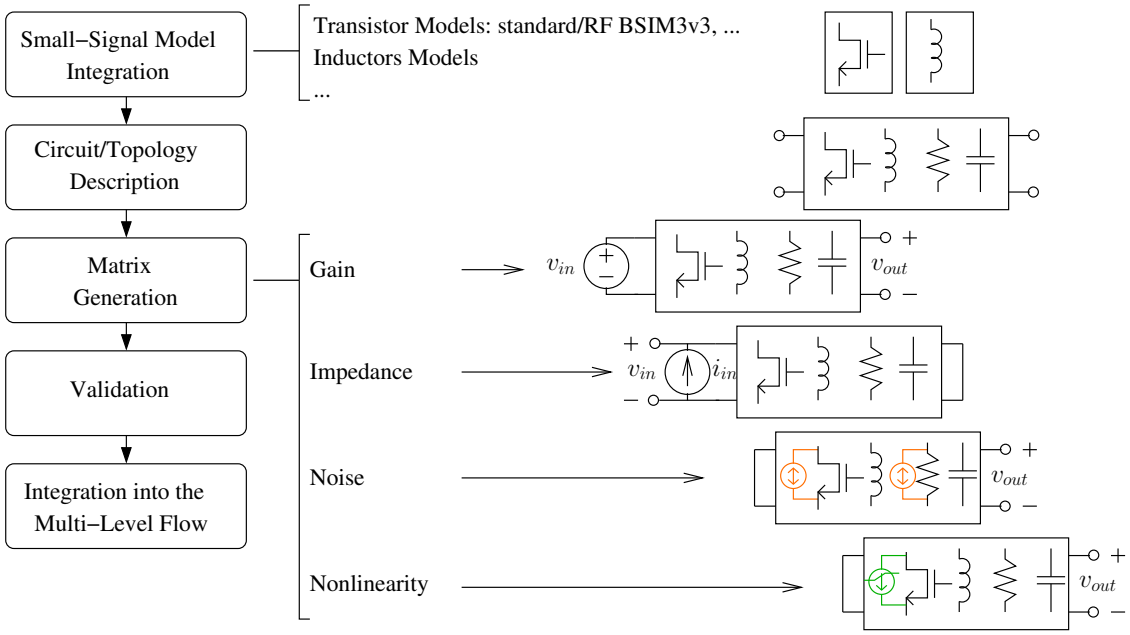

Fig. 5.2: Task Scheduling for a Systematic Circuit Analysis.

contains some standard models like standard or RF BSIM3v3 and standard integrated inductor model as specified in Table 5.1.

The second task is the description of the analyzed circuit. In this task, the designer builds the whole circuit netlist by instanciating the primitives or the macro functions that describe the small-signal model of components. The output is a symbolic matrix that describes the small-signal model of the circuit.

The third task is performance evaluation. This consist of setting the inputs and getting the output depending on the analyzed performance. For the Voltage Gain analysis, we set a voltage at the input of the circuit and get the voltage at the output. For the Input Impedance analysis, we set a current at the input of the circuit, we short-circuit the output and we get the voltage at the input. For the Output Voltage Noise analysis, we add current sources for each noisy component, we short-circuit the input of the circuit and we get the output voltage. This output will be the sum of each noise current source contribution to the output voltage. Finally, in a similar way, for the Nonlinearity analysis, we add current sources for each nonlinear component, we short-circuit the input of the circuit and we get the output voltage. This output will be the sum of each nonlinear current source contributions to the output voltage. As the nonlinearity involves more complex calculation, the detail is described in Section 5.5. The output is a symbolic matrix that describes the small-signal circuit and its testbench. For noise and nonlinearity, the output is a set of matrices for each contribution.

The fourth task is validation. In this task, the analyzed performance is compared to a circuit-level simulation. To achieve the task, the matrix is solved numerically thanks to the value of $s = 2\pi f$ and of the small-signal elements. The small-signal elements are extracted from CHAMS/CAIRO+ tools that will be used for the circuit sizing and optimization of the circuit-level design presented in Chapter 6

The fifth and final task is about the integration of the symbolic matrices into the circuit-level design flow to permit circuit-level optimization described in Chapter 6 and back-annotation of system-level models described in Chapter 7. As this integration is done in a C++ environment, the matrices are exported in a C++ language to keep the homogeneity of the design flow. As the matrices stay symbolic until we need a value of the analyzed performance, the performance evaluation

procedure does not need to call the matrices generation each time. This will be significant, in terms of execution time, when the performance evaluation procedure will be linked to an optimization procedure that will call, several times, the performance evaluation.

5.4 Linear Performance Evaluation

To illustrate the proposed formalism for circuit analysis, we expose a minimal example of a BSIM3v3 standard MOS transistor employed as a common source amplifier. At first, the small-signal model of the MOS transistor, presented in Fig. 5.3, is integrated in the MNA extended Maxima language as illustrated in Lines 2 to 15 of Listing 5.1. Secondly, the MOS transistor is instanciated, as described in Line 17 of Listing 5.1, the bulk is connected to the source for this example. The result is the matrix presented in Eq. (5.1)

$$\begin{pmatrix} (C_{sd}+C_{gd}+C_{bd})s+g_{ds} & -g_m-g_{ds} & (-C_m-C_{gd})s+g_m \\ & +(-C_{sd}+C_m-C_{bd})s & \\ (-C_{sd}-C_{bd})s-g_{ds} & g_m+g_{ds} & (C_m-C_{gs}-C_{gb})s-g_m \\ & +(C_{sd}-C_m+C_{gs}+C_{gb}+C_{bd})s & \\ -C_{gd}s & (-C_{gs}-C_{gb})s & (C_{gs}+C_{gd}+C_{gb})s \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_s \\ v_g \end{pmatrix} = \begin{pmatrix} i_d \\ i_s \\ i_g \end{pmatrix} \tag{5.1}$$

The next step of building the matrix of linear performance is completing the matrix with the testbench informations. The performance analysis formalization follows the procedure described in detail in Sections 5.4.1 to 5.4.3. We replaced each element of the admittance matrix by g_{xy}, to clarify

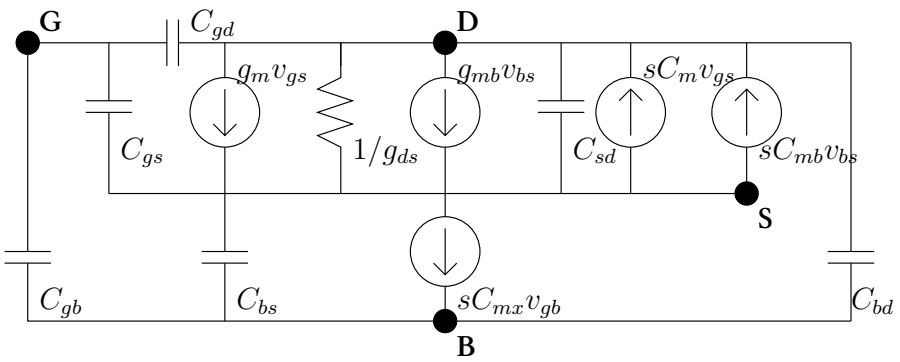

Fig. 5.3: CMOS transistor BSIM3v3 small-signal model.

Listing 5.1: Instructions that build the small-signal model matrix of a MOS transistor and solve the performance with MNA Maxima library.

```
1  batchload("AC_analysis.wxm");
2  addTransistor(A,nD,nG,nS,nB):=(
3    A:addAdmittance(A,GDS,nD,nS),
4    A:addVCCS(A,GM,nD,nS,nG,nS),
5    A:addVCCS(A,GMB,nD,nS,nB,nS),
6    A:addAdmittance(A,s*CSD,nD,nS),
7    A:addAdmittance(A,s*CGD,nG,nD),
8    A:addAdmittance(A,s*CBD,nB,nD),
9    A:addAdmittance(A,s*CGS,nG,nS),
10   A:addAdmittance(A,s*CGB,nG,nB),
11   A:addAdmittance(A,s*CBS,nB,nS),
12   A:addVCCS(A,-s*(CDB-CBD),nD,nS,nB,nS),
13   A:addVCCS(A,-s*(CDG-CGD),nD,nS,nG,nS),
14   A:addVCCS(A,s*(CBG-CGB),nS,nB,nG,nB)
15 )$
16 mat:init()$
17 mat:addTransistor(mat,D,G,S,S)$
18 gain:getVGain(mat,D,G,S,true);
19 zin:getIImpedance(mat,D,G,S,true);
20 onoise:getVNoise(mat,D,G,S,true);
```

the view of equations, the matrix becoming as presented in Eq. (5.2).

$$\begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_s \\ v_g \end{pmatrix} = \begin{pmatrix} i_d \\ i_s \\ i_g \end{pmatrix} \tag{5.2}$$

5.4.1 Voltage Gain

The voltage gain is computed by connecting an input voltage and declaring the output voltage. Line 18 of Listing 5.1 returns the voltage gain, building automatically the matrix of admittance with the testbench dedicated to voltage gain analysis. The resulting small-signal model is Fig. 5.4, in which, an input voltage is connected between nodes G and S. The testbench information of adding the input completes the admittance matrix (Eq. (5.3)), it derives from Eq. (5.4). The ideal voltage source is described by its voltage v_{in} and also by the current i_{in} that flows in the element. This way the left side matrix stays a square matrix.

$$\begin{pmatrix} g_{11} & g_{12} & g_{13} & 0 \\ g_{21} & g_{22} & g_{23} & 1 \\ g_{31} & g_{32} & g_{33} & -1 \\ 0 & -1 & 1 & 0 \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_s \\ v_g \\ i_{in} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} .v_{in} \tag{5.3}$$

$$\begin{cases} g_{11}v_d + g_{12}v_s + g_{13}v_g = 0 \\ g_{21}v_d + g_{22}v_s + g_{23}v_g + i_{in} = 0 \\ g_{31}v_d + g_{32}v_s + g_{33}v_g - i_{in} = 0 \\ v_g - v_s = v_{in} \end{cases} \tag{5.4}$$

Next, the transistor drain-source voltage is declared as the required output for the voltage gain computation. The testbench information of declaring the potential difference between voltages of nodes D ans S completes the matrix (Eq. (5.5)), it derives from Eq. (5.6)

$$\begin{pmatrix} g_{11} & g_{12} & g_{13} & 0 & 0 \\ g_{21} & g_{22} & g_{23} & 1 & 0 \\ g_{31} & g_{32} & g_{33} & -1 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_s \\ v_g \\ i_{in} \\ v_{out} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} .v_{in} \tag{5.5}$$

$$v_d - v_s - v_{out} = 0 \tag{5.6}$$

Finally, because one current law is linearly dependent to the others in a conservative system, we chose to remove the current law of the node connected to the ground: S. Also, we chose the reference of potential as the ground, so $v_s = 0$. Thus, the line of the node S and the column of the potential v_s are removed from the matrix (Eq. (5.7)).

$$\begin{pmatrix} g_{11} & g_{13} & 0 & 0 \\ g_{31} & g_{33} & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_g \\ i_{in} \\ v_{out} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} .v_{in} \tag{5.7}$$

The choice of describing v_{in} potential on the right side and v_{out} on the last column of left side

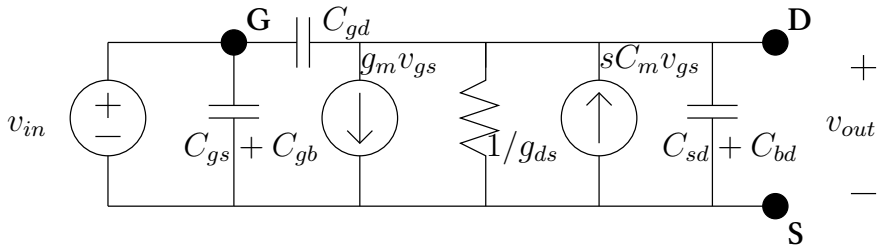

Fig. 5.4: The small-signal model for a transistor gain evaluation.

is for using the "echelon" Maxima command. This command calls a procedure of matrix Gaussian elimination, the result is presented in Eq. (5.8). Only the last line is entirely solved, but the procedure is faster when just one unknown is needed to be solved.

$$\begin{pmatrix} 1 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & C_{gd}s \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_g \\ i_{in} \\ v_{out} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ (C_{gs} + C_{gd} + C_{gb})s \\ ((C_m + C_{gd})s - g_m)/((C_{sd} + C_{gd} + C_{bd})s + g_{ds}) \end{pmatrix} .v_{in} \tag{5.8}$$

The last line of Eq. (5.8) represents Eq. (5.9). This way, the right side last element is the gain of the analyzed circuit.

$$v_{out} = \frac{(C_m + C_{gd})s - g_m}{(C_{sd} + C_{gd} + C_{bd})s + g_{ds}} v_{in} \tag{5.9}$$

Fig. 5.5 represents the testbench implemented informations connected to the transistor, for biasing correctly and for declaring the input and output elements. The graphic presents the gain AC analysis, validating the computed gain transfer function comparing to a simulator results.

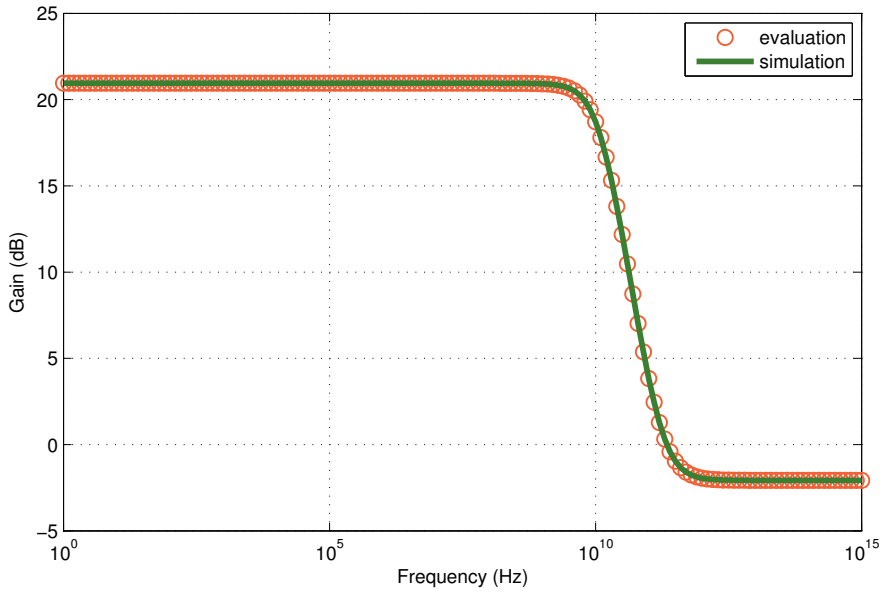

Fig. 5.5: CMOS transistor gain simulation to evaluation comparison.

5.4.2 Input Impedance

Input impedance is analyzed by short-circuiting the output, connecting an input current source and declaring the input voltage. Line 19 of Listing 5.1 returns the input impedance, building automatically the matrix of admittance with the testbench dedicated to input impedance analysis. The resulting small-signal model is Fig. 5.6, in which, an input current is connected between nodes G and S. The testbench information of adding the input completes the admittance matrix (Eq. (5.10)), it derives from Eq. (5.11).

$$\begin{pmatrix} g_{11}\ g_{12}\ g_{13} \\ g_{21}\ g_{22}\ g_{23} \\ g_{31}\ g_{32}\ g_{33} \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_s \\ v_g \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \end{pmatrix} .i_{in} \tag{5.10}$$

$$\begin{cases} g_{11}v_d + g_{12}v_s + g_{13}v_g = 0 \\ g_{21}v_d + g_{22}v_s + g_{23}v_g = i_{in} \\ g_{31}v_d + g_{32}v_s + g_{33}v_g = -i_{in} \end{cases} \tag{5.11}$$

Next, the transistor gate-source voltage is declared as the required input for the input impedance computation. The testbench information of declaring the potential difference between voltages of nodes G ans S completes the matrix (Eq. (5.12)), it derives from Eq. (5.13)

$$\begin{pmatrix} g_{11}\ g_{12}\ g_{13}\ 0 \\ g_{21}\ g_{22}\ g_{23}\ 0 \\ g_{31}\ g_{32}\ g_{33}\ 0 \\ 0\ -1\ 1\ -1 \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_s \\ v_g \\ v_{in} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ -1 \\ 0 \end{pmatrix} .i_{in} \tag{5.12}$$

$$v_g - v_s - v_{in} = 0 \tag{5.13}$$

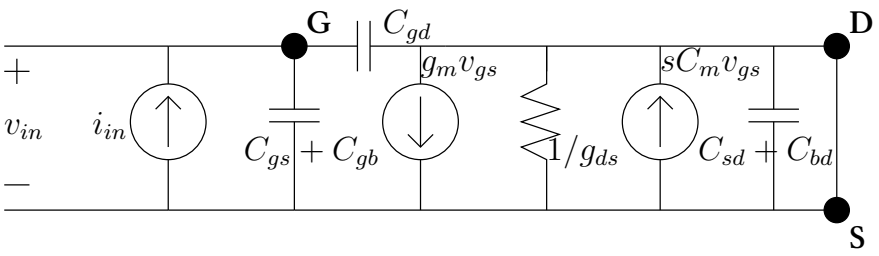

Fig. 5.6: The small-signal model for a transistor input impedance evaluation.

Finally, the line of the node S and the column of the potential v_s are removed from the matrix (Eq. (5.14)). Moreover, because D is short-circuited to S that is grounded, the line of the node D and the column of v_d are also removed from the matrix.

$$\begin{pmatrix} g_{33} & 0 \\ 1 & -1 \end{pmatrix} . \begin{pmatrix} v_g \\ v_{in} \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} .i_{in} \tag{5.14}$$

Eq. (5.15) is the solved matrix revealing the input impedance solution of Eq. (5.16).

$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} . \begin{pmatrix} v_g \\ v_{in} \end{pmatrix} = \begin{pmatrix} 0 \\ -1/((C_{gs} + C_{gd} + C_{gb})s) \end{pmatrix} .i_{in} \tag{5.15}$$

$$v_{in} = -\frac{1}{(C_{gs} + C_{gd} + C_{gb})s} i_{in} \tag{5.16}$$

Fig. 5.7 represents the testbench implemented informations connected to the transistor, for biasing correctly and for declaring the input and output elements. The graphic presents the input impedance AC analysis, validating the computed input impedance transfer function comparing to a simulator results.

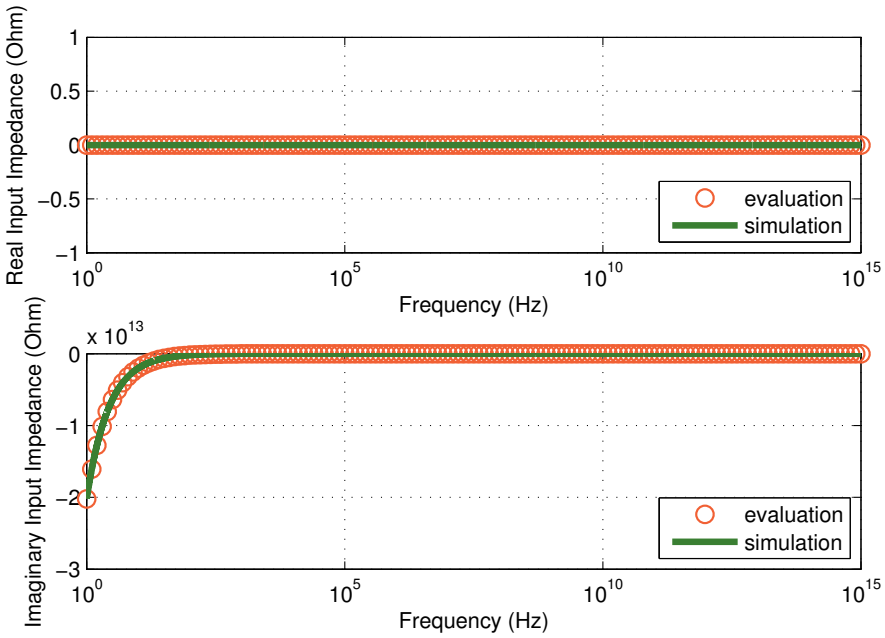

Fig. 5.7: CMOS transistor input impedance simulation to evaluation comparison.

5.4.3 Output Noise

Output noise is analyzed by short-circuiting the input, connecting a noisy current source and declaring the output voltage. Line 20 of Listing 5.1 returns the voltage output noise, building automatically the matrix of admittance with the testbench dedicated to output noise analysis. The resulting small-signal model is Fig. 5.8, in which, an input current is connected between nodes D and S. The testbench information of adding the input completes the admittance matrix (Eq. (5.17)).

$$\begin{pmatrix} g_{11} & g_{12} & g_{13} \\ g_{21} & g_{22} & g_{23} \\ g_{31} & g_{32} & g_{33} \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_s \\ v_g \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix} \cdot \sqrt{\bar{i}_{th}^2 + \bar{i}_f^2} \tag{5.17}$$

Next, the transistor drain-source voltage is declared as the required output for the output noise computation. The testbench information of declaring the potential difference between voltages of nodes D ans S completes the matrix (Eq. (5.18)).

$$\begin{pmatrix} g_{11} & g_{12} & g_{13} & 0 \\ g_{21} & g_{22} & g_{23} & 0 \\ g_{31} & g_{32} & g_{33} & 0 \\ 1 & -1 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_s \\ v_g \\ v_{out} \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 0 \\ 0 \end{pmatrix} \cdot \sqrt{\bar{i}_{th}^2 + \bar{i}_f^2} \tag{5.18}$$

Finally, the line of the node S and the column of the potential v_s are removed from the matrix (Eq. (5.19)). Moreover, because G is short-circuited to S that is grounded, the line of the node G and the column of v_g are also removed from the matrix.

$$\begin{pmatrix} g_{11} & 0 \\ 1 & -1 \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_{out} \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \end{pmatrix} \cdot \sqrt{\bar{i}_{th}^2 + \bar{i}_f^2} \tag{5.19}$$

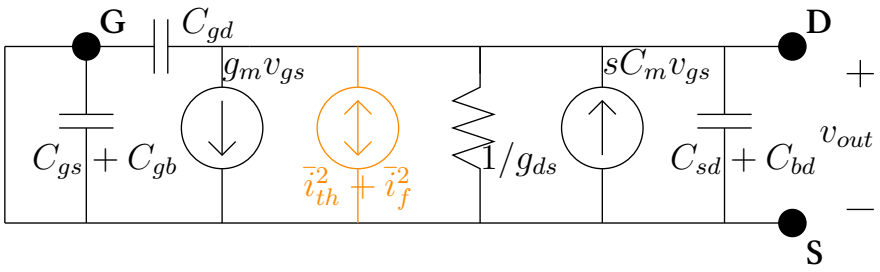

Fig. 5.8: The small-signal model for a transistor output noise evaluation.

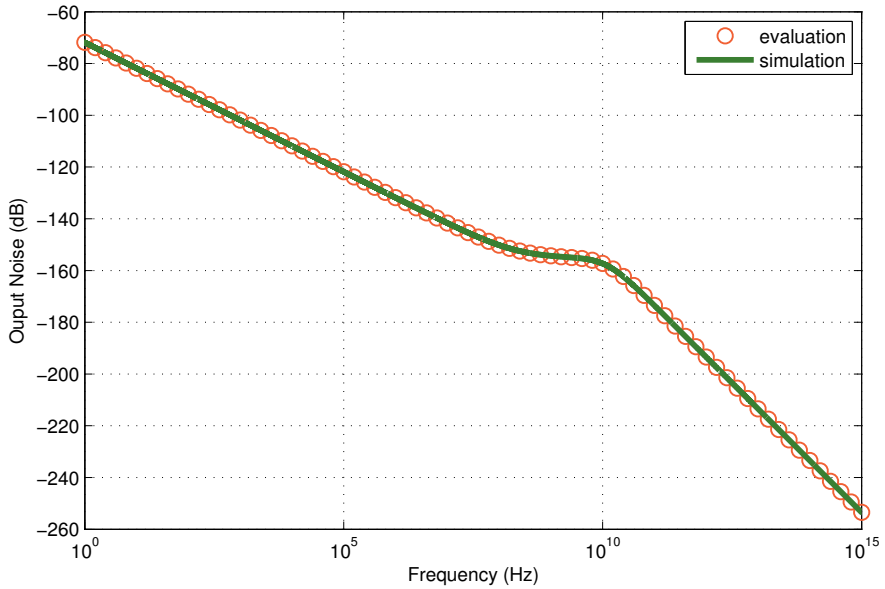

Fig. 5.9: CMOS transistor output noise simulation to evaluation comparison.

Eq. (5.20) is the solved matrix revealing the output noise solution of Eq. (5.21), where the result is squared.

$$\begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_d \\ v_{out} \end{pmatrix} = \begin{pmatrix} 0 \\ -\frac{1}{(C_{sd}+C_{gd}+C_{bd})s+g_{ds}} \end{pmatrix} \cdot \sqrt{\bar{i}_{th}^2 + \bar{i}_f^2} \tag{5.20}$$

$$v_{out}^2 = \frac{1}{((C_{sd}+C_{gd}+C_{bd})s+g_{ds})^2}(\bar{i}_{th}^2 + \bar{i}_f^2) \tag{5.21}$$

Fig. 5.9 represents the testbench implemented informations connected to the transistor, for biasing correctly and for declaring the input and output elements. The graphic presents the output noise AC analysis, validating the computed output noise transfer function comparing to a simulator results.

5.5 Nonlinear Performance Evaluation

We considered only g_m and g_{ds} as nonlinear, we did not manage the nonlinearity of the MOS capacitors and the nonlinearity analyzes has been limited to the 3^{rd} order.

5.5.1 Nonlinearity modeling in analog integrated circuits

A nonlinear element behavior can be represented as: $y = a_0 + a_1 x + a_2 x^2 + a_3 x^3$. When the small-signal parameters g_m and g_{ds} are considered nonlinear, the formula $i_{ds} = g_m v_{gs} + g_{ds} v_{ds}$ becomes

Eq. (5.22).

$$i_{ds} = g_m v_{gs} + g_{ds} v_{ds} + i_{NL} \tag{5.22}$$

After adding the current source i_{NL} in the admittance matrix, we can obtain Eq. (5.23). This can be represented in the small-signal model, Fig. 5.10, as a current source connected in parallel with g_m transconductance.

$$v_{out} = \frac{(C_m + C_{gd})s - g_m}{(C_{sd} + C_{gd} + C_{bd})s + g_{ds}} v_{in} - \frac{i_{NL}}{(C_{sd} + C_{gd} + C_{bd})s + g_{ds}} \tag{5.23}$$

In a first stage we can observe the effect of a nonlinearity described in Eq. (5.24).

$$i_{NL} = K_{2g_m} v_{gs}^2 + K_{3g_m} v_{gs}^3 \tag{5.24}$$

The coefficients can be extracted (Eq. (5.25)) by doing an approximation, because v_{gs} is small enough.

$$\begin{cases} g_m & = \dfrac{\partial i_{ds}}{\partial v_{gs}} \\ K_{2g_m} & = \dfrac{1}{2} \dfrac{\partial^2 i_{ds}}{\partial v_{gs}^2} \\ K_{3g_m} & = \dfrac{1}{6} \dfrac{\partial^3 i_{ds}}{\partial v_{gs}^3} \end{cases} \tag{5.25}$$

Finally, Eq. (5.26) combines Eq. (5.23) and (5.24) and declares that $v_{in} = v_{gs}$.

$$v_{out} = \frac{(C_m + C_{gd})s - g_m}{(C_{sd} + C_{gd} + C_{bd})s + g_{ds}} v_{in} - \frac{K_{2g_m}}{(C_{sd} + C_{gd} + C_{bd})s + g_{ds}} v_{in}^2 - \frac{K_{3g_m}}{(C_{sd} + C_{gd} + C_{bd})s + g_{ds}} v_{in}^3 \tag{5.26}$$

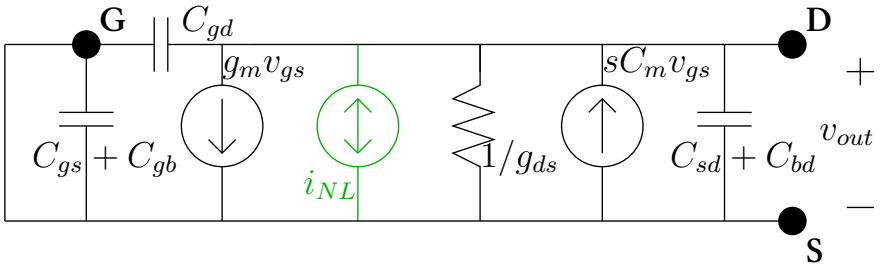

Fig. 5.10: The small-signal model for a transistor nonlinearity evaluation.

By considering also g_{ds} nonlinear, v_{gs} and v_{ds} are both contributors to the current i_{ds}. That's why, Eq. (5.27) contains additional elements $K_{2_{g_m \& g_{ds}}}$, $K_{3_{2g_m \& g_{ds}}}$ and $K_{3_{2g_{ds} \& g_m}}$.

$$\begin{aligned} i_{NL} = {} & K_{2g_m} v_{gs}^2 + K_{2g_{ds}} v_{ds}^2 + K_{2_{g_m \& g_{ds}}} v_{gs} v_{ds} \\ & + K_{3g_m} v_{gs}^3 + K_{3g_{ds}} v_{ds}^3 + K_{3_{2g_m \& g_{ds}}} v_{gs}^2 v_{ds} + K_{3_{2g_{ds} \& g_m}} v_{gs} v_{ds}^2 \end{aligned} \tag{5.27}$$

The coefficients can be extracted (Eq. (5.28) and (5.29)) by doing an approximation, because v_{ds} and v_{gs} are small enough.

$$\begin{cases} g_{ds} & = \dfrac{\partial i_{ds}}{\partial v_{ds}} \\ K_{2g_{ds}} & = \dfrac{1}{2} \dfrac{\partial^2 i_{ds}}{\partial v_{ds}^2} \\ K_{3g_{ds}} & = \dfrac{1}{6} \dfrac{\partial^3 i_{ds}}{\partial v_{ds}^3} \end{cases} \tag{5.28}$$

$$\begin{cases} K_{2_{g_m \& g_{ds}}} & = \dfrac{\partial^2 i_{ds}}{\partial v_{gs} \partial v_{ds}} \\ K_{3_{2g_m \& g_{ds}}} & = \dfrac{1}{2} \dfrac{\partial^3 i_{ds}}{\partial v_{gs}^2 \partial v_{ds}} \\ K_{3_{2g_{ds} \& g_m}} & = \dfrac{1}{2} \dfrac{\partial^3 i_{ds}}{\partial v_{ds}^2 \partial v_{gs}} \end{cases} \tag{5.29}$$

In the following equations we will consider Eq. (5.30) to simplify the aspect of formulas.

$$\begin{cases} H(s) & = TF_{v_{in} \to v_{out}}(s) = \dfrac{(C_m + C_{gd})s - g_m}{(C_{sd} + C_{gd} + C_{bd})s + g_{ds}} \\ & TF_{i_{NL} \to v_{out}}(s) = -\dfrac{1}{(C_{sd} + C_{gd} + C_{bd})s + g_{ds}} \end{cases} \tag{5.30}$$

Eq. (5.31) combines Eq. (5.23) and (5.27) and declares that $v_{in} = v_{gs}$ and $v_{out} = v_{ds}$.

$$\begin{aligned} v_{out} = {} & H(s) v_{in} \\ & + TF_{i_{NL} \to v_{out}}(s)(K_{2g_m} v_{in}^2 + K_{2g_{ds}} v_{out}^2 + K_{2_{g_m \& g_{ds}}} v_{in} v_{out} \\ & + K_{3g_m} v_{in}^3 + K_{3g_{ds}} v_{out}^3 + K_{3_{2g_m \& g_{ds}}} v_{in}^2 v_{out} + K_{3_{2g_{ds} \& g_m}} v_{in} v_{out}^2) \end{aligned} \tag{5.31}$$

At this point, to prevent nonlinear solving by some complicated algorithms with converging values, the equation is approximated by replacing v_{out} using the linear gain Eq. (5.9). Finally, we obtain Eq. (5.32).

$$
\begin{aligned}
v_{out} = {} & H(s)v_{in} \\
& + K_{2g_m} TF_{i_{NL}\rightarrow v_{out}}(s) v_{in}^2 + K_{2g_{ds}} TF_{i_{NL}\rightarrow v_{out}}(s)(H(s)v_{in})^2 \\
& + K_{2_{g_m \& g_{ds}}} TF_{i_{NL}\rightarrow v_{out}}(s)[v_{in}(H(s)v_{in})] \\
& + K_{3g_m} TF_{i_{NL}\rightarrow v_{out}}(s) v_{in}^3 + K_{3g_{ds}} TF_{i_{NL}\rightarrow v_{out}}(s)(H(s)v_{in})^3 \\
& + K_{3_{2g_m \& g_{ds}}} TF_{i_{NL}\rightarrow v_{out}}(s)[v_{in}^2(H(s)v_{in})] \\
& + K_{3_{2g_{ds} \& g_m}} TF_{i_{NL}\rightarrow v_{out}}(s)[v_{in}(H(s)v_{in})^2]
\end{aligned} \tag{5.32}
$$

5.5.2 Volterra kernels

The result in Eq. (5.32) can be implemented in a high-level simulation by giving an accurate nonlinearity behavior to a model. It is important to follow the brackets of the equation because the Laplace transfer function is not associative with time variant functions. Actually, the effect of a Laplace transfer function depends on the signal frequency:

$$[H(s)v_{in}]^2 \neq H(s)^2 v_{in}^2$$

To prevent this kind of ambiguity and also to factorize correctly v_{in}^2 and v_{in}^3, we can use a Volterra representation [Volterra59] [Crombez07]. For nonlinearity tests, the input is commonly considered as sine wave. This way, the frequency $s = j\omega$ can be anticipated on each involved transfer function. In a single tone input, the output signal can be represented as in Eq. (5.33).

$$v_{out} = H_1(s_1)v_{in} + H_2(s_1, s_1)v_{in}^2 + H_3(s_1, s_1, s_1)v_{in}^3 \tag{5.33}$$

Eq. (5.34) and (5.35) are small examples revealing the kind of transformation that will be applied.

$$v_2 = [H(s)v_1]^2 \Rightarrow H_2(s_1, s_1) = H(s_1)^2 \tag{5.34}$$

$$v_2 = H(s)v_1^2 \Rightarrow H_2(s_1, s_1) = H(2s_1) \tag{5.35}$$

Eq. (5.36) is result of combining Eq. (5.31) and (5.33), $H_1(s_1)$, $H_2(s_1, s_1)$ and $H_3(s_1, s_1, s_1)$ are called Volterra kernels. Some new elements provided by second order appeared in the third order because v_{out} is not approximated using the linear gain like in the result Eq. (5.32).

$$\begin{cases} H_1(s_1) & = H(s_1) \\ H_2(s_1,s_1) & = TF_{i_{NL}\to v_{out}}(s_1+s_1)[K_{2g_m}+K_{2g_{ds}}H_1(s_1)^2+K_{2_{g_m\&g_{ds}}}H_1(s_1)] \\ H_3(s_1,s_1,s_1) & = TF_{i_{NL}\to v_{out}}(s_1+s_1+s_1)[K_{3g_m}+K_{3g_{ds}}H_1(s_1)^3 \\ & \quad + K_{3_{2g_m\&g_{ds}}}H_1(s_1)+K_{3_{2g_{ds}\&g_m}}H_1(s_1)^2 \\ & \quad + K_{2_{g_m\&g_{ds}}}H_2(s_1,s_1)+2K_{2g_{ds}}H_1(s_1)H_2(s_1,s_1)] \end{cases} \tag{5.36}$$

5.5.3 Direct Performance Calculation

When the input signal is determined, it is also possible to predict the harmonics level of output (HD_2, HD_3), allowing us to determine the nonlinearity performance like IIP_2, IIP_3 without any transient simulation.

$$\begin{cases} v_{in} & = V_{in}cos(\omega t) \\ v_{in}^2 & = \dfrac{V_{in}^2}{2}+\dfrac{cos(2\omega t)}{2}V_{in}^2 \\ v_{in}^3 & = \dfrac{3}{4}cos(\omega t)V_{in}^3+\dfrac{cos(3\omega t)}{4}V_{in}^3 \end{cases} \tag{5.37}$$

Eq. (5.38) declares v_{out} as a sum of harmonics.

$$v_{out} = V_{out,1,0}cos(\omega t)+V_{out,2,0}cos(2\omega t)+V_{out,3,0}cos(3\omega t) \tag{5.38}$$

With Eq. (5.31), (5.37) and (5.38), we can compute Eq. (5.39), where $V_{out,n,0}$ is the n-th order harmonic value of output voltage. The result, for each n-th order harmonic, has been simplified by keeping only the expressions that are multiplied by the lower order input: V_{in}^n.

$$\begin{cases} V_{out,1,0} & = H(s_1)V_{in}+\dots \\ V_{out,2,0} & = TF_{i_{NL}\to v_{out}}(s_1+s_1)V_{in}^2[K_{2g_m}+K_{2g_{ds}}V_{out,1,0}^2+K_{2_{g_m\&g_{ds}}}V_{out,1,0}]/2+\dots \\ V_{out,3,0} & = TF_{i_{NL}\to v_{out}}(s_1+s_1+s_1)V_{in}^3[K_{3g_m}+K_{3g_{ds}}V_{out,1,0}^3 \\ & \quad + K_{3_{2g_m\&g_{ds}}}V_{out,1,0}+K_{2g_{ds}\&g_m}V_{out,1,0}^2 \\ & \quad + 2K_{2_{g_m\&g_{ds}}}V_{out,2,0}+4K_{2g_{ds}}V_{out,1,0}V_{out,2,0}]/4+\dots \end{cases} \tag{5.39}$$

Moreover, we can extend this result by considering a multi-tone input presented in Eq. (5.40), this allows a direct computation of intermodulation nonlinearity. When considering a two-tone test input signal:

$$\begin{cases} v_{in} & = V_{in}[cos(\omega_1 t) + cos(\omega_2 t)] \\ v_{in}^2 & = V_{in}^2[cos((\omega_1+\omega_2)t) + cos((\omega_2-\omega_1)t) + cos(2\omega_2 t)/2 + cos(2\omega_1 t)/2 + 1] \\ v_{in}^3 & = V_{in}^3[3cos((\omega_1+2\omega_2)t) + 3cos((2\omega_2-\omega_1)t) + 3cos((2\omega_1+\omega_2)t) + 3cos((\omega_2-2\omega_1)t) \\ & \quad + cos(3\omega_2 t) + 9cos(\omega_2 t) + cos(3\omega_1 t) + 9cos(\omega_1 t)]/4 \end{cases} \tag{5.40}$$

This way, the representation of v_{out} is extended to Eq. (5.41)

$$\begin{aligned} v_{out} = {} & V_{out,1,0}cos(\omega_1 t) + V_{out,0,1}cos(\omega_2 t) + V_{out,2,0}cos(2\omega_1 t) \\ & + V_{out,-1,1}cos((\omega_2-\omega_1)t)) + V_{out,3,0}cos(3\omega_1 t) + V_{out,2,-1}cos((2\omega_1-\omega_2)t) \end{aligned} \tag{5.41}$$

With Eq. (5.31), (5.40) and (5.41), we can compute Eq. (5.42), where $V_{out,m,n}$ is the $m-n$-th order intermodulation product at frequency $\omega_1 - \omega_2$.

$$\begin{cases} V_{out,1,0} & = H(s_1)V_{in} + \dots \\ V_{out,0,-1} & = H(-s_2)V_{in} + \dots \\ V_{out,2,0} & = TF_{i_{NL}\to v_{out}}(2s_1)V_{in}^2[K_{2g_m} + V_{out,1,0}^2 K_{2g_{ds}} + V_{out,1,0}K_{2_{g_m \& g_{ds}}}]/2 + \dots \\ V_{out,1,-1} & = TF_{i_{NL}\to v_{out}}(s_1-s_2)V_{in}^2[V_{out,0,-1}K_{2_{g_m \& g_{ds}}} + 2V_{out,0,-1}V_{out,1,0}K_{2g_{ds}}]/2 + \dots \\ V_{out,2,-1} & = TF_{i_{NL}\to v_{out}}(2s_1-s_2)V_{in}^3[3V_{out,0,-1}V_{out,1,0}^2 K_{3g_{ds}} + V_{out,0,-1}K_{3_{2g_m \& g_{ds}}} \\ & \quad + 2V_{out,0,-1}V_{out,1,0}K_{3_{2g_{ds} \& g_m}} + 2V_{out,1,-1}K_{2_{g_m \& g_{ds}}} \\ & \quad + (4V_{out,0,-1}V_{out,2,0} + 4V_{out,1,0}V_{out,1,-1})K_{2g_{ds}}]/4 + \dots \end{cases} \tag{5.42}$$

Finally, a last extension of the model is to allow to consider the input signal also as a nonlinear function (Eq. (5.43)). This way, the transistor nonlinearity behavior can be connected to a global circuit nonlinearity behavior.

$$\begin{aligned} v_{in} = {} & V_{in,1,0}cos(\omega_1 t) + V_{in,0,1}cos(\omega_2 t) + V_{in,2,0}cos(2\omega_1 t) \\ & + V_{in,-1,1}cos((\omega_2-\omega_1)t)) + V_{in,3,0}cos(3\omega_1 t) + V_{in,2,-1}cos((2\omega_1-\omega_2)t) \end{aligned} \tag{5.43}$$

Output voltage nonlinearities are presented in Eq. (5.44), the expressions use the input voltage intermodulation products amount. It is determined by the eventual circuit connected to the input of the analyzed transistor.

$$
\begin{cases}
V_{out,1,0} & = H(s_1)V_{in} + \dots \\
V_{out,0,-1} & = H(-s_2)V_{in} + \dots \\
V_{out,2,0} & = TF_{i_{NL}\to v_{out}}(2s_1)[V_{out,1,0}V_{in,1,0}K_{2_{g_m\&g_{ds}}} + V_{in,1,0}^2K_{2g_m} + V_{out,1,0}^2K_{2g_{ds}}]/2 + \dots \\
V_{out,1,-1} & = TF_{i_{NL}\to v_{out}}(s_1 - s_2)[(V_{out,0,-1}V_{in,1,0} + V_{out,1,0}V_{in,0,-1})K_{2_{g_m\&g_{ds}}} \\
& \quad + 2V_{in,0,-1}V_{in,1,0}K_{2g_m} + 2V_{out,0,-1}V_{out,1,0}K_{2g_{ds}}]/2 + \dots \\
V_{out,2,-1} & = TF_{i_{NL}\to v_{out}}(2s_1 - s_2)[3V_{in,0,-1}V_{in,1,0}^2K_{3g_m} + 3V_{out,0,-1}V_{out,1,0}^2K_{3g_{ds}} \\
& \quad + (V_{out,0,-1}V_{in,1,0}^2 + 2V_{out,1,0}V_{in,0,-1}V_{in,1,0})K_{3_{2g_m\&g_{ds}}} \\
& \quad + (2V_{out,0,-1}V_{out,1,0}V_{in,1,0} + V_{out,1,0}^2V_{in,0,-1})K_{3_{2g_{ds}g_m}} \\
& \quad + (2V_{out,0,-1}V_{in,2,0} + 2V_{out,1,0}V_{in,1,-1} + 2V_{out,1,-1}V_{in,1,0} + 2V_{out,2,0}V_{in,0,-1})K_{2_{g_m\&g_{ds}}} \\
& \quad + (4V_{in,0,-1}V_{in,2,0} + 4V_{in,1,0}V_{in,1,-1})K_{2g_m} \\
& \quad + (4V_{out,0,-1}V_{out,2,0} + 4V_{out,1,0}V_{out,1,-1})K_{2g_{ds}}]/4 + \dots
\end{cases}
\tag{5.44}
$$

5.5.4 Systematic Nonlinear Performance Evaluation

In the point of view of implementation, the equations has been generated by the Maxima library that was developed and presented in Appendix B. An important scheduling for intermodulation products computation is revealed by the expressions. In fact, N-th order expressions are depen-

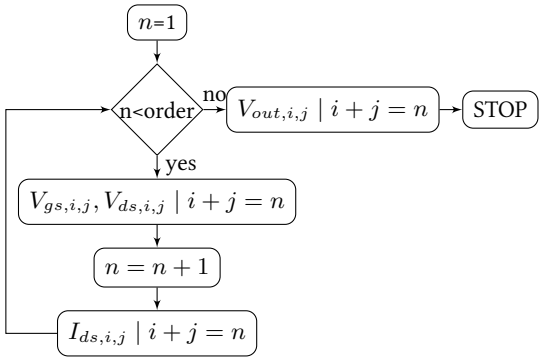

Fig. 5.11: Flow chart for the direct computation of n-th order intermodulation products of a circuit with x nonlinear elements.

dent from $(N-1)$-th to 1^{st} order expressions. The flow chart presented in Fig. 5.11, explains this scheduling.

The values $V_{gs_x,r,s}$, $V_{ds_x,r,s}$ and $V_{out_x,r,s}$ are computed by solving the admittance matrix, this final stage of computing the amount of nonlinearity is implemented in Listing 5.2. In this matrix, for 1^{st} order, the input is V_{in} whereas the input is the list of all nonlinear current sources $i_{ds_x,r,s}$ for other orders.

Fig. 5.12 presents the testbench for nonlinearity analyzes and the plot of 1^{st}, 2^{nd} and 3^{rd} order output voltage intermodulation products following the linear input variation. We can confirm the precision of the implementation in the range of linear variation. Over this linear variation, the input becomes high and the neglected expressions, coming from higher order modulation, are becoming dominant. Moreover, a noise floor appears at -300 dB in the simulation result due to the simulation precision setings.

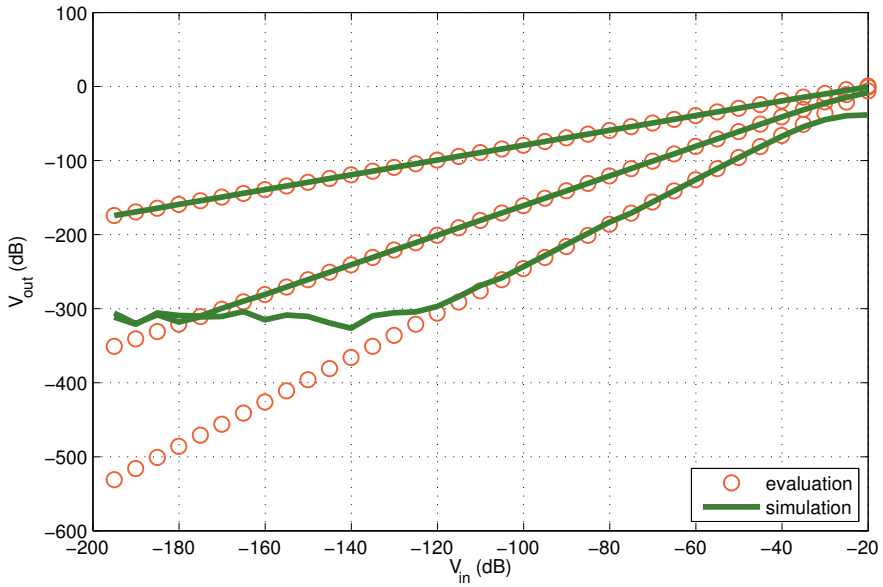

Fig. 5.12: 1^{st}, 2^{nd} and 3^{rd} order voltage output intermodulation products of a CMOS transistor simulation to evaluation comparison.

Listing 5.2: Maxima source code for generating the intermodulation products of a MOS transistor output voltage

```
batchload("AC_analysis.wxm")$
batchload("nonlinearity.wxm")$
A:init()$
/* Connect */
D:OUT$
G:IN$
S:SS$
/* "Transistor 1" instance */
A:addTransistor(A,D,G,S,S,1)$
/* 1st order evaluation (linear)*/
_Vout10:subst(s1,s,getVGain(A,OUT,SS,IN,SS))$

_Vgs:getVGain(A,G,S,IN,SS)$
_Vgs10:subst(s1,s,_Vgs)$
_Vgs01:subst(-s2,s,_Vgs)$

_Vds:getVGain(A,D,S,IN,SS)$
_Vds10:subst(s1,s,_Vds)$
_Vds01:subst(-s2,s,_Vds)$
/* Short Circuit the input for nonlinear evaluation (here orders 2 and 3)*/
A:shortCircuit(A,SS,IN)$
IN:SS$
/* Injecting the nonlinear current in the transistor model*/
A:setIin(A,D,S,Igm)$
/* 2nd order Vout at w1-w2 (for IIP2 evaluation)*/
_Vout:getVout(A,OUT,SS)$
_Vout11:subst(_Igm11,Igm,subst(s1-s2,s,_Vout))$

_Vgs:getVout(A,G,S)$
_Vgs11:subst(_Igm11,Igm,subst(s1-s2,s,_Vgs))$
_Vgs20:subst(_Igm20,Igm,subst(2*s1,s,_Vgs))$

_Vds:getVout(A,D,S)$
_Vds11:subst(_Igm11,Igm,subst(s1-s2,s,_Vds))$
_Vds20:subst(_Igm20,Igm,subst(2*s1,s,_Vds))$

/* 3rd order Vout at 2*w1-w2 (for IIP3 evaluation)*/
_Vout21:subst(_Igm21,Igm,subst(2*s1-s2,s,_Vout))$
```

5.6 Conclusion

A systematic method to extract the performance from circuit-level description has been presented. It permits the designer to have access to the performance of a circuit through a symbolic performance matrix without calling any circuit simulator. Both linear and nonlinear performance evaluation have been described in this methodology. The proposed methodology permits easy introduction of new small-signal models. It has been applied to the developed low frequency and RF BSIM3v3 models. Linear performance evaluation tool achieves a precision identical to circuit simulators. Nonlinear performance evaluation tool achieves a precision accurate up to -300 dB with respect to circuit simulators.

6

Systematic Circuit-Level Design and Optimization of Analog and RF Circuits

6.1 Introduction

In this chapter, we propose a systematic analog and RF circuit design flow based on the performance evaluation tools presented in Chapter 5. The chapter is structured as follows. In Section 6.2, we present the circuit-level design methodology. Sections 6.3 and 6.4 present the case studies related to Wireless Sensor Network nodes design. In these sections, we present our proposed circuit-level design flow for a GmC integrator in a continuous-time $\Sigma\Delta$ ADC and for a Low Noise Amplifier in a ZigBee transceiver.

6.2 Proposed Circuit-Level Design Flow

As depicted in Fig. 6.1, the design methodology consists of three main procedures: transistor and passive elements biasing and sizing, performance evaluation and optimization procedure. Those procedures are executed sequentially in a unified C++ implementation.

6.2.1 Transistor and Passive Elements Biasing/Sizing

Depending on the circuit specifications: supply voltage, noise budget, linearity, input/output common-mode voltage, we determine the DC biasing of each transistor in the circuit.

The transistor sizing procedure is run when the DC biasing is finished. This way, all the sizing transistor input parameters are known: the biasing voltages V_D, V_G, V_S, V_B, the current I_{DS}, the length L_M and the number of fingers $nfinger$. A sizing tool is used to make a technology matched

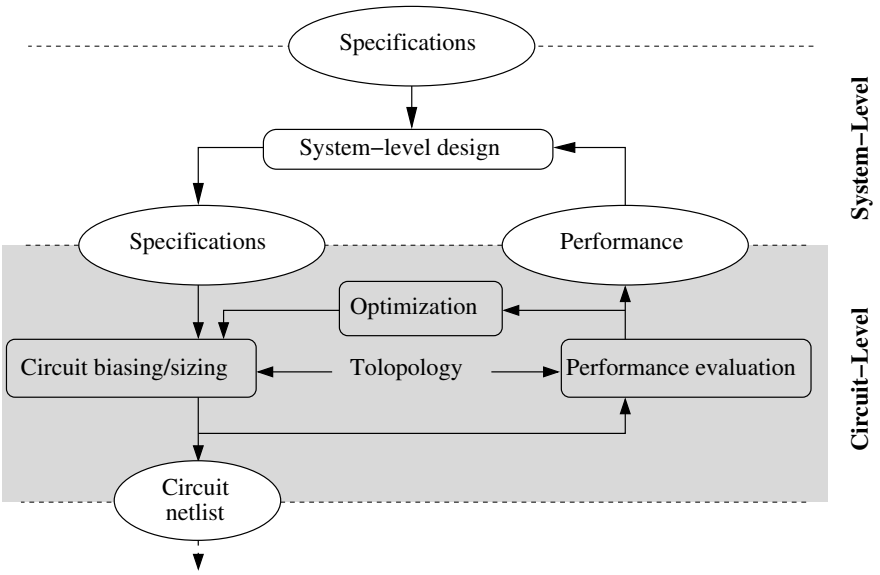

Fig. 6.1: Unified multi-level design environment for mixed signal systems.

sizing with the biasing parameters. The tool has been extended to support inductors, RF transistor additional small-signal parameters and nonlinearity parameters extraction. The libraries have been upgraded by making available the RF dedicated MOS small-signal parameters: r_d, r_g, r_s, r_b, r_{sti}, C_{dsmet}, C_{bsmet}, C_{bsmet}, the thermal noise of r_d, r_g, r_s, r_{sti}, the integrated inductor small-signal parameters: l_{s1}, l_{s2}, r_{pat1}, r_{pat2}, r_{s1}, r_{s2}, r_{patm}, r_{mp}, c_{ox1}, c_{ox2}, c_{oxm}, and the nonlinearity parameters: K_{2g_m}, K_{3g_m}, $K_{2g_{ds}}$, $K_{3g_{ds}}$, $K_{2_{g_m \& g_{ds}}}$, $K_{3_{2g_m \& g_{ds}}}$, $K_{3_{2g_{ds} \& g_m}}$.

CAIRO+ [Iskander07] has been used, as the sizing tool of the methodology, in the first case study (Section 6.3) and CHAMS [Javid09] has been used for the second case study (Section 6.4). As they are both C++ libraries, the homogeneity of the flow has been maintained.

At the end of this procedure, the small-signal, noise (thermal and flicker) and nonlinearity parameters are available, allowing the performance evaluation.

6.2.2 Performance Evaluation

In [Iskander07] [Belfort09], the circuit performance is estimated by approximate equations and it is limited to linear performance. In this work, we propose to use the nonlinear and exact linear performance evaluation techniques presented in Chapter 5 The following procedure is implemented in an equation-based methodology. It takes advantage of the performance evaluation implementation

presented in Chapter 5. It solves the generated performance matrices in a C++ library. As each required performance is solved in C++, the procedure does not need to call an external simulator and can be called by a C++ function, maintaining, in this procedure also, the homogeneity of the flow.

GiNaC is the C++ library [gin] that has been used for symbolic expression manipulation, it is licenced under GNU General Public License (GPL). Especially, it is implemented with the CLN C++ library [cln] for arbitrary sized integers and rationals, and for arbitrary precision floating points. This is interesting for the precision of numerical results after solving performance matrices. It has been used for its linear system solving ability based on matrix gaussian elimination. As it is a C++ library, we could integrate the matrix solving process into the design flow.

The designer has the freedom of getting the gain, impedances, trans-admittances, noise, nonlinearity. Gain can be a voltage, current or power gain. Noise is expressed in term of input, output, squared noise, noise figure. Nonlinearity can be harmonic distortion of 2^{nd} or 3^{rd} order, IIP_2, IIP_3. The designer also controls the performance unit: linear, dB, dBm.

Eq. (6.1) presents the different stages of circuit design that have been implemented in the both biasing/sizing and performance evaluation procedures. It resumes the previously presented both procedures. The colorized elements represent the contribution to the circuit design.

$$\overbrace{\left\{\begin{array}{c} Temp \\ V_{IN} \\ V_{OUT} \\ V_{EFF} \\ V_{DD} \\ . \end{array}\right.}^{\text{Design circuit parameters}} \Rightarrow \overbrace{\left\{\begin{array}{c} W \\ V_{GS} \\ V_{DS} \\ I_{DS} \\ L \\ . \end{array}\right.}^{\text{Transistor biasing and sizing}} \Rightarrow \overbrace{\left\{\begin{array}{c} g_m \\ g_{ds} \\ C_{gd} \\ \bar{i}^2_{th} \\ \bar{i}^2_{f} \\ K_{3_{2g_m \& g_{ds}}} \end{array}\right.}^{\text{Transistor small-signal parameters and nonlinearity parameters}} \Rightarrow \overbrace{\left\{\begin{array}{c} Gain \\ Noise \\ Impedance \\ Distortion \\ . \\ . \end{array}\right.}^{\text{Circuit performance}} . \quad (6.1)$$

6.2.3 Optimization Procedure

The procedure is dedicated for optimizing the design and/or for reaching the necessary performance. It is a loop-based set of algorithms that analyze the performance evolution in relation to some input parameters. The input parameters can be circuit-level or transistor-level parameters. The procedure can implement three kind of algorithms:

- A convergence loop to achieve a chosen value of the performance, for example, achieving a 50Ω input impedance.
- A maximization/minimization loop, for example maximizing the gain.
- A scanning loop to browse and visualize the performance in a range of multiple input parameters, this has been used to visualize, graphically, the variation of a performance.

The implemented optimization procedures are detailed in the design examples (Sections 6.3 and 6.4). They are impedance matching, noise reduction, nonlinearity reduction, gain poles rejection.

Moreover, process migration has been interesting to analyze for technology exploration and time-to-market evaluation.

6.3 Case Study I: GmC Integrator

In this section, we illustrate the proposed systematic circuit-level design flow through the design of a GmC integrator presented in Fig. 6.2. First, we present the GmC integrator design in the context of $\Sigma\Delta$ modulators, the circuit-level systematic design procedures are detailed following Fig. 6.3. Fi-

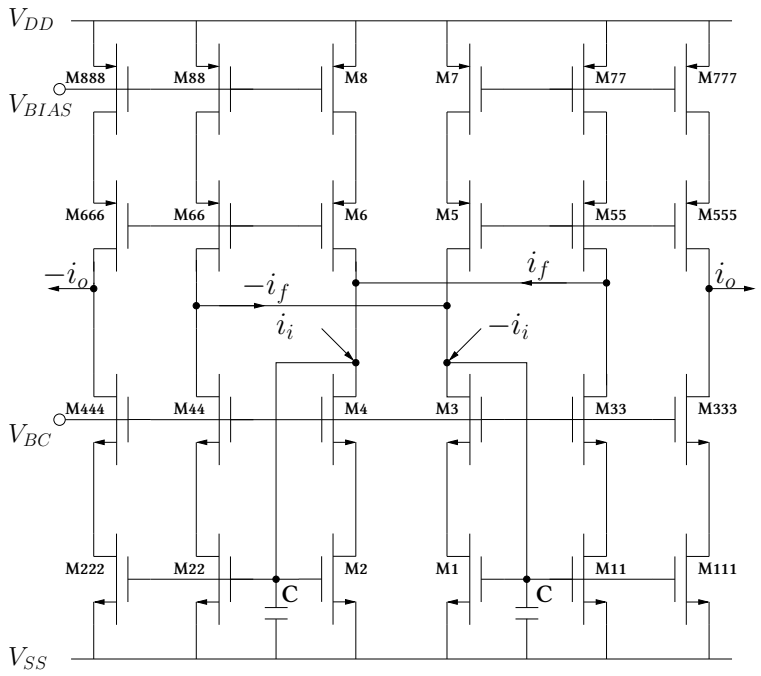

Fig. 6.2: Differential current-mode GmC integrator.

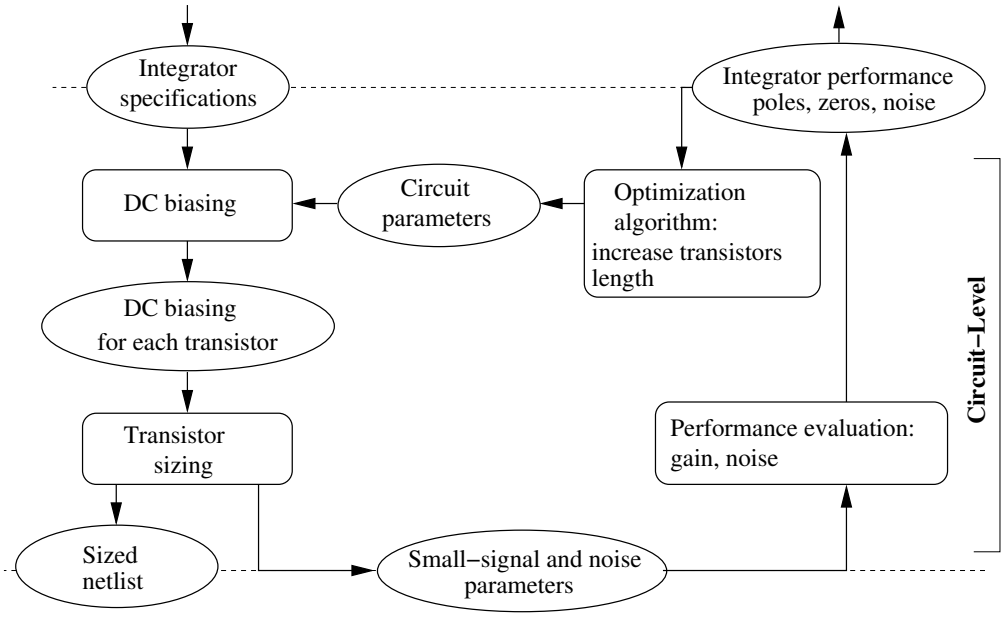

Fig. 6.3: The optimized circuit design flow dedicated to GmC integrators.

nally, design examples are presented revealing the usefulness of the method for design optimization, technology exploration and specification feasibility.

6.3.1 DC Biasing

The biasing procedure is solved by getting the user defined parameters: $V_{EFF3} = V_{GS3} - V_{TH3}$, $V_{EFF5} = V_{GS5} - V_{TH5}$, V_{IN}, V_{BIAS} and computing the voltage of all the nodes. The biasing current I_0 is actually computed in relation to the expected Signal-to-Noise Ratio (SNR) of the $\Sigma\Delta$ modulator defined by Eq. (6.2)

$$SNR_{TH} = \frac{\text{Signal Power}}{\text{Noise Power}} = \frac{\frac{1}{2}A_{\Sigma\Delta}^2 m^2 I_0^2}{\frac{2}{3}g_m 4KTN_{tr}BW} \tag{6.2}$$

Where $A_{\Sigma\Delta}$ is the amplitude of the $\Sigma\Delta$ input signal, I_0 is the biasing current, $m = I_{IN}/I_0$ is the modulation index, g_m is the transconductance of the transistors (approximating that there have all the same value), N_{tr} is the number of transistors, K is Boltzmann constant, T is temperature in Kelvin, and BW is bandwidth of the $\Sigma\Delta$. The transconductance can be expressed by Eq. (6.3).

$$g_m = \frac{2I_0}{V_{EG}} \tag{6.3}$$

By substitution from Eq. (6.3) into Eq. (6.2), we got Eq. (6.4).

$$I_0 = \frac{8}{3}\frac{SNR_{th}4KTN_{tr}BW}{A_{\Sigma\Delta}^2 m^2 V_{EG}} \tag{6.4}$$

This way, we could estimate the biasing current needed to achieve the expected SNR.

6.3.2 Transistor and Passive Elements Sizing

The length of transistors has been chosen to be the variable parameter, the sizing procedure will generate a design for each required values of this parameter. The reason of using the transistor length is that we located this parameter as having an important influence to the position of poles and zeros and the level of noise.

There is one passive element C that is the output capacitance. It mainly controls the first pole position of the integrator. The ideal transfer function can be described by Eq. (6.5), where A_{int} is integrator desired gain and f_s is $\Sigma\Delta$ sampling frequency.

$$\begin{cases} H(s) = \dfrac{A_{int}f_s}{s} \\ H(s) = \dfrac{g_m}{sC} \end{cases} \Rightarrow A_{int}f_s = \frac{g_m}{C} \tag{6.5}$$

By combining Eq. (6.3) and Eq. (6.5) we produce Eq. (6.6).

$$f_s = \frac{2I_0}{V_{EG}A_{int}C} \tag{6.6}$$

In a $\Sigma\Delta$, the bandwidth is expressed by Eq. (6.7).

$$BW = \frac{f_s}{2OSR} \tag{6.7}$$

Combining Eq. (6.6) and Eq. (6.7) produces Eq. (6.6).

$$BW = \frac{I_0}{V_{EG}CA_{int}OSR} \tag{6.8}$$

Finally, the passive capacitance value is computed by Eq. (6.9).

$$C = \frac{8}{3}\frac{SNR_{th}}{A_{\Sigma\Delta}^2 OSRA_{int}}\frac{4KTN_{tr}}{m^2 V_{EG}} \tag{6.9}$$

6.3.3 Performance Evaluation

When designing an integrator for a $\Sigma\Delta$ modulator, the ideal transfer function has to satisfy Eq. (6.5). To check the correctness of the design, the transition frequency (f_T) is computed:

$$\frac{A_{int} f_s}{s} = 1 \Rightarrow f_T = \frac{A_{int} f_s}{2\pi} \tag{6.10}$$

Some publications [Aboushady01b] [Smith96] [Zele96] presented a small-signal model built from a simplified GmC integrator. This model has such transfer function:

$$\begin{cases} H(s) = A_0 \dfrac{(1 - \frac{s}{z_1})}{(1 + \frac{s}{p_1})} \\ z_1 = \dfrac{(g_m - g_{ds})}{2C_{gd}}, p_1 = \dfrac{2g_{ds}}{(C + 4C_{gd})} \text{ and } A_0 = \dfrac{g_m - g_{ds}}{2g_{ds}} \end{cases} \tag{6.11}$$

This model is very simplified. In this work we use the "exact" transfer function.

We chose to upgrade the transfer function (Eq. (6.11)) by taking into account cascoding transistors and including every parasitic capacitance of the standard BSIM3v3 complete small-signal transistor model, presented in Fig. 6.4, in order to avoid inaccuracy when another technology is selected or when design specifications or parameters, like frequency, are changed. This could be done thanks to the performance evaluation automated procedure, that exploits a automatically generated precise transfer function.

As the matrix solver can process symbolic parameters, we have chosen to keep the parameter $s = j\omega$ to be symbolic. This has been done to extract precisely two poles and two zeros.

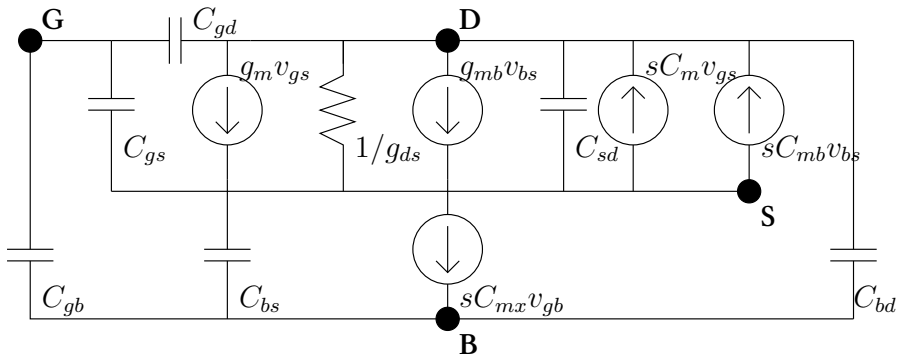

Fig. 6.4: CMOS transistor BSIM3v3 small-signal model.

Table 6.1: Specifications of the 2^{nd} order $\Sigma\Delta$ modulator.

SNR	$>$ 60 dB
OSR	64
BW (MHz)	10
A_{int}	0.33
$A_{\Sigma\Delta}$	0.56

6.3.4 Optimization Procedure

The implemented algorithm is finding an optimal value for transistor lengths, by resolving the constraint of achieving a theoretical SNR linked to the noise performance, and verifying the position of the 2^{nd} pole trying to reject it as far as possible from the f_T frequency. In fact, the SNR performance of a $\Sigma\Delta$ modulator is related to the level of noise but also to a good shaping, achieved when the 2^{nd} pole is sufficiently rejected.

6.3.5 Design Examples

Table 6.1 presents the top-level specifications that have been chosen as constraints and the GmC integrator specifications that have been obtained by system-level simulations. Fig. 6.5 plots the AC analysis of the GmC gain. The ideal model in the figure is from the transfer function presented in Eq. (6.5). The 1 pole/1 zero curve plots the transfer function of the simplified model given by the approximative Eq. (6.11). The 2 poles/2 zeros curve plots the transfer function from the Modified Nodal Analysis presented in Chapter 5. We can observe that it matches to the transistor-level simulation, the small differences come from neglecting the effect of the PMOS current sources. A deeper observation reveals the need of this new model, in fact with the 1 pole/1 zero, the deterioration due to the second pole cannot be noticed.

Table 6.2 shows different designs and justifies the method used to optimize performance. We present each transistor sizing of GmC integrator, the NMOS transistors (M1), its cascode transistors (M3), the PMOS transistors (M7) and the PMOS cascode transistors (M5). The table also compares the f_T to 2^{nd} pole frequency: f_{p_2}, if the second pole is too close, the design is not valid, as it affects

the noise shaping of the $\Sigma\Delta$. We will see, in Chapter 7, that the position of f_{p_2} has a strong influence on the $\Sigma\Delta$ maximum achievable Signal-to-Noise Ratio. We computed the SNR_{cir}, by getting the noise level from the performance evaluation procedure, this value is of course more precise than the result of Eq. (6.2) that was just used to choose an initial value of I_0.

$$SNR_{cir} = \frac{\text{Signal Power}}{\text{Thermal Noise Power} + \text{Flicker Noise Power}} \tag{6.12}$$

The first column illustrates the link between cascode transistors (M3,M5) length and the position of the 2^{nd} pole. We chose L3=1.30 μm and L5=0.60 μm and we noticed that, after the sizing procedure, the 2^{nd} pole reached 480.4 MHz. When cascode transistors (M3,M5) length is too high, the 2^{nd} pole becomes too close to the f_T frequency. It is admitted that $f_{p_2} > 10f_T$ and a simulation can confirm the noise shaping deterioration. We will see the $\Sigma\Delta$ noise shaping deterioration when the back-annotated $\Sigma\Delta$ modulator will be simulated at system level in next chapter.

The second column shows the relation between transistors length and noise. We chose minimal transistors length and noticed that the theoretical SNR_{cir}=41.66 dB is very low and can not comply the specifications ($SNR_{cir} > 60$dB). In fact, the noise is related to transistor's area, it is lower when transistor's area is increased. By analyzing the contribution of each transistor, we concluded that cascode transistors have less influence to noise.

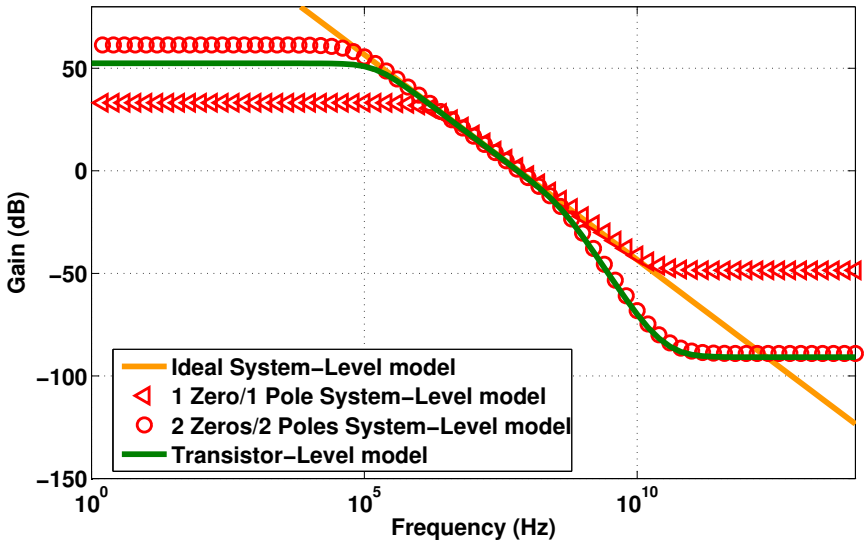

Fig. 6.5: Frequency response of a $\Sigma\Delta$ GmC integrator designed for BW=10 MHz, comparing back-annotated Zeros/Poles models, SPICE model and ideal model (0.13μm process).

Table 6.3 presents the results for technology migration, comparing 0.13 μm to 0.25 μm. We can observe that the current is much lower for 0.25 μm process. Whereas the 2^{nd} pole is going closer to f_T for 0.25 μm process. For BW=10 MHz, the position of the pole is sufficient but in higher bandwidths there could be a limitation for the 0.25 μm process that will make valid designs impossible. Also the integrated capacitance C of GmC integrator is related to desired bandwidth, for the same specifications the integrated capacitance is much lower in 0.25 μm process. We can also predict that for higher bandwidth specifications, the f_T frequency can not be achieved. In this case, the parasitic capacitances would be higher than the desired C in the transfer function: $\frac{g_m}{sC}$. This technology migration analysis allows us to conclude that the choice of CMOS process is very related to the specifications and an automatic tool is very useful for this exploration.

Table 6.2: Integrators circuit characteristics, comparing three different transistors length settings (0.13μm CMOS process) with Vdd=1.2V.

	Cascode transistors length too high	Transistors length too low
W1/L1 (μm/μm)	122.2/1.18	12.8/0.13
W3/L3 (μm/μm)	894.3/1.30	30.0/0.13
W5/L5 (μm/μm)	1463.0/0.60	92.0/0.13
W7/L7 (μm/μm)	112.3/0.23	26.8/0.13
I_0 (mA)	1.3	7.4
Desired f_T (MHz)	67.9	67.9
2^{nd} pole (MHz)	480.4	5575.0
C (pF)	13.0	13.5
SNR_{cir}	61.66 dB	41.66 dB

Table 6.3: Integrators circuit characteristics, analyzing the process migration with Vdd=1.2V.

CMOS Process	0.13 μm	0.25μm
W1/L1 (μm/μm)	181.1/0.73	21.7/0.85
W3/L3 (μm/μm)	113.6/0.13	14.9/0.25
W5/L5 (μm/μm)	337.0/0.13	48.1/0.25
W7/L7 (μm/μm)	240.9/0.23	92.2/1.05
I_0 (mA)	2.7	0.3
Desired f_T (MHz)	67.9	67.9
2^{nd} pole (MHz)	3255.7	788.924
C (pF)	38.9	3.4
SNR_{cir}	61.77 dB	61.64 dB

6.4 Case Study II: Low Noise Amplifier

In this section, we illustrate the proposed systematic circuit-level design flow through the design of a Low Noise Amplifier presented in Fig. 6.6. First, we present the LNA design in the context of RF receivers, the circuit-level systematic design procedures are detailed following Fig. 6.7. Secondly, the technique is applied to several design examples with two different topologies.

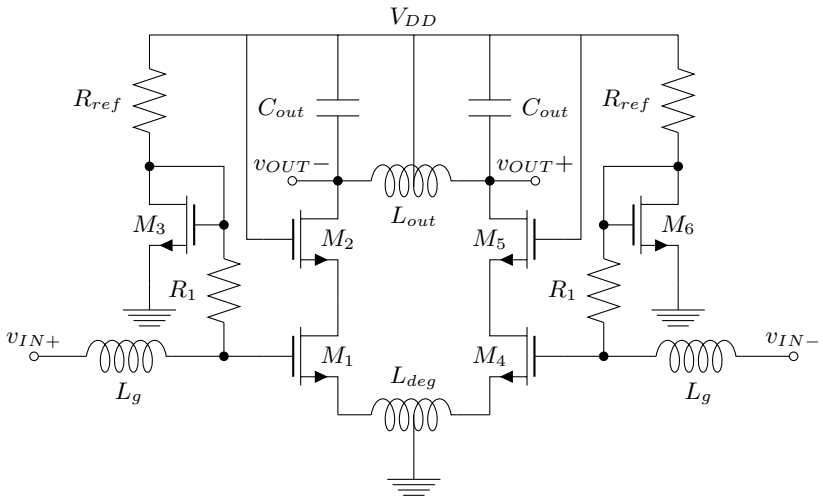

Fig. 6.6: The targeted circuit topology: cascode LNA with inductive source degeneration.

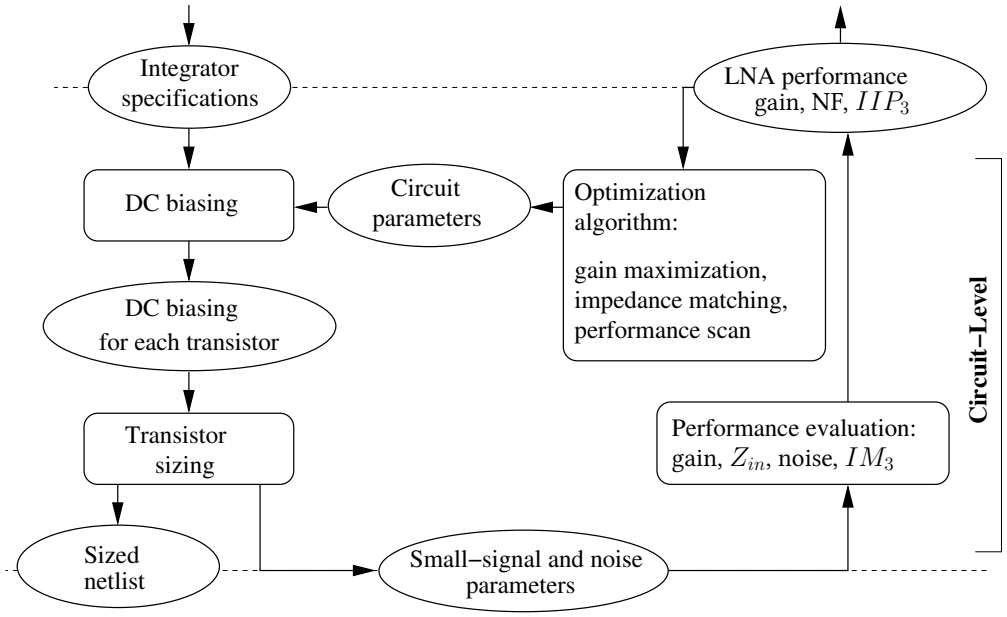

Fig. 6.7: The optimized circuit design flow dedicated to LNA.

The cascode LNA with inductive source degeneration topology shown in Fig. 6.6 has been chosen. The degeneration technique has several advantages over the other matching techniques. It is simple and requires only one supplementary series component. A current-mirror concept is applied to bias the transistor M1. The load capacitance is used to adjust the gain resonance frequency.

6.4.1 DC Biasing

At first, the chosen topology involves a choice of biasing voltages. The V_D,V_G,V_S,V_B voltages and I_{DS} have to be set for each transistor. The scheduling of biasing is the following:

- The value of $I_{DS_1} = I_{DS_2}$ is let as a top-level input variable. The current I_{DS_3} flowing in M_3 is set to be 10 times lower to I_{DS_1} flowing in M_1.
- The inductor equivalent resistance (Fig. 6.8) is used to compute the voltage at nodes S_1 and D_2.

$$\begin{cases} V_{S_1} & = I_{DS_1} r_{mp_{deg}} + \dfrac{I_{DS_1}}{2} r_{s1_{deg}} \\ V_{D_2} & = V_{DD} - \left(I_{DS_1} r_{mp_{out}} + \dfrac{I_{DS_1}}{2} r_{s1_{out}} \right) \end{cases} \tag{6.13}$$

- We chose $V_{DS_1} = V_{DS_2}$.

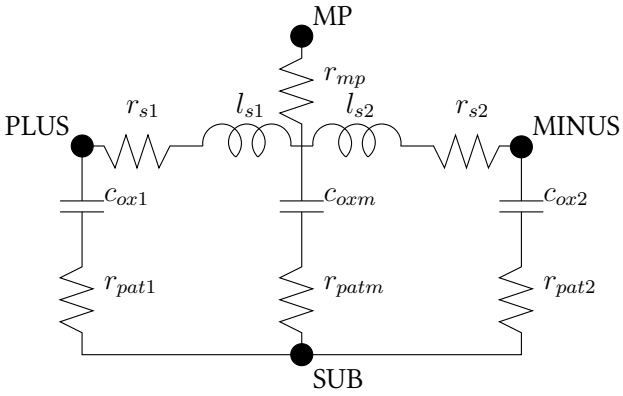

Fig. 6.8: Differential inductor pi model.

$$\begin{cases} V_{D_1} - V_{S_1} &= V_{D_2} - V_{S_2} \\ V_{S_2} &= V_{D_1} \end{cases} \Rightarrow \begin{cases} V_{D_1} &= V_{D_2} - V_{S_2} + V_{S_1} \\ V_{S_2} &= V_{D_1} \end{cases} \Rightarrow \begin{cases} V_{D_1} &= \left(V_{D_2} + V_{S_1}\right)/2 \\ V_{S_2} &= V_{D_1} \end{cases} \tag{6.14}$$

- The M2 gate is biased to V_{DD}, the M1 gate is determined by specifying the effective gate-source voltage $V_{EFF1} = V_{GS1} - V_{TH1}$. V_{EFF1} is an important parameter, it has been used in the optimization procedure. It is set as a top-level variable to let the optimization procedure to modify this parameter.

$$V_{G_2} = V_{DD} \tag{6.15}$$

6.4.2 Transistor and Passive Elements Sizing

The next procedure manages the parameters for transistor sizing. Before calling the transistors sizing operator, the saturation conditions are verified. The procedure can throw a C++ exception in case of no saturation (because of a V_{EFF1} too high) or a sizing failed (because of a width too low or too high regarding to the chosen technology). The parameter L_{M1} is set as a top-level input variable. This way, it can be adjusted during the optimization procedure. The parameters L_{M2} and L_{M3} are set at minimum size $0.13\mu m$. Thus, in a first stage, the number of fingers is set at 1 and the sizing operator is called. In a second stage, the number of fingers is computed depending on the resulting transistor width and the sizing operator is called again. The value of R_1 is set at $5k\Omega$ to prevent the current to flow in this section. The value of R_{ref} is chosen to verify Eq. (6.16).

$$R_{ref} = \left(V_{DD} - V_{G_3}\right) / I_{DS_3} \tag{6.16}$$

L_{out} is set at $5nH$. The value of the passive elements L_g, L_{deg}, C_{out} is set as a top-level input variable, because they are related to the performance as it will be presented in the next procedures.

6.4.3 Performance Evaluation

The performance is computed from the matrices of performance, which are linked to a topology, so they just need to be generated once. The matrices of performance are generated considering the RF extended BSIM3v3 model presented in Fig. 6.9 and the inductor pi-model presented in Fig. 6.8. A symbolic solving was too long to succeed as the RF model contains internal nodes that make the circuit very complex, for this reason, the performance is solved numerically for one value of frequency thanks to the small-signal parameters. For linear performance evaluation, the result is not approximated, that's why the frequency response matches perfectly, also at a very high frequency. For nonlinear performance evaluation, there are approximations due to the number of considered harmonics, moreover just g_m and g_{ds} are considered nonlinear and just the dominant coefficient is taken into account. Those approximations justify the loss of precision of nonlinearity model when the amplitude is high, but they are acceptable to get a precise evaluation in a reasonable range of amplitudes. We show in the design examples how performance evaluation matches to transistor-level simulation.

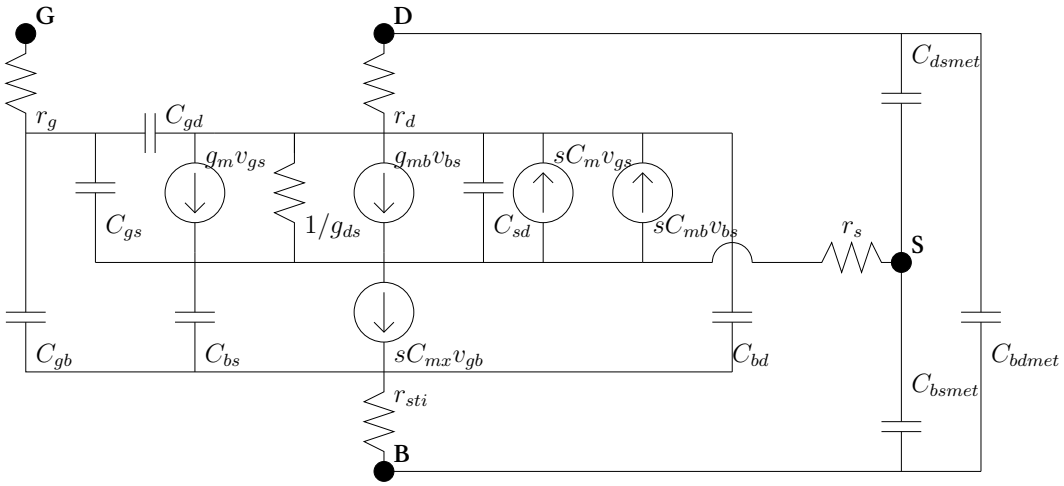

Fig. 6.9: CMOS BSIM3v3 transistor model extended for RF applications.

6.4.4 Optimization Procedure

This last procedure implements converging or scanning loops that try to satisfy a constraint (a specification to achieve) by varying a parameter. The link between the variable parameter and the constraint is determined by the designer. In the case of the LNA, the real part of input impedance is linked to L_{deg} inductance value, the imaginary part is linked to L_g, the gain value is linked to C_{out}. The first established constraints to satisfy are: a 50Ω input impedance and maximize the gain. We implemented converging loops that find the required L_{deg}, L_g and C_{out}. A second level of optimization implements scanning loops, that means that the parameters will scan a range of values, for each iteration, all the performance results are recorded. It is used when the designer want to explore the influence of a parameter to a performance. This way, an intelligent parameters selection can be done, with constraints to specifications: $NF <$ value, $Gain >$ value, $IIP_3 >$ value. To browse all the optimized and valid designs, this first level of optimization: impedance matching, gain maximization, is called at each iteration of the second level.

6.4.5 Design examples

In this section, the results of circuit design examples are represented with the central frequency of 2.4 GHz using the 0.13 μm CMOS technology with a power supply of 1.0 V. Two kinds of LNA L_{deg} inductor models have been experimented: a differential inductor connected directly from the source of M_1 to the source of M_4 (Fig. 6.10) and two single ended inductors, each inductor connected from each source to the ground (Fig. 6.11). Fig. 6.12 to 6.14 present the gain, NF and IIP_3

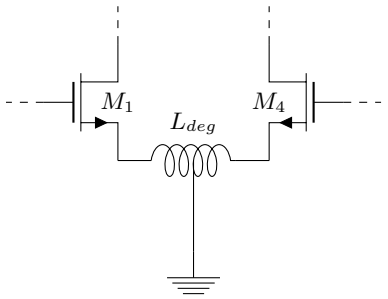

Fig. 6.10: One of the targeted circuit topology: cascode LNA with a differential inductive source degeneration.

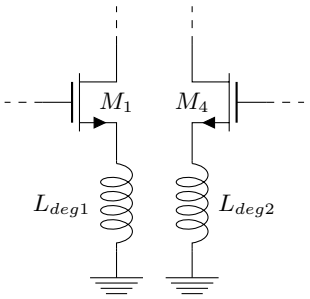

Fig. 6.11: One of the targeted circuit topologies: cascode LNA with two single-ended inductive source degeneration.

performance results for the differential inductor model after the execution of the scanning loop. Fig. 6.15 to 6.17 present the gain, NF and IIP_3 performance results for the differential inductor model after the execution of the scanning loop. Because the structure changes when using this new model of inductance, the matrices of performance are regenerated. In terms of execution speed, one point in the graphics is generated in 12 seconds, as it selects the optimal L_{deg} and L_g for impedance matching and the optimal C_{out} for gain maximization.

A first set of performance is plot in Fig. 6.12. The figures are 4 dimensional, the axis are the 3 variable parameters: I_{DS1}, $V_{EFF1} = V_{GS1} - V_{TH}$, L_{M1}, the fourth dimension is the color representing the analyzed performance: Gain, NF, IIP_3. Designers have to keep in mind that the analyzed performance does not follow exclusively the three parameters variation I_{DS1}, V_{EFF1}, L_{M1}. But also, for each iteration, an impedance matching and gain maximization is called. When impedance matching was unsuccessful or the saturation conditions was not achieved, the design is not validated and will not be recorded to the data file. That's why there are some empty areas in the figures.

About the effect of changing the model of L_{deg} inductor(s). Comparing Fig. 6.12 and 6.15, we can conclude that the use of single ended inductors can achieve a higher gain, until 46 dB, because the impedance matching stays valid when the L_{M1} parameter is low and V_{EFF1} is high. Whereas, the nonlinearity could be unacceptable.

The specifications to achieve, presented in Eq. (6.17), are the constraints that permitted to make the choices for the three variable parameters: I_{DS1}, V_{EFF1}, L_{M1}. Thus, we have chosen to analyze the performance in detail, by setting one of the three variable dimensions, in the following Fig. 6.13 and 6.14, L_{M1} parameter has been set to $0.35\mu m$, respectively the parameter I_{DS1} has been set to $1mA$, as they are good candidates for achieving the constraints.

$$\begin{cases} Gain > 29 \text{ dB} \\ NF < 2.2 \text{ dB} \\ IIP_3 > -9 \text{ dBm} \end{cases} \tag{6.17}$$

Moreover in the 3-D plots presented in Fig. 6.13 and 6.14, the colored points represent the designs that verify the constraints of Eq. (6.17). The designer can simply select one of these points to be

sure to get a valid design that follows the constraints. The analysis has been reproduced with the another topology in Fig. 6.16 and 6.17.

Thereby, a design selection could be done by visualizing the 4-D plots, then setting one dimension and visualizing the more detailed 3-D plots, finally selecting the two remaining dimensions to comply the constraints presented in Eq. (6.17).

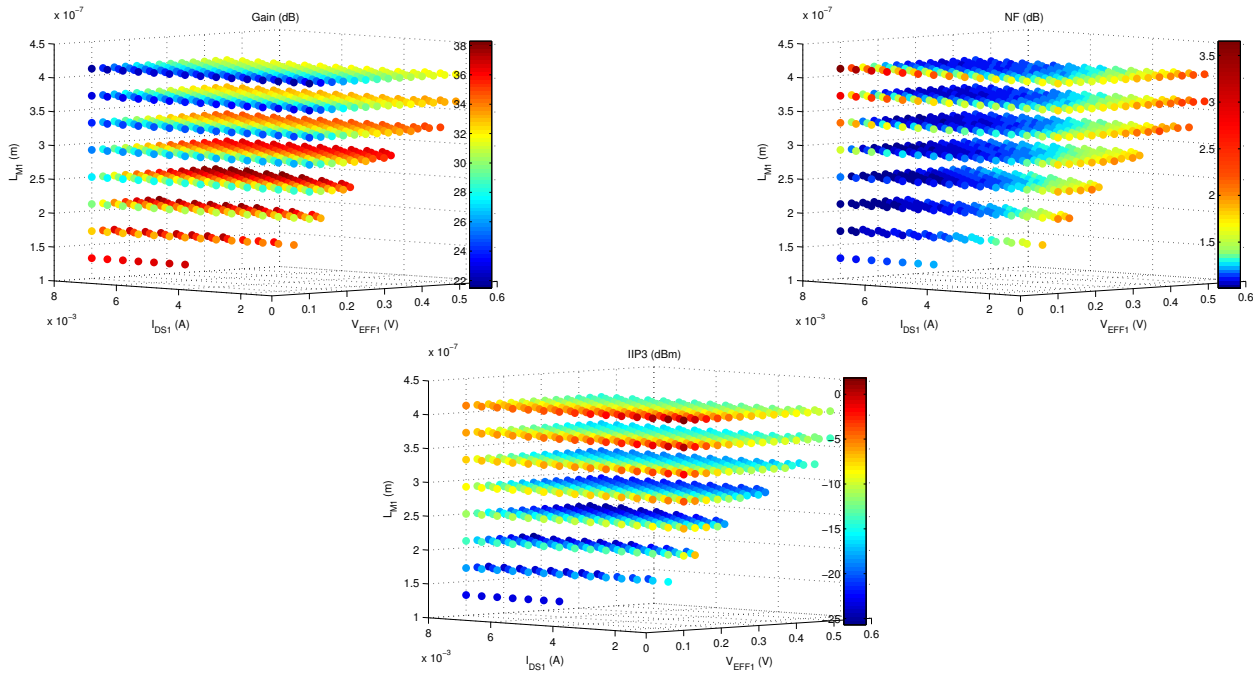

Fig. 6.12: Differential inductor model: 4 dimensional figure: {Gain,NF,IIP_3} in relation to {I_{DS1},L_{M1},V_{EFF1}}, the 4th dimension is color.

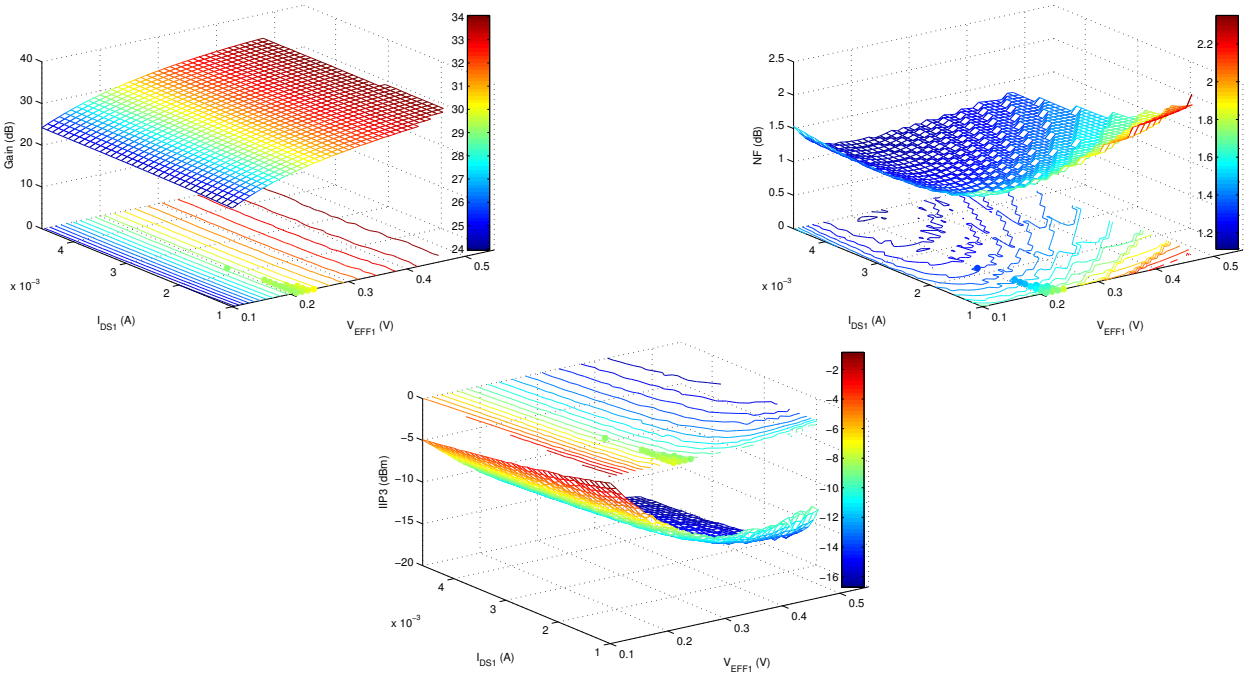

Fig. 6.13: Differential inductor model 3 dimensional figure: {Gain,NF,IIP_3} in relation to {I_{DS1},V_{EFF1}}, L_{M1}=0.35μm.

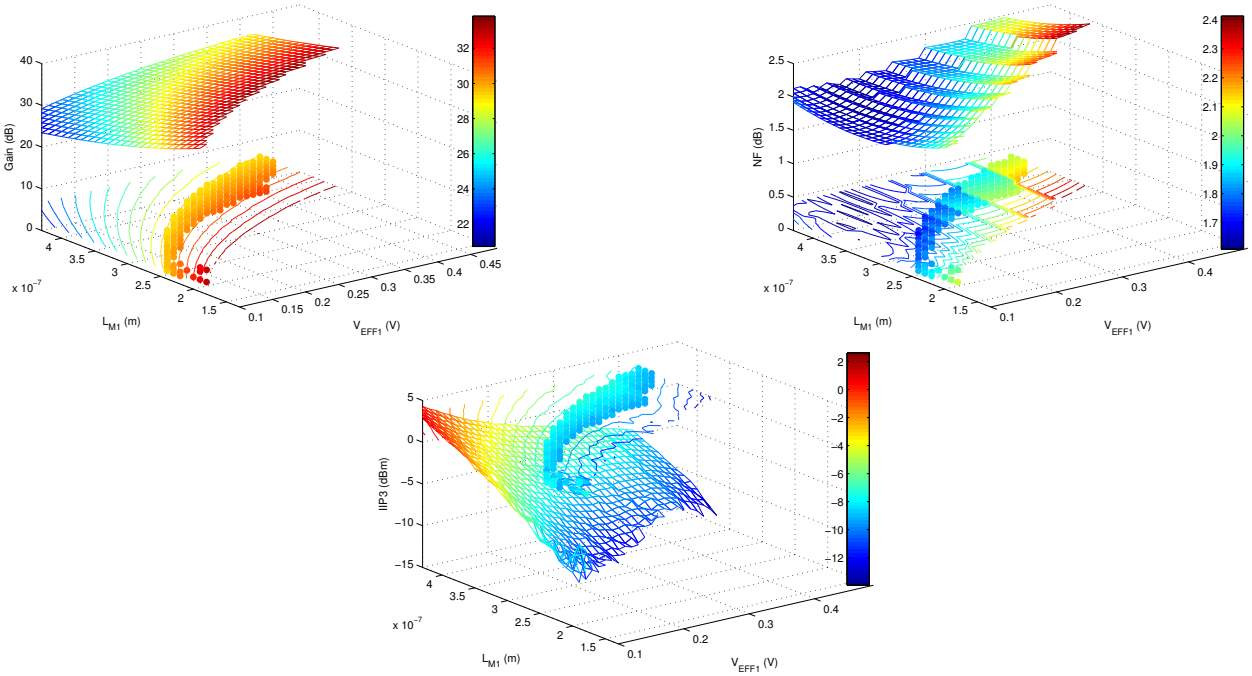

Fig. 6.14: Differential inductor model: 3 dimensional figure: {Gain,NF,IIP_3} in relation to {L_{M1},V_{EFF1}}, I_{DS1}=1 mA.

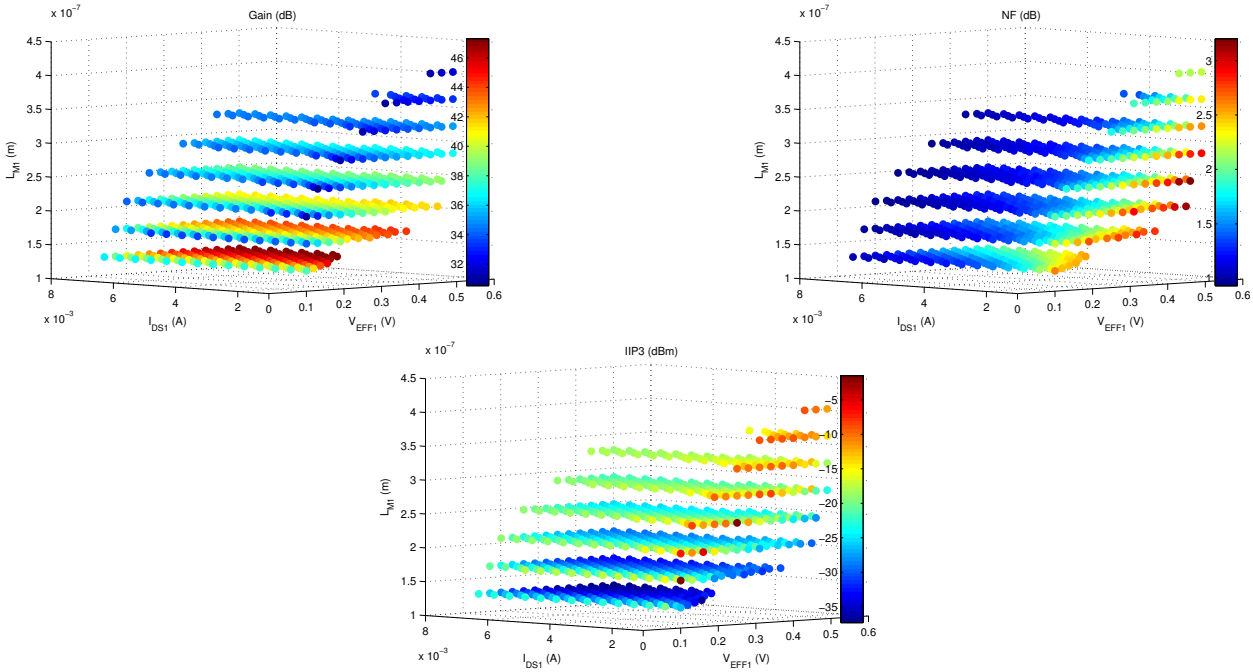

Fig. 6.15: Single-ended inductor model: 4 dimensional figure: {Gain,NF,IIP_3} in relation to {I_{DS1},L_{M1},V_{EFF1}}, the 4th dimension is color.

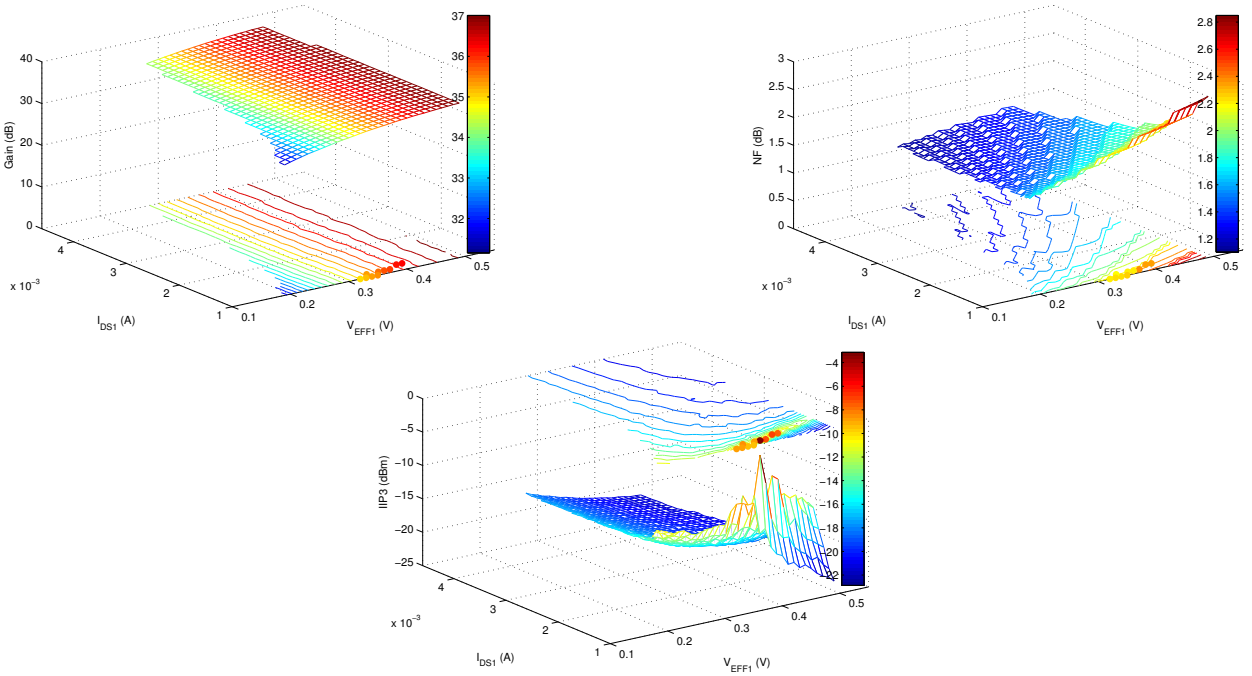

Fig. 6.16: Single-ended inductor model: 3 dimensional figure: {Gain,NF,IIP_3} in relation to {I_{DS1},V_{EFF1}}, L_{M1}=0.29μm.

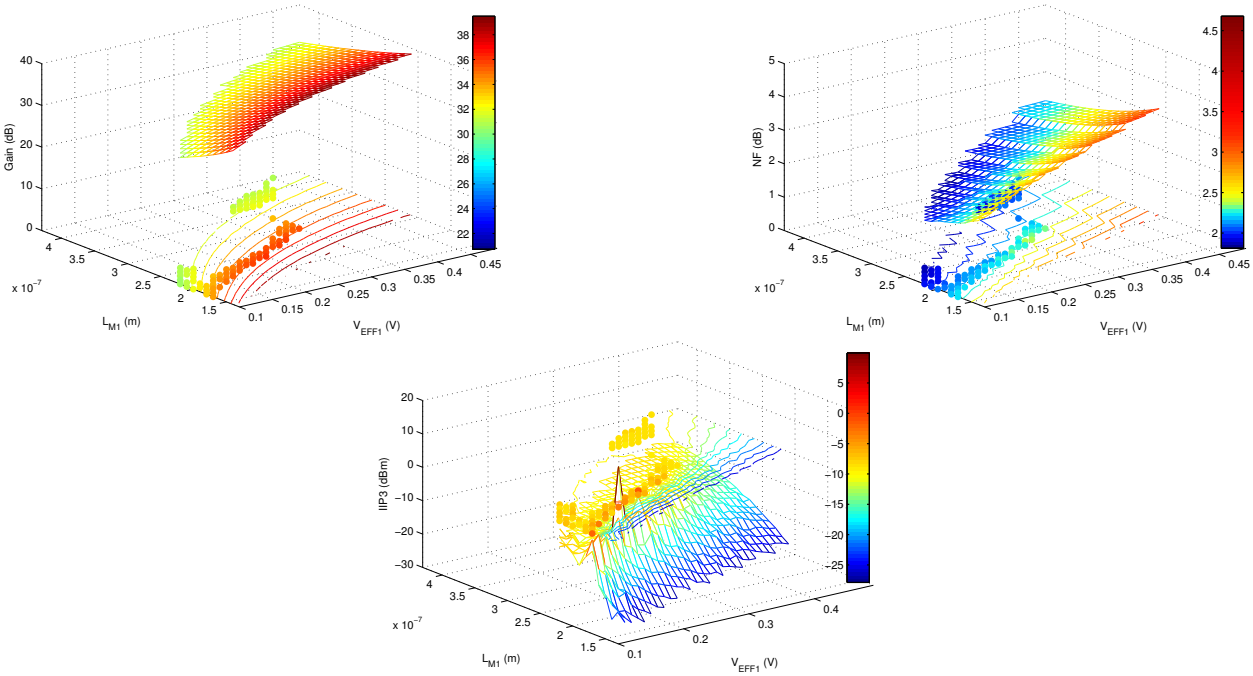

Fig. 6.17: Single-ended inductor model: 3 dimensional figure: {Gain,NF,IIP_3} in relation to {L_{M1},V_{EFF1}}, I_{DS1}=1 mA.

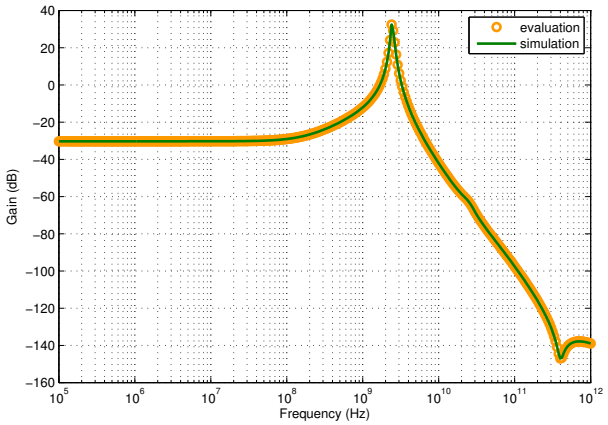

Fig. 6.18: LNA gain frequency response with L_{M1}=0.21 μm, I_{DS}=1 mA, V_{EFF1}=0.12 V.

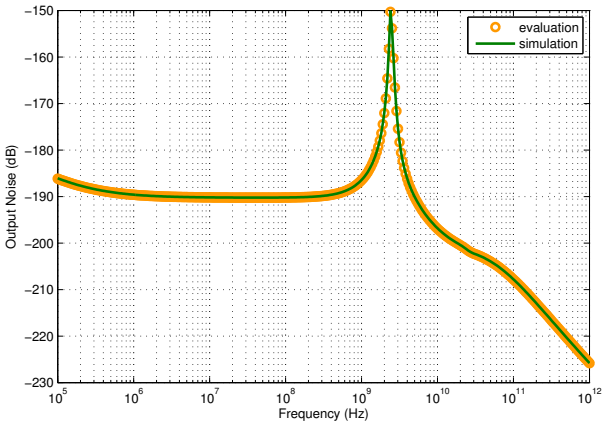

Fig. 6.19: LNA output noise frequency response with L_{M1}=0.21 μm, I_{DS}=1 mA, V_{EFF1}=0.12 V.

We selected one design from the differential L_{deg} inductor topology by setting the variable parameters to one value:

$$I_{DS1} = 1 \text{ mA}, \qquad V_{EFF1} = 0.12 \text{ V}, \qquad L_{M1} = 0.21\mu m \tag{6.18}$$

We can confirm the high precision of the performance evaluation comparing to the simulation as it has been presented in Fig. 6.18 to 6.22, for the analyzed performance: Gain, Noise, Input Impedance and IIP3. Fig. 6.23 illustrates the difference between the evaluated and simulated IIP_3 performance. The synthesized circuit parameters are presented in Table 6.4 and the performance in Table 6.5.

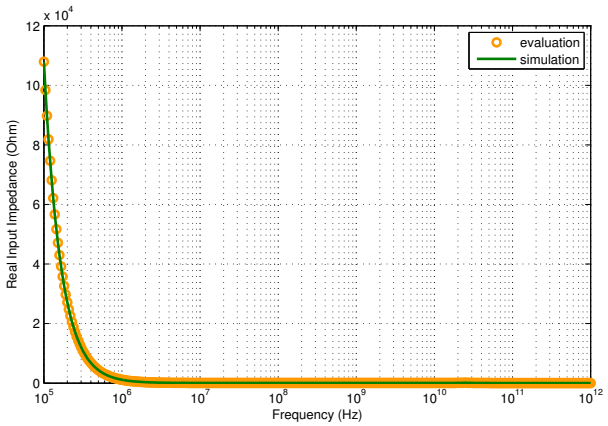

Fig. 6.20: LNA real-part impedance frequency response with L_{M1}=0.21 μm, I_{DS}=1 mA, V_{EFF1}=0.12 V.

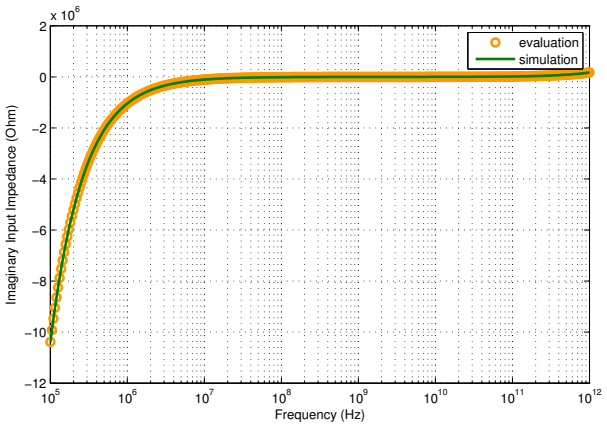

Fig. 6.21: LNA imaginary-part impedance frequency response with L_{M1}=0.21 μm, I_{DS}=1 mA, V_{EFF1}=0.12 V.

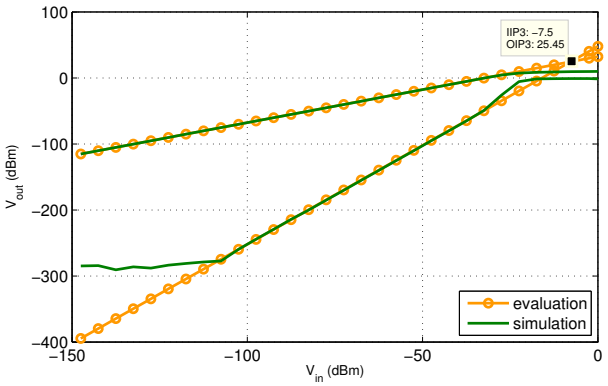

Fig. 6.22: LNA 1^{st} order harmonic and 3^{rd} order intermodulation product with L_{M1}=0.21 μm, I_{DS}=1 mA, V_{EFF1}=0.12 V.

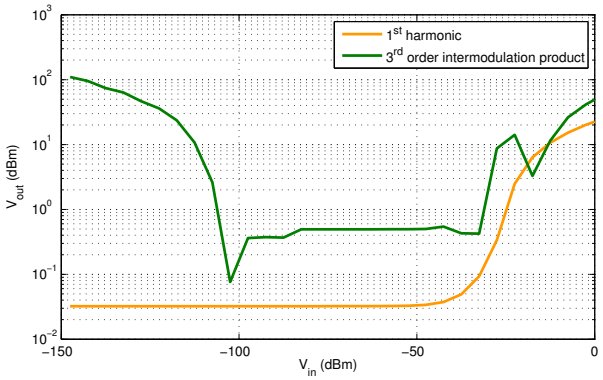

Fig. 6.23: LNA 1^{st} order harmonic and 3^{rd} order intermodulation product (in dB) difference between simulation and evaluation with L_{M1}=0.21 μm, I_{DS}=1 mA, V_{EFF1}=0.12 V.

Table 6.4: Optimized design LNA circuit-level parameters.

Frequency (GHz)	2.4
I_{DS} (A)	0.001
V_{DD} (V)	1.0
V_{EFF1}=V_{GS1}-V_{TH1} (V)	0.12
L_{deg} (nH)	0.7483
L_g (nH)	27.58
L_{out} (nH)	5.000
C_{out} (pF)	1.448
R_{ref} (Ω)	6579
W_{M1}/L_{M1} ($\mu m/\mu m$) | $n_{finger1}$	53.31/0.21 | 15
W_{M2}/L_{M2} ($\mu m/\mu m$) | $n_{finger2}$	9.184/0.13 | 9
W_{M3}/L_{M3} ($\mu m/\mu m$) | $n_{finger3}$	3.728/0.13 | 6

Table 6.5: Optimized design LNA performance.

	Evaluation	Simulation
Gain (dB)	32.42	32.42
Input impedance (Ω)	50.05+ 0.03932i	50.05+ 0.03932i
NF (dB)	2.101	2.140
IIP_3 (dBm)	-7.765	-7.534

6.5 Conclusion

In this chapter, it is shown how the linear and nonlinear performance evaluation method, developed in Chapter 5, is used to optimize analog and RF circuits. The proposed technique is applied to GmC integrator design for $\Sigma\Delta$ ADC and Low Noise Amplifier design for RF tramsceiver.

7

Unified Multi-Level Design Environment for Mixed Signal Systems

7.1 Introduction

In the previous chapters, a model refined system-level design environment and a systematic and optimized circuit-level design environment, have been presented. The final step is to unify both environments into a multi-level design platform. The realization of such platform is facilitated by two assets of our environments. At first, they are both implemented in C++, helping the integration and making the platform completely seamless. Secondly, the specifications and performance are the only data that are managed. This allows to build the multi-level design flow presented in Fig. 7.1, for

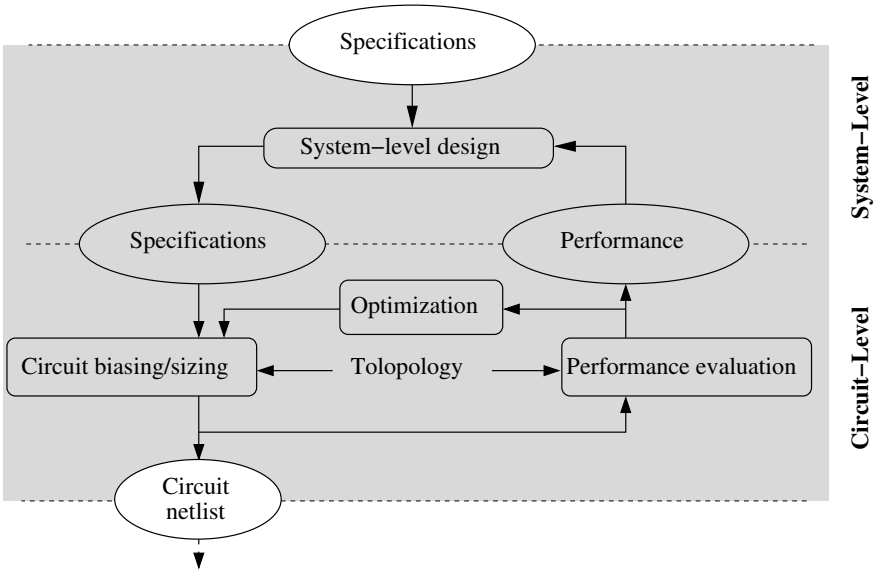

Fig. 7.1: The multi-level design flow with both top-down and bottom-up processes.

which the specifications produced by system-level are sent to the circuit-level and the performance results produced by the circuit-level are sent to the system-level back-annotating the refined models.

The chapter describes, at first, the unified multi-level design flow, merging the circuit-level optimized design flow into the system-level design. Finally, two case studies illustrate the work that has been implemented and presented. The case studies are chosen to follow the target of a WSN node design. Thereby, two essential components have been selected to be designed in the multi-level design flow, a GmC integrator for $\Sigma\Delta$ modulator in the ADC part of the node and the LNA for an RF ZigBee receiver in the RF part of the node.

7.2 The Unified Multi-Level Design Flow

The flow manages both system and circuit levels combining top-down and bottom-up processes. A detailed representation of the flow is presented in Fig. 7.2. The flow is based on a simulation-based system-level design flow using the refined models presented in Chapter 4, and on the equation-based design flow presented in Chapters 5 and 6.

In a first step, the top-level specifications, related to the constraints from the targeted application, for example, a communication standard, are incorporated, eventually through theoretical computation, in the system-level model. The device model is simulated with the initial conditions of the refined components model, those initial conditions make the models ideal. The performance is compared to the specifications, as long as the system-level model does not validate the specifications, the model is modified by changing some parameters or by changing the architecture.

In a second step, the system-level simulations and the top-level specifications having produced the circuit specifications, they are incorporated into the DC biasing and transistors sizing procedures to generate a sized netlist and to allow a performance evaluation. By modifying the circuit parameters, and eventually the topology or the technology, we can produce an optimal circuit design.

A last step performs back-annotation of the system-level components with the performance that has been evaluated in the circuit level. The back-annotation consists of filling the evaluated performance into the system-level refined models. This way, the system-level models are linked to

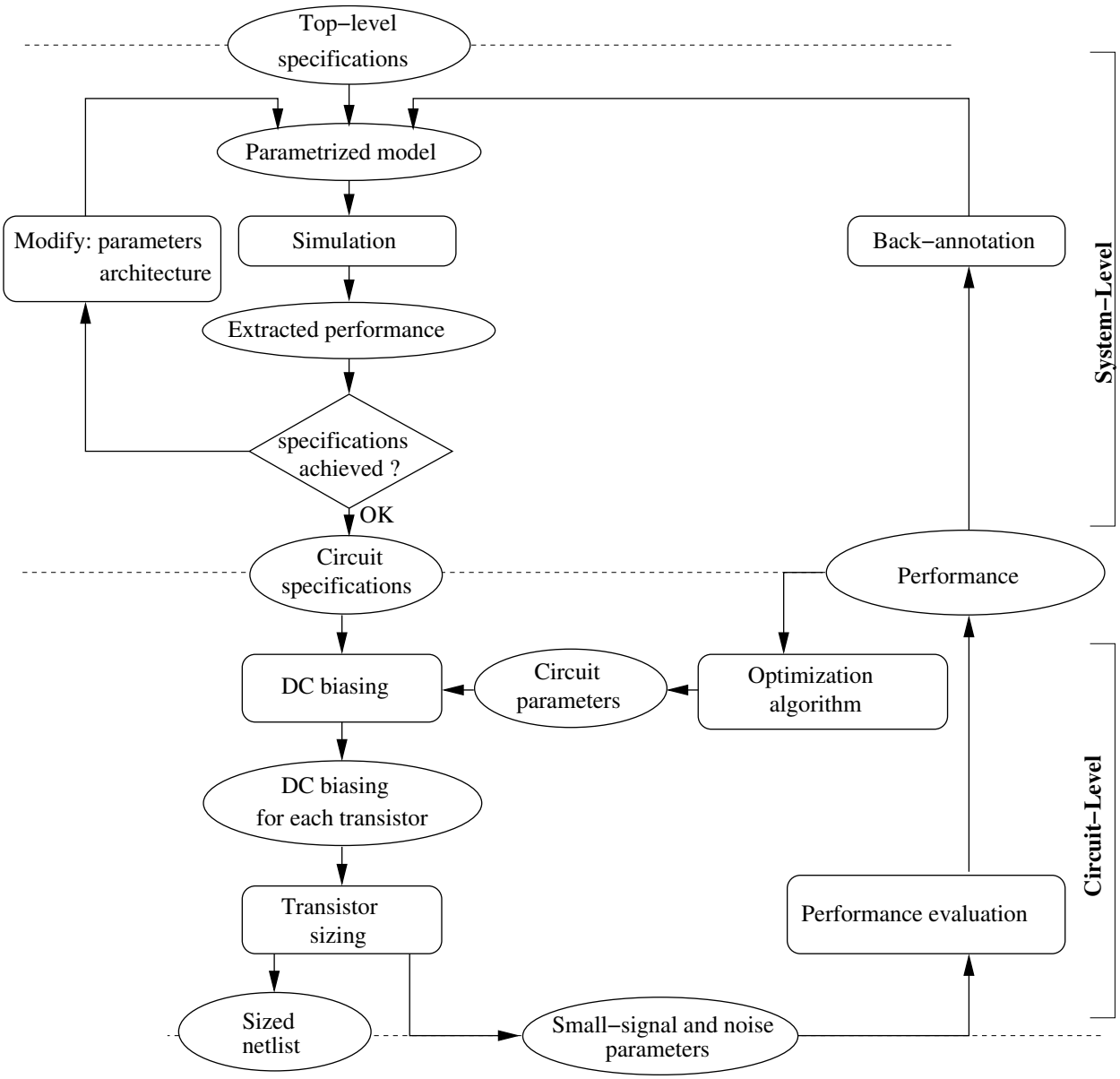

Fig. 7.2: The generic unified multi-level design flow.

the chosen circuit-level topology and technology with its expected performance. A precise system-level simulation can then be performed to validate the overall system architecture.

The multi-level design flow has been implemented for the particular case of WSN node design. As depicted in Fig. 7.3, two instances are presented in the following sections as case studies: the GmC integrator design for continuous-time $\Sigma\Delta$ ADC and the LNA design for RF ZigBee transceiver. Each of these instances constitutes one case study that will now be described.

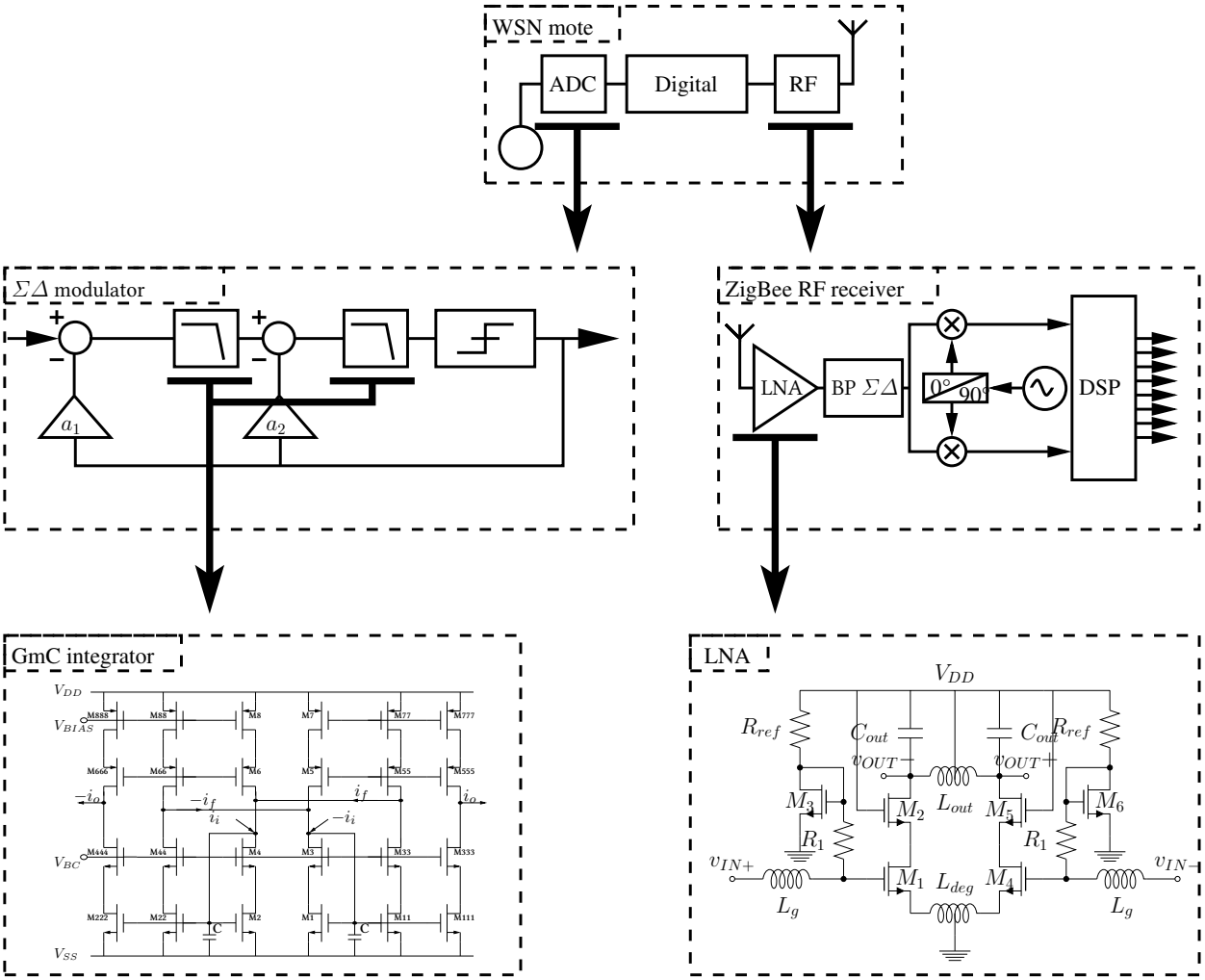

Fig. 7.3: A wireless sensor network node, the link between the presented two design examples: A GmC integrator for a $\Sigma\Delta$ ADC and a LNA for a ZigBee RF receiver.

7.3 Case Study I: GmC Integrator Design for Sigma-Delta ADC

7.3.1 Design Flow

The GmC integrator unified multi-level design flow uses a specifications/performance communication between the levels as depicted in Fig. 7.4. At the system level, the integrator performance is validated inside the 2^{nd} order continuous-time $\Sigma\Delta$ ADC model with a refined model of the integrators.

A detailed description of the proposed unified multi-level design flow chart, applied to a $\Sigma\Delta$ modulator with GmC integrator, is shown in Fig. 7.5. The system-level design consists of simulat-

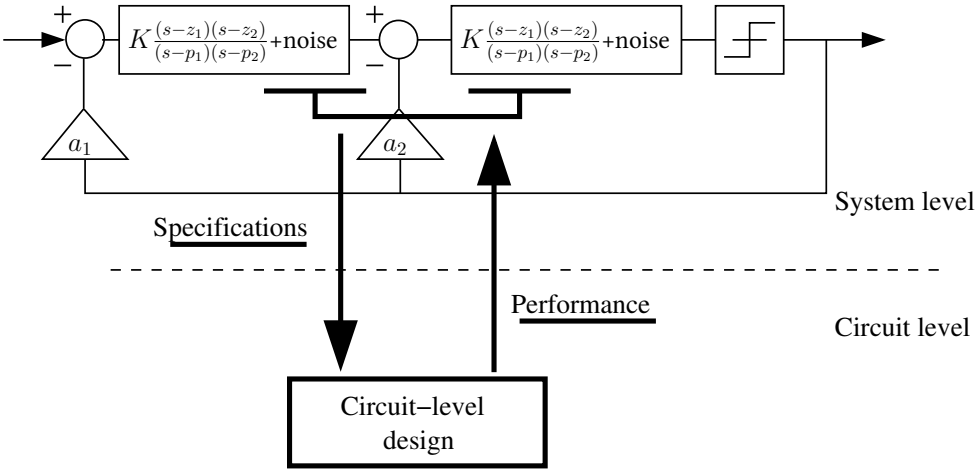

Fig. 7.4: The unified multi-level design flow involving specifications and performance exchange for the $\Sigma\Delta$ modulator with GmC integrator.

ing the $\Sigma\Delta$ model into a testbench, to get the performance and eventually modify parameters to optimize the performance. The analyzed performance is SNR, it can be linked to a set of input parameters: the oversampling rate: OSR, the input amplitude of the ADC: $A_{\Sigma\Delta}$, the integrators gain: A_{int}, the order of the $\Sigma\Delta$ ADC. The system-level description is implemented with SystemC AMS (Chapter 3) using the Timed Data Flow (TDF) model of computation (MOC). The integrator model is described using the generic refined modeling technique presented in Chapter 4.

The circuit-level design is optimized following the constraint: $SNR_{cir} > 60$, by modifying the length of transistors, as it has been described in Section 6.3.

The back-annotation process is called after finding a circuit-level design to validate, by system-level refined model simulation, the SNR performance. The back-annotated integrator performance are Poles, Zeros and Noise.

7.3.2 Results

The system-level back-annotated model simulation permitted to refine the design of the GmC integrators as described in the following two points. At first, the design of 1^{st} column was supposed to be valid regarding to the noise level as the theoretical SNR is $61.66dB$. However, the $\Sigma\Delta$ output spectrum, presented in Fig. 7.6, illustrates a problem that is related to the presence of a pole close to the f_T. It is confirmed by the simulated SNR=$51.27dB$ in the results table. It is always hard

to make a theoretical and quantitative link between the position of the GmC integrators 2^{nd} pole and the effect to the $\Sigma\Delta$ ADC SNR performance. That's why, the simulation of the $\Sigma\Delta$ ADC was unavoidable.

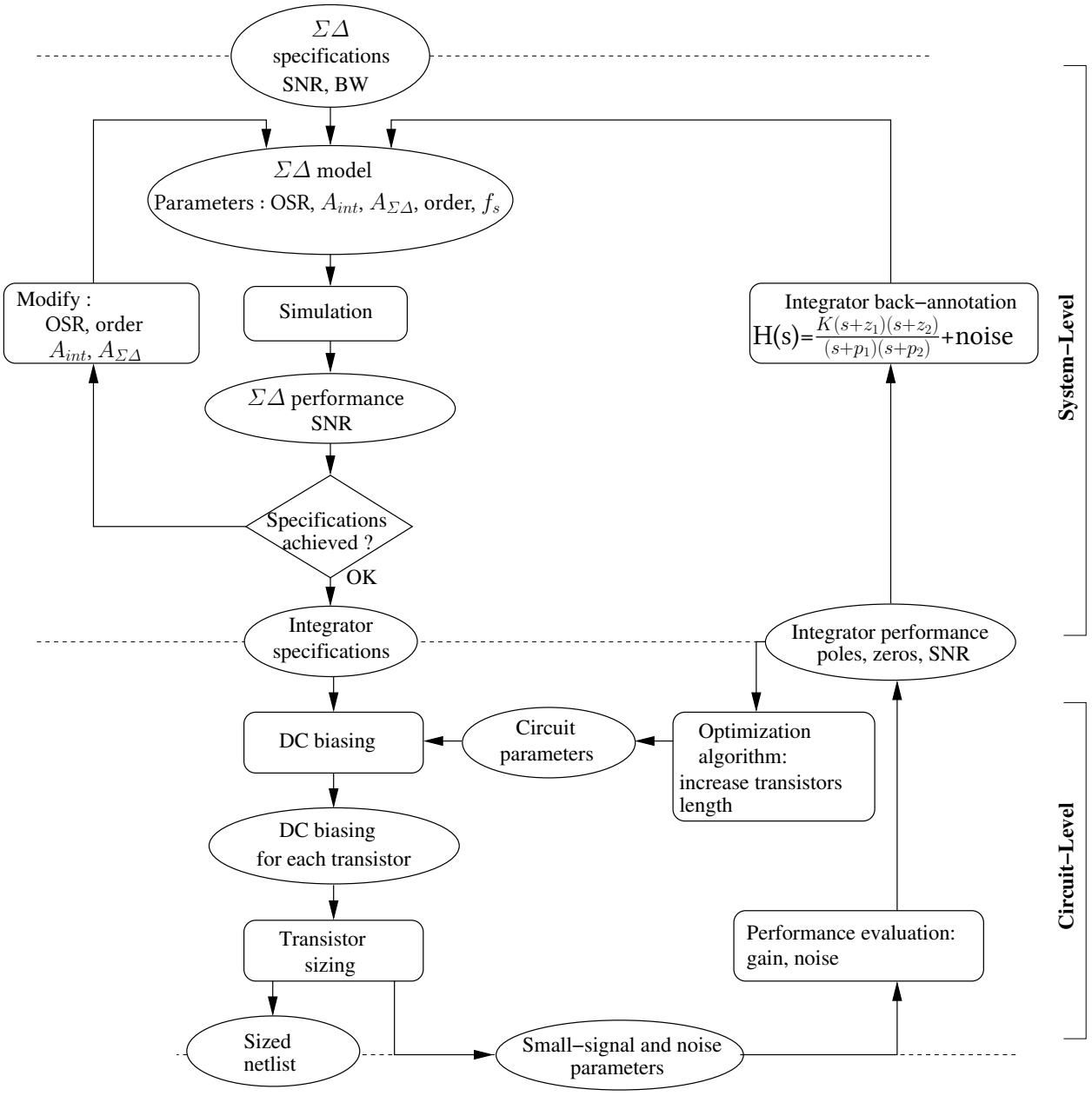

Fig. 7.5: The proposed C++ based environment for the unified multi-level design flow of a GmC integrator in the context of $\Sigma\Delta$ modulator design.

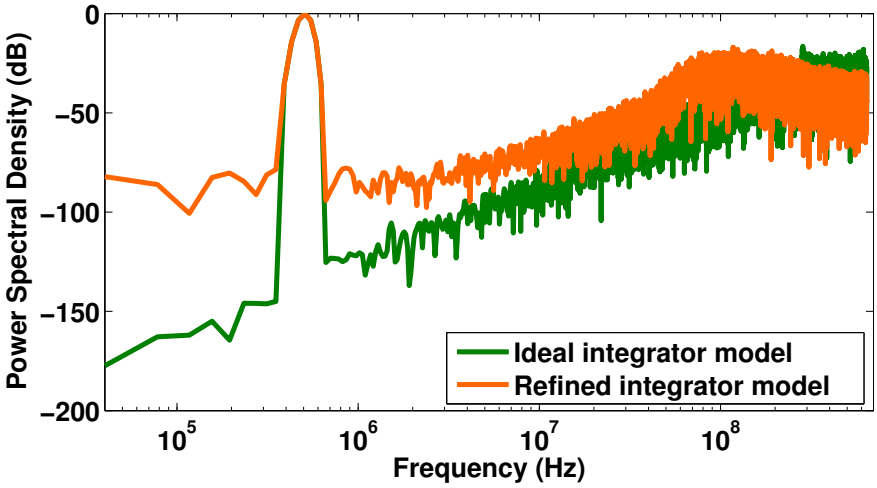

Fig. 7.6: 2^{nd} order $\Sigma\Delta$ modulator output spectrum for an ideal integrators model and a poles/zeros and noise back-annotated integrators model. Case of the cascode transistors length too high (first column of Table 6.2).

Secondly, when the 2^{nd} pole is sufficiently rejected to high frequency (designs of columns 2 and 3), the theoretical SNR matches with the system-level simulated SNR. We can observe the $\Sigma\Delta$ output spectrum of design 2, respectively 3, in Fig. 7.7, respectively in Fig. 7.8 In this case, the use

Table 7.1: Integrators circuit characteristics, comparing three different transistors length settings ($0.13\mu m$ CMOS process) with Vdd=1.2V.

	Cascode transistors length too high	Transistors length too low	Automatically optimized transistors length
W1/L1 (μm/μm)	122.2/1.18	12.8/0.13	181.1/0.73
W3/L3 (μm/μm)	894.3/1.30	30.0/0.13	113.6/0.13
W5/L5 (μm/μm)	1463.0/0.60	92.0/0.13	337.0/0.13
W7/L7 (μm/μm)	112.3/0.23	26.8/0.13	240.9/0.23
I_0 (mA)	1.3	7.4	2.7
Desired f_T (MHz)	67.9	67.9	67.9
2^{nd} pole (MHz)	480.4	5575.0	3255.7
C (pF)	13.0	13.5	38.9
SNR_{cir}	61.66 dB	41.66 dB	61.77 dB
Total SNR	51.27 dB	41.74 dB	61.41 dB

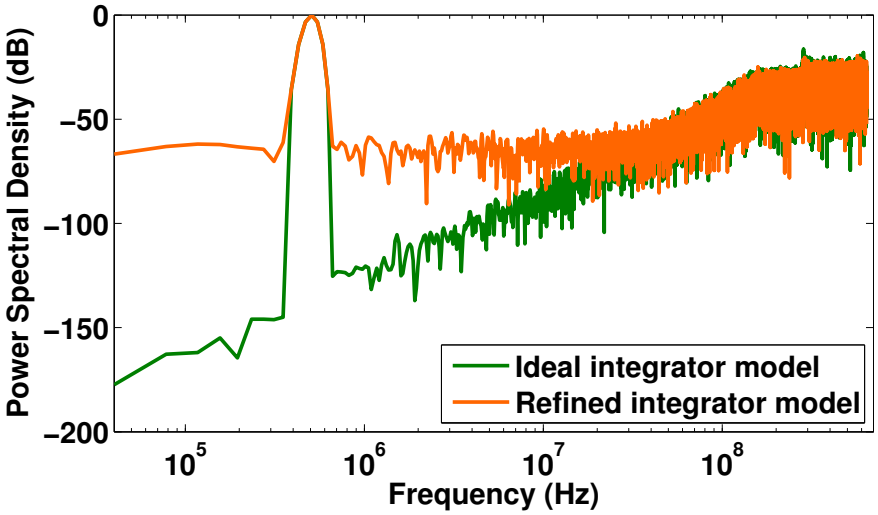

Fig. 7.7: 2^{nd} order $\Sigma\Delta$ modulator output spectrum for an ideal integrators model and a poles/zeros and noise back-annotated integrators model. Case of the transistors length too low (second column of Table 6.2).

of system-level refined model of integrators is helpful to validate the hypothesis of the theoretical SNR presented in Eq. (6.2). In fact, this formula made some approximations: it considers a simplified transistor noise model, it supposes that every transistors have the same noise as M_1, it simply makes the sum of each contribution instead of building a transfer function of contributions. In the designs that have been matching it was a valid approximation, nevertheless it is always better to confirm this theory in an entire $\Sigma\Delta$ simulation.

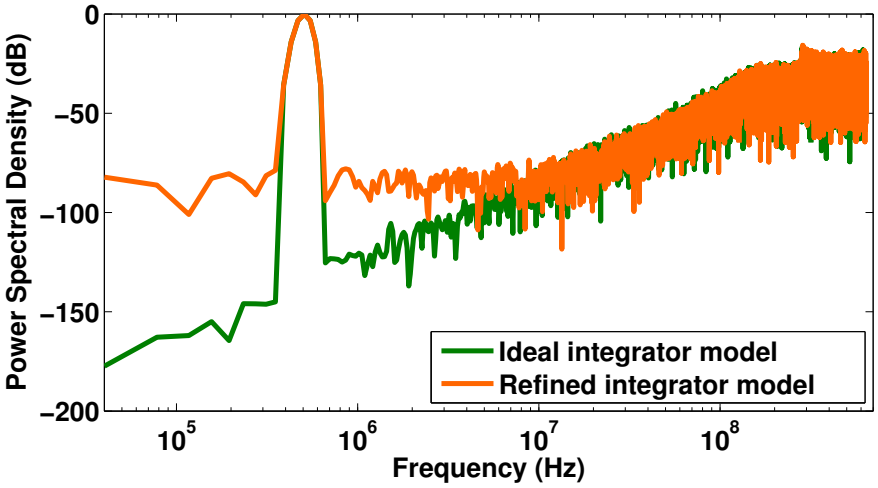

Fig. 7.8: 2^{nd} order $\Sigma\Delta$ modulator output spectrum for an ideal integrators model and a poles/zeros and noise back-annotated integrators model. Case of the automatically optimized transistors length (third column of Table 6.2).

This case of designs highlighted the usefulness of the method, because the higher level of hierarchy simulation ($\Sigma\Delta$ ADC) was helpful for validation and unavoidable in some cases. As it would have spent a long time in a transistor level simulation, this hybrid solution of simulating at system level with back-annotated circuit-level performance is the best compromise between speed and precision.

7.4 Case Study II: Low Noise Amplifier Design for ZigBee RF Transceiver

The second case study is related to the design of a LNA for an RF receiver that follows the ZigBee specifications. In Section 6.4, the systematic circuit-level design flow of a LNA has been presented, the technique has been applied to several design examples with two different topologies. This section shows the whole ZigBee RF receiver design and illustrates the usefulness of a back-annotated LNA model for fast and accurate system-level simulations. In a first part, we describe the design flow, then we describe the RF receiver architecture, finally we present the results.

7.4.1 Design Flow

The RF receiver architecture, shown in Fig. 7.10, is based on the RF $\Sigma\Delta$-based architecture proposed in [Beilleau09] [Ashry11]. As presented in Fig. 7.9, to apply the unified multi-level design flow to the

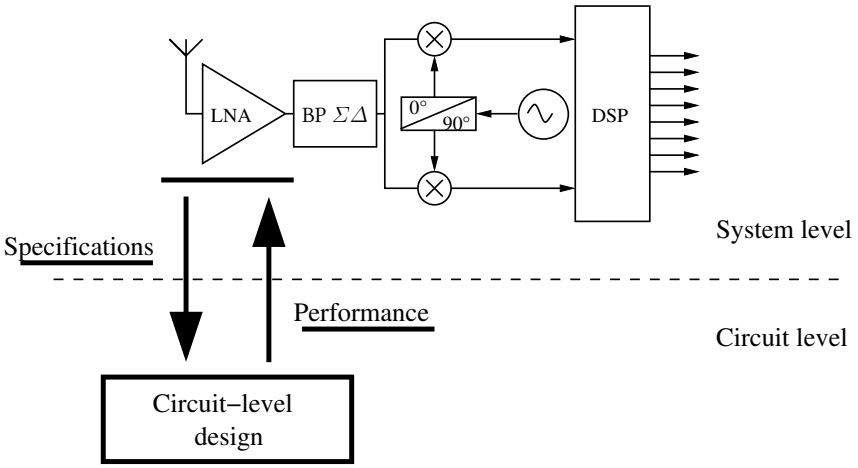

Fig. 7.9: The LNA unified multi-level design flow involving a data specifications/performance production/consumption by the ZigBee RF receiver system-level model.

RF receiver circuit, the first step was to determine the specifications of the LNA, the second step was to design the circuit to satisfy these specifications and the last step was the back-annotation of the high-level models for a fast verification of the overall performance. RF circuit design is a relatively complex procedure involving several design constraints and trade-offs between parameters such

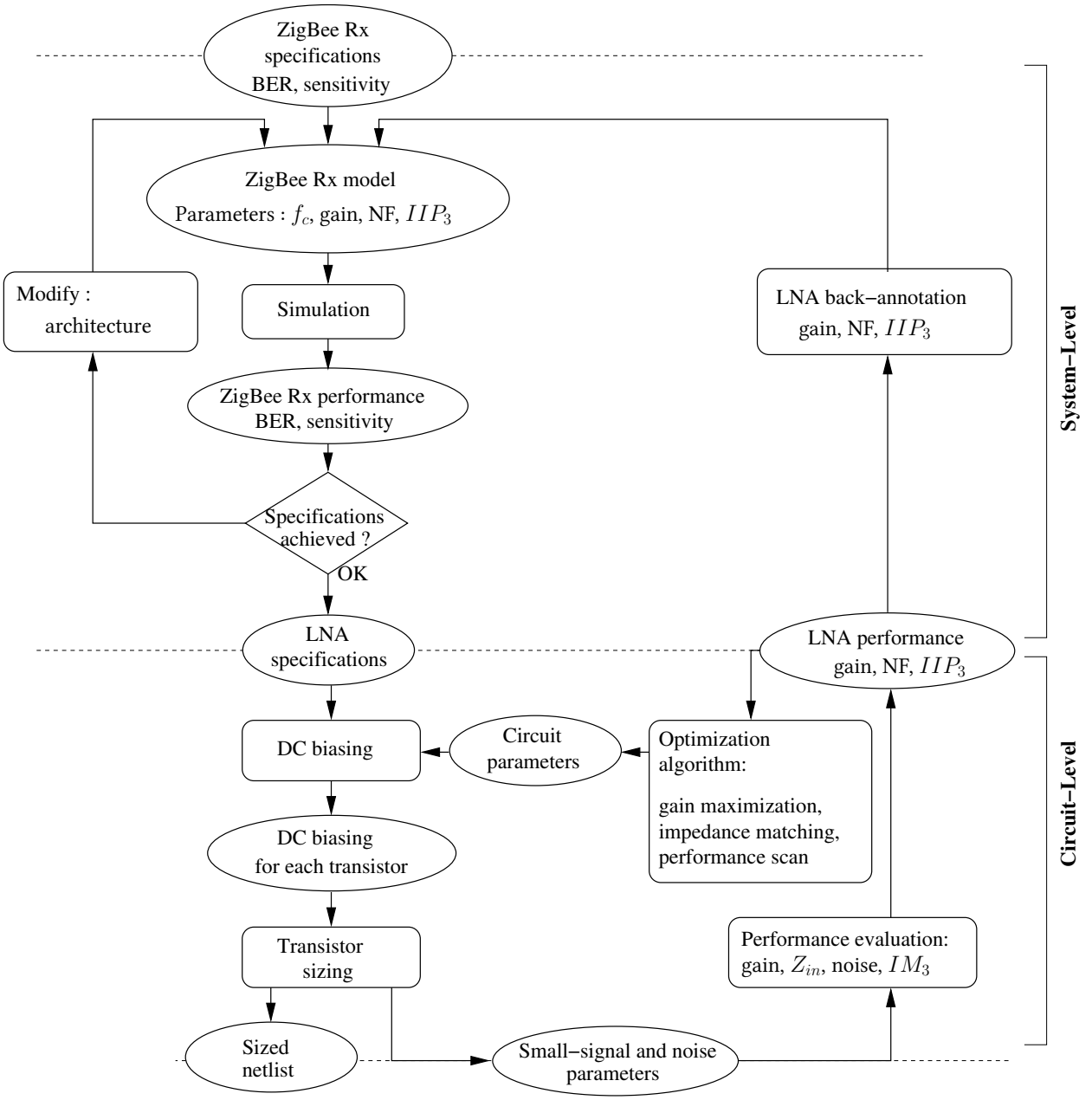

Fig. 7.10: The proposed C++ based environment for the unified multi-level design flow of a LNA in the context of ZigBee RF receiver design, (Simulation → SystemC AMS, Synthesis → CHAMS, Performance evaluation → GiNaC).

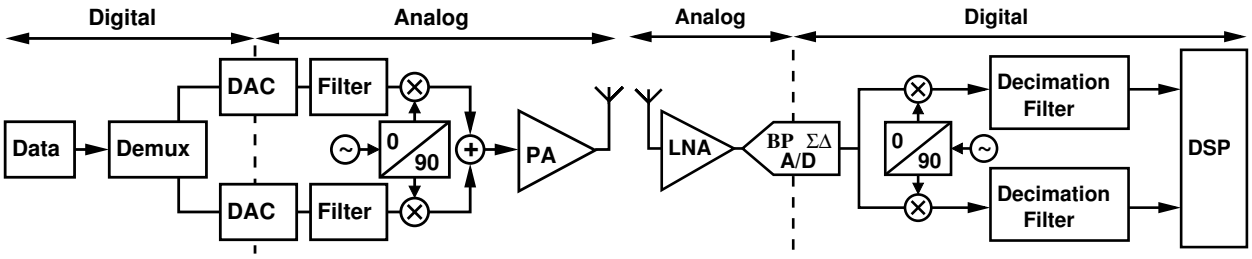

Fig. 7.11: RF transmitter and receiver designed for ZigBee standard in a context of Software-Defined Radio (SDR).

as Gain, NF, IIP_3, power consumption and input impedance [Leenaerts01]. The presented unified multi-level design flow takes into account most of those constraints.

A detailed description of the proposed unified multi-level design flow chart, applied to a LNA for ZigBee RF transceiver, is shown in Fig. 7.10. By specializing the design flow for the LNA design, the specifications and performance are documented. In the case of a ZigBee receiver design , Bit-Error Rate (BER) evaluation in a range of sensitivity is necessary to validate the ZigBee standard [Hafez09]. The LNA specifications are Gain, NF and IIP_3.

The work that has been done in Chapter 4, to provide executable specifications like Gain, NF, IIP_3, allowed us to easily back-annotate the LNA with the previously presented performance analyzed from the circuit level. A model of a whole node containing digital components and back-annotated analog components opens the possibility to verify the hypothesis expressed by the experience of the designer and finally to validate an architecture of RF receiver for ZigBee standard.

7.4.2 ZigBee RF Receiver Architecture And Implementation

The chosen RF receiver architecture, presented in Fig. 7.11 is dedicated to Software Defined Radio (SDR), is based on an RF $\Sigma\Delta$ ADC [Beilleau09, Ashry11]. It digitizes the RF signal, directly after the LNA, to perform down-conversion and the filtering in the digital domain.

A 4^{th} order undersampled RF $\Sigma\Delta$ ADC, illustrated in Fig. 7.12, is used in this architecture. The $\Sigma\Delta$ 1-bit output is down-converted with a digital mixer. The decimator performs the lowpass filtering behavior thanks to the decimation filter and with a rate of OSR.

ZigBee transmitter and receiver models have been implemented in SystemC AMS language with the TDF model of computation. Fig. 7.13 illustrates the different modules and simulation settings

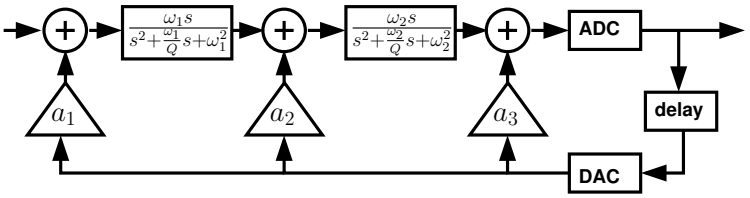

Fig. 7.12: $4th$ order bandpass $\Sigma\Delta$ modulator architecture.

that have been chosen. When looking at the simulation frequency of Fig. 7.13 the ratio between bitstream and RF simulation frequency is $38.4e^9/250e^3$ = 153600. Which means that, 153600 samples are simulated for each bit. The amount of samples to generate is extremely high. When a transmission of $1e^5$ bits is necessary to be simulated, the time spent to simulate this transmission is around ten hours. Anyway, this is a good result when keeping in mind the level of details that is simulated.

7.4.3 Results

We characterized the RF bandpass $\Sigma\Delta$ modulator by matching the model to the measured SNR performance of the fabricated chip presented in [Ashry11]. The specifications and performance are presented in Table 7.2. Fig. 7.14 represents the measured SNR of $\Sigma\Delta$ modulator output signal for different input powers expressed in dBm. The maximum SNR is achieved with an input close to $-20dBm$, this optimal configuration was used to represent the $\Sigma\Delta$ output spectrum in Fig. 7.15. Fig. 7.14 also permitted to determine the $\Sigma\Delta$ input dynamic range equal to the theoretical LNA output dynamic range, it covers a power from -59 dBm to -15 dBm.

This analysis helps the designer to choose the LNA gain constraint. Table 7.3 brings together the computation done to predict the necessary LNA gain to satisfy the ZigBee sensitivity constraints.

Table 7.2: Specifications and performance of 4^{th} order $\Sigma\Delta$ modulator, measurement results from [Ashry11].

BW	$25MHz$
OSR	64
f_c	$2.4GHz$
maximum SNR	$40dB$

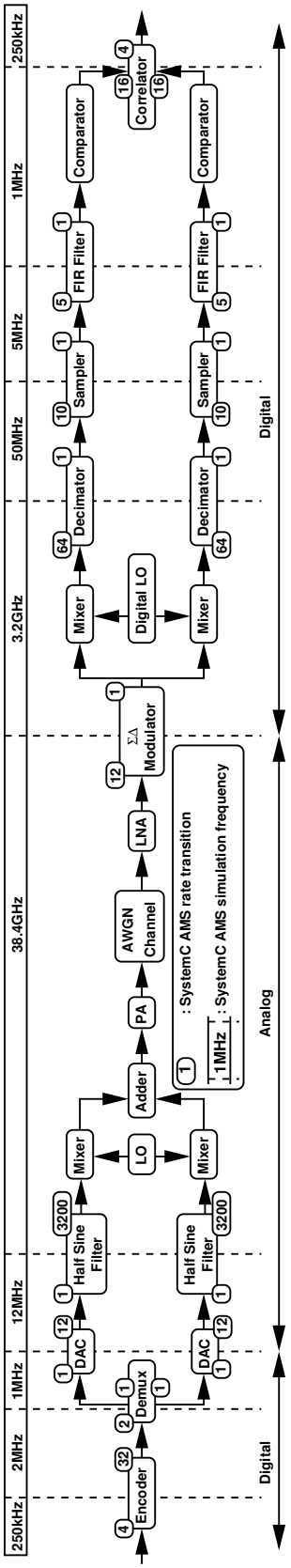

Fig. 7.13: RF transmitter and receiver SystemC AMS models designed for ZigBee standard.

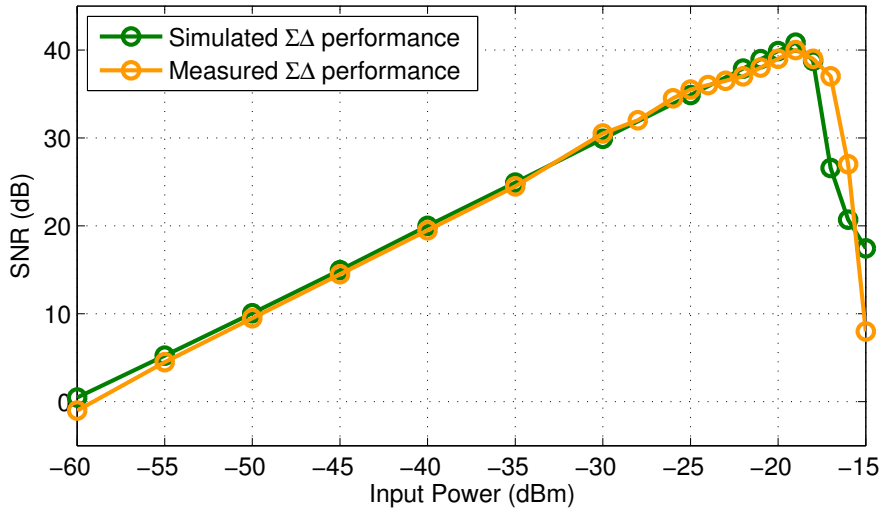

Fig. 7.14: 4^{th} order bandpass $\Sigma\Delta$ modulator SNR in relation to input power. Comparison between SystemC AMS refined model and measurement results [Ashry11].

The ZigBee specification requires an input signal to be detected in a range of -85 dBm to -20 dBm. With a margin of 3 dB, to acquire correctly a ZigBee signal, we need a gain higher than (-59+3)-(-85)=29 dB for the lowest power ZigBee signal and a gain lower than (-15-3)-(-20)=2 dB to acquire the highest power signal. The LNA needs a variable gain, this has been possible by changing the value of output capacitance.

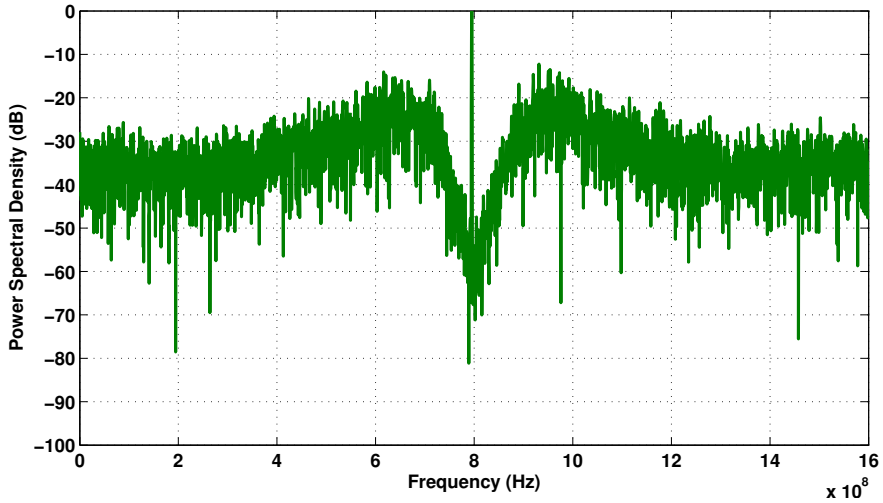

Fig. 7.15: .

Table 7.3: Theoretical LNA gain computation to follow ZigBee standard specifications.

	Minimum	Maximum
ZigBee receiver input power	-85 dBm	-20 dBm
ADC input power	-59 dBm	-15 dBm
ADC input power with 3 dB margin	-56 dBm	-18 dBm
LNA gain	29 dB	2 dB

Those computations refer to an important hypothesis, the $\Sigma\Delta$ dynamic range of -59 dBm to -15 dBm, and with 3dB margin, covers input power values that achieve a satisfying SNR to acquire correctly the signal. In fact, the link has to be done with another important ZigBee standard specification, BER (Bit-Error Rate) < $6.3e^{-5}$. To detect a maximum of 6.3 errors per $1e^5$ bits, we needed to simulate a transmission of $1e^5$ bits.

In a first step of BER analysis, the system is simulated without an LNA. Fig. 7.16 illustrates the BER variation in relation to input power, the dynamic range covers a power from -62 dBm to -16 dBm. This is sensibly close to the supposed dynamic range (-59 dBm to -15 dBm). We do not fulfill

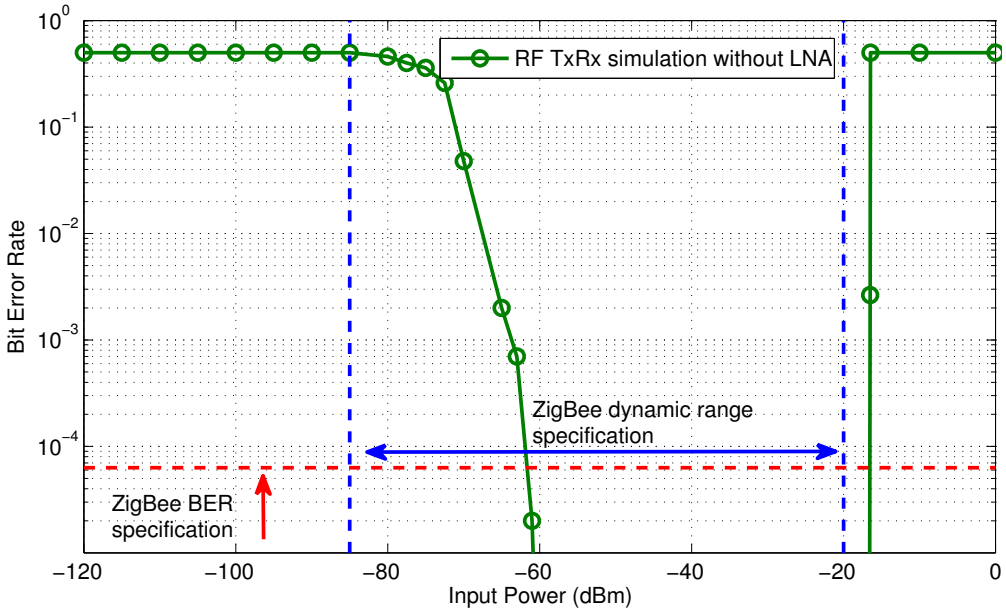

Fig. 7.16: Bit-Error Rate simulation results of a receiver designed for ZigBee standard. The LNA was not included to validate the computed gain constraints.

the ZigBee standard specifications, signal power below -62 dBm are not detected. As it has been said, when commenting the theoretical computations from Table 7.3, the LNA has been built with a variable output capacitance to be able to decrease the gain for the high values of the input power.

In a second step, another BER analysis is done with a back-annotated LNA, the results are represented in Fig. 7.17. A first LNA design called LNA_1 is the one that has been chosen in the previously described design example in Section 6.4.5, with the design parameters and the performance presented in the first column of Table 7.4. We chose a design with a gain = 32.42 dB higher than the constraint gain > 29 dB. The BER analysis results validate the BER ZigBee specifications in the ZigBee dynamic range. A second LNA design called LNA_2 has been chosen to show how a badly designed LNA can deteriorate the performance and how our approach is useful to help the designer to make good choice of designs. We selected a design that complies to the gain constraint but not to the NF and IIP_3 constraint. This second LNA design parameters and performance are presented in the second column of Table 7.4. The BER suffered losses and the dynamic range has reduced, it covers a power from -83 dBm to -24 dBm. The analysis does not validate the ZigBee BER specifications in the required ZigBee dynamic range.

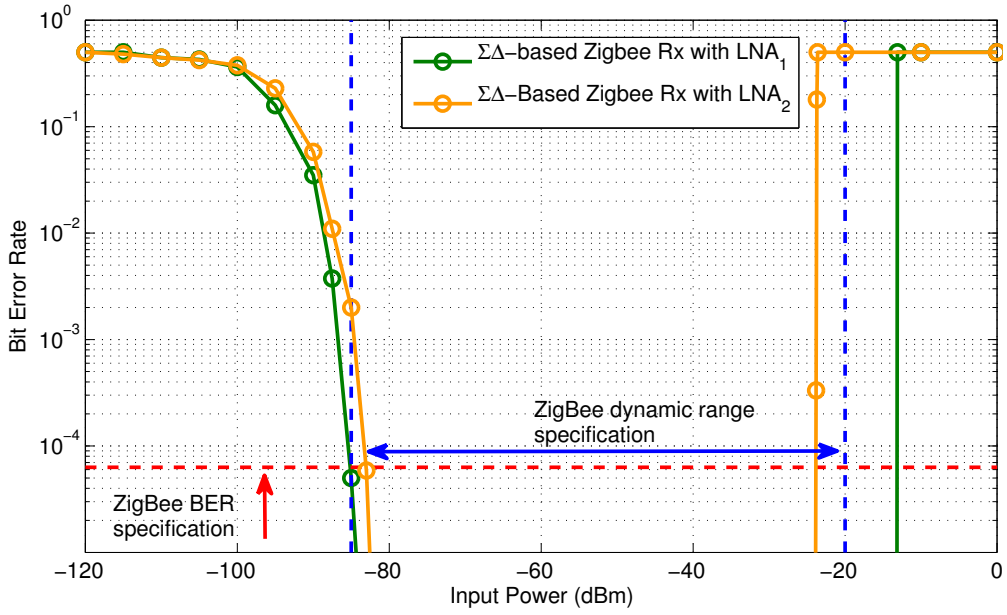

Fig. 7.17: Bit-Error Rate simulation results of a receiver designed for ZigBee standard. Two LNA designs are analyzed, LNA_1: {gain=32.42 dB, NF=2.1 dB, IIP_3=-7.7 dBm}, LNA_2: {gain=39.51 dB, NF=2.9 dB, IIP_3=-29.93 dBm}.

Table 7.4: Design parameters and performance of the two presented LNA designs.

	LNA_1	LNA_2
I_{DS1} (mA)	1.0	1.0
V_{EFF1} (V)	0.12	0.31
L_{M1} (μm)	0.21	0.21
Gain (dB)	32.42	39.51
NF (dB)	2.101	2.900
IIP_3 (dBm)	-7.765	-29.93

The design based on a theoretical gain constraint presented in Table 7.3 was not sufficient to make a ZigBee RF receiver validation. Whereas, a back-annotated system-level model simulation was able to select more precisely the valid LNA designs for the ZigBee standard as they considered Gain, NF and IIP_3 constraints. Moreover, the validation is done very fast as it runs a system-level simulation.

7.5 Conclusion

The chapter exposed the unification of the model refined system-level design environment and the circuit-level systematic design environment. This unified multi-level design flow implements a communication of the specifications and performance between the both system and circuit levels. Moreover, the entire platform is based on C++ making the implementation completely unified.

Both case studies, the GmC integrator designed for a $\Sigma\Delta$ ADC and the LNA designed for a ZigBee RF receiver, presented in the previous chapter for circuit-level optimized design, have been designed into the unified multi-level design flow. They revealed the usefulness of the approach as they permitted a true system and circuit level co-design.

8

Conclusion and Future Work

8.1 Conclusion

A unified multi-level design flow dedicated to Mixed Signal systems has been presented. The flow combines optimized circuit sizing and biasing thanks to a fast and accurate performance analysis, and model refinement for system-level precise simulations of Mixed Signal systems. The circuit-level flow operates with a biasing/sizing tool, related to technology and specifications, and a performance evaluation equation-based tool that can evaluate linear as well as nonlinear performance to prevent from the exploding time for the design when using several optimizing loops. The system-level models are back-annotated with the optimized circuit design performance, they are simulated with SystemC AMS to allow system-level Mixed Signal systems fast simulation. By the exclusive use of C++, the whole environment is built as a unified platform, allowing a perfect interoperability and a capacity of evolution for eventual extensions of the analyzed performance and the optimization algorithms.

Two case studies, related to Wireless Sensor Network nodes design, have been presented. At first, the GmC integrator design for a $\Sigma\Delta$ ADC, optimizing the Poles/Zeros and Noise performance by modifying the transistors length, to achieve the SNR of the $\Sigma\Delta$ ADC. Secondly, the Low Noise Amplifier design for a ZigBee RF receiver, optimizing the Gain, NF, and IIP_3 by modifying the transistors length, the effective gate-source voltage and the biasing current, to achieve the BER of the ZigBee RF receiver. In this second case study, the circuit-level procedure also applies a impedance matching and gain maximization at each iteration, by modifying the inductances and the output capacitance.

The system-level model is one of the first such complex SystemC AMS implementation, as it models, with refinement, a WSN node. At circuit-level, the presented approach introduced a high precision performance evaluation to narrow the selection of potential designs. By choosing an equation-based methodology, the procedures could be unified in a global design environment, increasing the speed when a several number of optimization loops are processed.

As the time-to-market is continuously shrinking, as advance in process technology and the RF applications are bringing circuit harder and harder to design, the presented platform responds to a real need of flexibility, interoperability and capacity of evolution.

8.2 Future Work

At first, the work can be extended by development of design and optimization procedures for a library of analog and RF blocks. This will provide an additional procedure implementing a smart topology selection phase and could realize a platform-based design flow as suggested in [Sangiovanni-Vincentelli07].

At second, the proposed design flow can be linked with the layout. This link will allow to take into account the post-layout extraction for an extended multi-level design flow, as proposed in [Youssef11].

9

List of Publications

- 2007
  - [Vasilevski07b] M. Vasilevski, F. Pecheux, H. Aboushady, L. de Lamarre, "Modeling Heterogeneous Systems Using SystemC-AMS Case Study: A Wireless Sensor Network Node", International Behavioral Modeling and Simulation Conference, BMAS'07, San Jose, CA, USA, September 2007.(10 citations).
  - [Vasilevski07a] M. Vasilevski, H. Aboushady, F. Pecheux and L. de Lamarre, "Modeling Wireless Sensor Network Nodes Using SystemC-AMS", International Conference on Microelectronics, ICM'07, Cairo, Egypt, December 2007.(7 citations).
- 2008
  - [Vasilevski08b] M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady, K. Einwich, "Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN", Design, Automation and Test in Europe, DATE'08, Munich, Gemany, March 2008.(19 citations).
  - [Vasilevski08a] M. Vasilevski, N. Beilleau, H. Aboushady, F. Pecheux, "Efficient and Refined Modeling of Wireless Sensor Network Nodes Using SystemC-AMS", Ph.D. Research in Microelectronics and Electronics, PRIME'08, Istanbul, Turkey, June 2008.(8 citations).
- 2009
  - [Vasilevski09] M. Vasilevski, H. Aboushady, M.-M. Louerat, "Automatic Model Refinement of GmC Integrators for High-Level Simulation of Continuous-Time Sigma-Delta Modulators", International Symposium on Circuits and Systems, ISCAS'09, Taipei, Taiwan, Mai 2009.(3 citations).

- 2010
  - [Leveque10] A. Leveque, F. Pecheux, M.-M. Louerat, H. Aboushady, M. Vasilevski, "SystemC-AMS Models for Low-Power Heterogeneous Designs: Application to a WSN for the Detection of Seismic Perturbations", International Conference on Architecture of Computing Systems, ARCS'10, Germany, February 2010.(1 citations).
  - [Haghighitalab10] D. Haghighitalab, M. Vasilevski, H. Aboushady, "LNA automatic synthesis and characterization for accurate RF system-level simulation" International Midwest Symposium on Circuits and Systems, MWSCAS'10, USA, Seattle, August 2010.
  - [Massouri10] A. Massouri, A. Leveque, L. Clavier, M. Vasilevski, A. Kaiser, M.-M. Louerat, "Baseband Fading Channel Simulator for Inter-vehicle Communication using SystemC-AMS" International Behavioral Modeling and Simulation Conference, BMAS'10, USA, San Jose, September 2010.

A

SystemC AMS: Timed Data Flow Modeling

A.1 Modeling using Timed Data Flow Model of Computation

The following listings present some TDF modules which represent essential components of an RF transmitter in a simplified way. Each module has been chosen to illustrate some SystemC AMS TDF implementation aspects. The implementation has been intentionally simplified, to present a first view of the language without the complexity of a refined and generic model.

A.1.1 Amplifier Model

Listing A.1 introduces the basics of a SystemC AMS TDF module implementation using the example of the implementation of an amplifier presented in Fig. A.1.

- Line 5: The TDF MoC manages the primitive modules of a data flow hierarchy with the module definition **SCA_TDF_MODULE**, whereas the structure is managed by a classical SystemC **SC_MODULE** definition. Those macros stand for a C++ structure definition, that's why we will talk about methods of the module instead of functions.
- Lines 6 and 7: Input and output ports are declared as transporting **double** values, which are double precision floating point numbers.
- Line 9: An optional **void initialize()** method is automatically called before the simulation starts. It is used to set up some parameters linked to the application.
- Line 12: A method **void processing()** contains the behavior of the module, a simplified linear amplifier computes the output by applying a constant gain to the input.

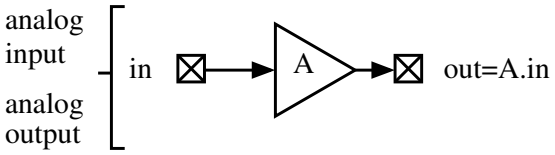

Fig. A.1: The amplifier schematic.

Listing A.1: SystemC AMS implementation of an amplifier.

```
1  #ifndef AMPLIFIER_H
2  #define AMPLIFIER_H
3  #include "systemc-ams.h"
4
5  SCA_TDF_MODULE(amplifier){
6    sca_tdf::sca_in<double> in;
7    sca_tdf::sca_out<double> out;
8    double A;
9    void initialize(){
10     A=10.0;
11   }
12   void processing(){
13     out.write(A*in.read());
14   }
15   amplifier(sc_core::sc_module_name)
16     :in("in"),out("out"){}
17 };
18 #endif
```

A.1.2 1-bit DAC Model

Listing A.2 presents the domain/MoC conversion, the input signal is digital, the output is analog, the MoC is converted from SystemC Discrete Event (DE) to SystemC AMS TDF. The example that is used to illustrate this behavior is a 1-bit DAC presented in Fig. A.2.

- Line 6: The input port is a DE to TDF converter. It should be connected with a **sc_signal**.
- Line 8: The optional **void set_attributes()** method is automatically called before the simulation starts. It is used to set up some parameters linked to the simulation settings. In this case, we called the **void set_timestep(...)** method set up the simulation time step. SystemC AMS assigns every sample to a time point. Constant time distance between samples is assumed. This sampling time must be assigned at least to one port or module of a data flow cluster and is automatically propagated to all other modules during the elaboration process. The presented module will be activated at 10 MHz. Because the input data is digital, we just need one sample per acquired bit to represent the dataflow without losses. At this point, the application signal frequency is 10 MHz, the simulation frequency is 10 MHz.

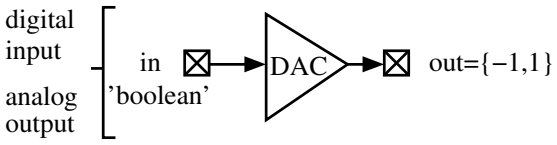

Fig. A.2: The 1-bit DAC schematic.

Listing A.2: SystemC AMS implementation of a 1-bit DAC

```
1  #ifndef DAC_H
2  #define DAC_H
3  #include "systemc-ams.h"
4
5  SCA_TDF_MODULE(dac){
6    sca_tdf::sc_in<bool> in;
7    sca_tdf::sca_out<double> out;
8    void set_attributes(){
9      set_timestep(100,SC_NS);
10   }
11   void processing(){
12     out.write(2.0*double(in.read())-1.0);
13   }
14   dac(sc_core::sc_module_name)
15     :in("in"),out("out"){}
16 };
17 #endif
```

- Line 11: The behavior defined in **void processing()** method is to write a -1 (respectively 1) to the output when the input level is 0 (respectively 1).

A.1.3 Rate Transition

Listing A.3 illustrates the multi-rate data flow features with the example of a rate transition module (Fig. A.3) that oversamples the simulation time step. Multi-rate TDF is especially very well suited for strongly oversampled systems like RF circuits. In a SystemC AMS design with a single clock domain, the **processing()** function of each connected module of a data flow cluster is called periodically at each simulation time step.

To adapt the simulation time step to the application described into the modules, each module can have a different time step based on an integer ratio. This is possible thanks to the **set_rate()** function. This function can be called on a specific module port to define its sample rate, the default configuration is a sample rate of 1. Sample rates control simulation time step, when a given module consumes a different number of samples that it produces. The input/output time step and rate of the module have to satisfy Eq. (A.1).

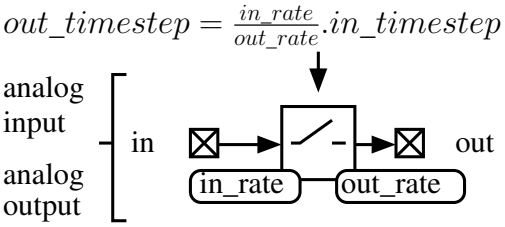

Fig. A.3: The rate transition module schematic.

Listing A.3: The RF rate_transition description.

```
1  #ifndef RATE_TRANSITION_H
2  #define RATE_TRANSITION_H
3
4  SCA_TDF_MODULE(rate_transition){
5    sca_tdf::sca_in<double> in;
6    sca_tdf::sca_out<double> out;
7    double buffer;
8    void set_attributes(){
9      in.set_rate(1);       //1 sample consumed
10     out.set_rate(1000); //1000 samples produced
11   }
12   void processing(){
13     buffer=in.read();
14     for(int i=0;i<1000;i++)
15       out.write(buffer,i);
16   }
17   rate_transition(sc_core::sc_module_name)
18     :in("in"),out("out"){
19     buffer=0.0;
20   }
21 };
22 #endif
```

$$\frac{out\_timestep}{in\_rate} = \frac{in\_timestep}{out\_rate} \tag{A.1}$$

- Line 8: Another kind of TDF attribute that can be set in **void set_attributes()** method is to change the simulation sample rate. This rate assigned to input and output ports (rate=1 by default), represents the number of available samples. If this number of samples is greater than one, we have multi-rate data flow model. The number of samples read from the input ports and written to the output ports are known before the simulation starts. As for single rate simulation, this allows a static scheduling during elaboration, which leads to very high simulation performance.
- Line 12: The method **void processing()** is always activated if enough samples are available at the module input ports. Thus, if the rate attribute of an input port in a module is set to 1, one sample will be available (not necessarily read) per activation, and an output port rate of 1000, means 1000 samples are written per activation. The designer has the freedom to manage each

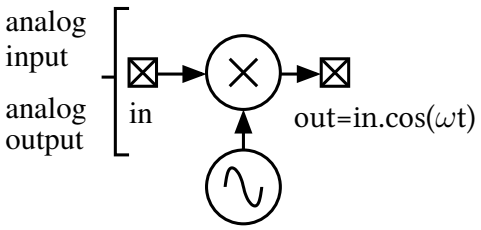

Fig. A.4: The modulator schematic.

Listing A.4: SystemC AMS implementation of a modulator module

```
#ifndef MODULATOR_H
#define MODULATOR_H
#include "systemc-ams.h"

SCA_TDF_MODULE(modulator){
  sca_tdf::sca_in<double> in;
  sca_tdf::sca_out<double> out;
  double fc; // Carrier frequency [Hz]
  void processing(){
    out.write(in.read()*cos(2.0*M_PI*fc*
            out.get_time().to_seconds()));
  }
  modulator(sc_core::sc_module_name,double fc)
    :in("in"),out("out"){
    this->fc=fc;
  }
};
#endif
```

one of those available input samples (using **T in_port.read(int i)** or **T in_port.read()** for rate=1) and has to calculate the value of each output sample (using **void out_port.write(T value,int i)** or **void out_port.write(T value)** for rate=1). Note that the parameter **int i** is the value that index the chosen sample, **T** is the type of the signal (**double**, **int**, ...).

Returning to the example, the signal is going to be mixed with a high frequency (1 GHz) sine wave as explained in Appendix A.1.4. The input signal is low frequency (10 MHz), the output has to be prepared for the high frequency (1 GHz) signal. By considering that 10 samples per period are sufficient for describing the sine wave. To accomplish a simulation of a signal at 1 GHz with 10 samples per period, the simulation frequency needs to be at 10 GHz. With an input simulation frequency at 10 MHz, one input sample will produce 1000 output samples. Choosing the optimum rate, timestep and delay values for the modeled application are the main difficulty to profit from multi-rate data flow in terms of simulation performance. However, it gives the designer interesting degrees of freedom to optimally describe and simulate his application.

A.1.4 Modulator Model

Listing A.4 finalizes the presentation of behavioral models with a high frequency modulator description presented in Fig. A.4. This module does not introduces a new primordial feature but is presented to complete the RF transmitter description.

- Line 9: The method **void processing()** is activated for each sample that is available at the input. Standard C++ functions like **cos(...)** or SystemC/SystemC AMS functions like **get_time()** can be called in the behavior description.

A.2 Hierarchical Modeling

A.2.1 Transmitter Model

Listing A.5 is a classical SystemC structural definition, it implements the example presented in Fig. A.5.

- Line 9: The module definition **SC_MODULE** is used for netlisting the behavioral modules, the user has the freedom of declaring hierarchical levels.
- Lines 10 to 14: The declaration of each input/output port and signal has to be related to the type of the module ports that will be connected. Thus, **in** is declared as DE SystemC input port transporting **bool** values. It is bound to the **dac** module input port. **sig_modulator** is TDF SystemC AMS signal, transporting **double** values. It is bound to the **modulator** module output port and the **amplifier** module input port.
- Line 20: The transmitter constructor is used for the netlist definition.

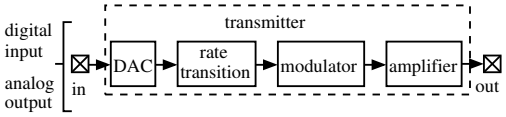

Fig. A.5: The transmitter hierarchical schematic.

Listing A.5: SystemC AMS implementation of a simple transmitter module

```
#ifndef TRANSMITTER_H
#define TRANSMITTER_H
#include "systemc-ams.h"
#include "transmitter/dac.h"
#include "transmitter/rate_transition.h"
#include "transmitter/amplifier.h"
#include "transmitter/modulator.h"

SC_MODULE(transmitter){
  sc_in< bool > in;
  sca_tdf::sca_out<double> out;
  sca_tdf::sca_signal< double > sig_dac;
  sca_tdf::sca_signal< double > sig_rate_transition;
  sca_tdf::sca_signal< double > sig_modulator;
  dac *i_dac;
  rate_transition *i_rate_transition;
  modulator *i_modulator;
  amplifier *i_amplifier;
  SC_CTOR(transmitter)
    :in("in"),out("out"){
    i_dac=new dac("i_dac");
    i_dac->in(in);
    i_dac->out(sig_dac);
    i_rate_transition=
      new rate_transition("i_rate_transition");
    i_rate_transition->in(sig_dac);
    i_rate_transition->out(sig_rate_transition);
    i_modulator=new modulator("i_modulator",1e9);
    i_modulator->in(sig_rate_transition);
    i_modulator->out(sig_modulator);
    i_amplifier=new amplifier("i_amplifier");
    i_amplifier->in(sig_modulator);
    i_amplifier->out(out);
  }
};
#endif
```

A.3 The Testbench

A.3.1 Digital Pulse Source Model

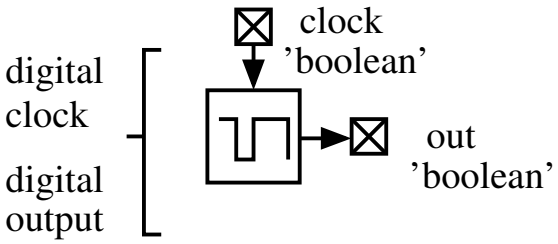

Fig. A.6: The digital pulse source schematic.

Listing A.6: SystemC implementation of a digital pulse source module based on the pattern {0,1,1,0,1,1,0,0}

```
#ifndef PULSE_SRC_H
#define PULSE_SRC_H
#include "systemc.h"

SC_MODULE(pulse_src){
  sc_in<bool> clk;
  sc_out<bool> out;
  bool tab[8];
  int i;
  void do_pulse(){
    out.write(tab[i]);
    i=(i+1)%8;
  }
  SC_CTOR(pulse_src)
    :clk("clk"),out("out"){
    i=0;
    SC_METHOD(do_pulse);
    sensitive<<clk.neg();
    tab[0]=0;
    tab[1]=1;
    tab[2]=1;
    tab[3]=0;
    tab[4]=1;
    tab[5]=1;
    tab[6]=0;
    tab[7]=0;
  }
};
#endif
```

Listing A.6 is a classical DE SystemC behavioral definition of a digital pulse generator illustrated in Fig. A.6, this module has been written for the testbench that will be presented in Listing A.7.

- Line 17: This expression declares the method **void do_pulse()** as a SystemC method with an associated list of sensitivity.
- Line 18: The list of sensitivity is a declaration of ports that will wake up the module by calling a chosen method, when the signal edge becomes negative, positive, or both.
- Line 10: The method has a user-defined name that should be the same as Line 17. It runs the behavior of generating digital values from the pattern {0,1,1,0,1,1,0,0}

A.3.2 The Main Function

Listing A.7 is the testbench of the pulse generator connected to the RF transmitter as described in Fig. A.7.

- Line 5: A SystemC main function is defined by this syntax, it is the entry point for the model execution.
- Line 10: A clock is defined at 10 MHz.
- Line 20: A tabular trace file is defined. It will generate a file called "trace10M.dat". The format of a tabular file is, the first column for the time stamp of each sample, the next column for each signal that is added to the trace file. Each line represents an instant of the simulation, each instant is spaced by the constant time step of the traced signal.
- Line 21: The signal **i_transmitter->sig_dac** is added to the trace file.
- Line 27: By calling **sc_start(...)** function, the elaboration run, then simulation started. With regard to the frequency settings, with 800 ns of simulated time, 8 samples will be generated by the pulse generator clocked at 10 MHz, 8 samples will be generated by the DAC module, 8000 samples will be generated by the modulator and the amplifier modules.
- Lines 28 to 29: The trace file closing function that should be called to be sure that every sample has been written when the execution terminates.

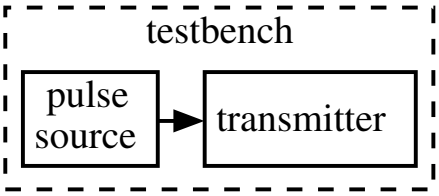

Fig. A.7: The testbench hierarchical schematic.

Listing A.7: Implementation of the "main" function as a testbench that instantiates the previously declared SystemC AMS and SystemC modules

```
#include "systemc-ams.h"
#include "pulse_src.h"
#include "transmitter.h"

int sc_main(int argc,char *argv[])
{
  sc_signal<bool> sig_pulse_src;
  sca_tdf::sca_signal<double> sig_transmitter;

  sc_clock sig_clk("clock",100,SC_NS);

  pulse_src *i_pulse_src=new pulse_src("pulse_src");
  i_pulse_src->clk(sig_clk);
  i_pulse_src->out(sig_pulse_src);

  transmitter *i_transmitter=new transmitter("i_transmitter");
  i_transmitter->in(sig_pulse_src);
  i_transmitter->out(sig_transmitter);

  sca_util::sca_trace_file *tf1=sca_create_tabular_trace_file("trace10M.dat");
  sca_util::sca_trace(tf1,i_transmitter->sig_dac,"dac");

  sca_util::sca_trace_file *tf2=sca_create_tabular_trace_file("trace10G.dat");
  sca_util::sca_trace(tf2,i_transmitter->sig_modulator,"modulator");
  sca_util::sca_trace(tf2,sig_transmitter,"amplifier");

  sc_start(800,SC_NS);
  sca_util::sca_close_tabular_trace_file(tf1);
  sca_util::sca_close_tabular_trace_file(tf2);
  return 0;
}
```

B

Modified Nodal Analysis library for Maxima language

Listing B.1: Documentation of Modified Nodal Analysis library for Maxima language

```
init();
/*
Returns a couple [A,n] containing an empty matrix "A" and an empty list "n".
A is dedicated to contain the matrix of admittance.
n is dedicated to contain the list of names representing the potential voltages.
There order corresponds to the matrix columns.
*/
addVCCS(A,y,n0,n1,v0,v1);
/*
Updates the matrix A with the four elements corresponding to a voltage v0-v1
controlled current source accross n0 to n1 called y.
*/
addAdmittance(A,y,n0,n1);
/*
Updates the matrix A with the four elements corresponding to an addmittance
connected between n0 and n1 called y.
*/
addVeryVerySimpleTransistor(A,nD,nG,nS,nB,i);
/*
Updates the matrix "A" with the elements corresponding to a transistor small-signal model.
The model contains gm, gds, Cgd elements.
The transistor drain, gate, source, bulk are connected to "nD", "nG", "nS", "nB".
The elements are singular in the global matrix as an index is specified by "i".
*/
addVerySimpleTransistor(A,nD,nG,nS,nB,i);
/*
Updates the matrix "A" with the elements corresponding to a transistor small-signal model.
The model contains gm, gmb, gds, Cgd, Cgs elements.
The transistor drain, gate, source, bulk are connected to "nD", "nG", "nS", "nB".
The elements are singular in the global matrix as an index is specified by "i".
```

```
*/
addSimpleTransistor(A,nD,nG,nS,nB,i);
/*
Updates the matrix "A" with the elements corresponding to a level 1 transistor small-signal model.
The model contains gm, gmb, gds, Cgd, Cbd, Cgs, Cgb, Cbs elements.
The transistor drain, gate, source, bulk are connected to "nD", "nG", "nS", "nB".
The elements are singular in the global matrix as an index is specified by "i".
*/
addTransistor(A,nD,nG,nS,nB,i);
/*
Updates the matrix "A" with the elements corresponding to a BSIM3v3 transistor small-signal model.
The model contains gm, gmb, gds, Csd, Cgd, Cbd, Cgs, Cgb, Cbs, Cm, Cmb, Cmx elements.
The transistor drain, gate, source, bulk are connected to "nD", "nG", "nS", "nB".
The elements are singular in the global matrix as an index is specified by "i".
*/
addRFTransistor(A,nD,nG,nS,nB,i,j);
/*
Updates the matrix "A" with the elements corresponding to a complete BSIM3v3 transistor
small-signal model upgraded for RF.
The model contains gm, gmb, gds, Csd, Cgd, Cbd, Cgs, Cgb, Cbs, Cm, Cmb, Cmx, Rg, Rd, Rs,
Rsti, Cdsmet, Cbdmet, Cbsmet elements.
The transistor drain, gate, source, bulk are connected to "nD", "nG", "nS", "nB".
The elements are singular in the global matrix as an index is specified by "i".
The internal nodes are indexed by "j".
In case of differential topology, if the small-signal elements are supposed identical,
user can choose to let the same "i" index but the index "j" of internal nodes are different.
*/
addInductanceLVI(A,nPLUS,nMINUS,nSUB,i,j);
/*
Updates the matrix "A" with the elements corresponding to a LVI inductor model.
The model contains Rs1, Rs2, Rpatt1, Ls1, Ls2, Cox1.
The external connexions are "nPLUS", "nMINUS", "nSUB".
"i" indexes the small-signal elements.
"j" indexes the internal node names.
*/
addInductanceIDNW(A,nPLUS,nMINUS,nSUB,nMP,i,j);
/*
Updates the matrix "A" with the elements corresponding to a IDNW differential inductor model.
The model contains Rs1, Rs2, Rpat1, Rpat2, Rpatm, Rmp, Ls1, Ls2, Cox1, Cox2, Coxm.
The external connexions are "nPLUS", "nMINUS", "nSUB", "nMP".
"i" indexes the small-signal elements.
"j" indexes the internal node names.
*/
addInductanceISLA(A,nPLUS,nMINUS,nSUB,i,j);
```

```
/*
Updates the matrix "A" with the elements corresponding to a ISLA low area inductor model.
The model contains Rs1, Rs2, Rpat1, Rpat2, Rpatm, Ls1, Ls2, Cox1, Cox2, Coxm.
The external connexions are "nPLUS", "nMINUS", "nSUB".
"i" indexes the small-signal elements.
"j" indexes the internal node names.
*/
shortCircuit(A,n0,n1);
/*
Updates the matrix by short-circuiting the node n0 and n1.
The resulting node is called n0.
*/
differentialVoltage(A,v0,v1,sup);
/*
Updates the matrix by declaring voltage potential "v1" as the differential
potential voltage of "v0".
*/
setVin(A,pni,nni,Vin);
/*
Updates the matrix by declaring an independent input voltage source.
*/
setIin(A,nni,pni,Iin);
/*
Updates the matrix by declaring an independent input current source.
*/
getVout(A,pno,nno,comp);
/*
The voltage between "pno" and "nno" in the matrix "A" is returned
as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
*/
getIout(A,pno,nno,comp);
/*
The current that flows through "pno" to "nno" in the matrix "A" is returned
as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
*/
getVNoise(A,no,ni,nss,comp);
/*
The total noise computed as a voltage brought between "no" and "nss" in the matrix "A" is
returned as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
Nodes "ni" and "nss" are short-circuited.
*/
getINoise(A,no,ni,nss,comp);
/*
The total noise current that flows through "pno" to "nno"
```

```
    in the matrix "A" is returned as a symbolic expression if "comp" is true
120 and as an unsolved matrix if "comp" is false.
    Nodes "ni" and "nss" are short-circuited.
    */
    getOImpedance(A,no,ni,nss,comp);
    /*
125 The output impedance, in the matrix "A", is returned
    as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
    "no" is the output node, "ni" is the input node, "nss" is the ground.
    */
    getIImpedance(A,no,ni,nss,comp);
130 /*
    The input impedance, in the matrix "A", is returned
    as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
    "no" is the output node, "ni" is the input node, "nss" is the ground.
    */
135 getTransAdmittance(A,no,ni,nss,comp);
    /*
    The trans-admittance, in the matrix "A", is returned
    as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
    "no" is the output node, "ni" is the input node, "nss" is the ground.
140 */
    getIGain(A,no,ni,nss,comp);
    /*
    The current gain, in the matrix "A", is returned
    as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
145 "no" is the output node, "ni" is the input node, "nss" is the ground.
    */
    getVGain(A,no,ni,nss,comp);
    /*
    The voltage gain, in the matrix "A", is returned
150 as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
    "no" is the output node, "ni" is the input node, "nss" is the ground.
    */
    getVGain(A,pno,nno,pni,nni,comp);
    /*
155 The voltage gain, in the matrix "A", is returned
    as a symbolic expression if "comp" is true and as an unsolved matrix if "comp" is false.
    This version of getVGain allows to specify a input voltage reference different to
    the output reference.
    This is specially suited for nonlinearity evaluation.
160 */
```

References

Aboushady01a. H. Aboushady, Y. Dumonteix, M.-M. Louerat, and H. Mehrez. Efficient polyphase decomposition of comb decimation filters in Sigma-Delta analog-to-digital converters. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(10):898–903, 2001.

Aboushady01b. H. Aboushady and M.-M. Louerat. Low-power design of low-voltage current-mode integrators for continuous-time $\Sigma\Delta$ modulators. In *IEEE International Symposium on Circuits and Systems*, volume 1, pages 276–279. IEEE, 2001.

Aboushady02. H. Aboushady, F. Montaudon, F. Paillardet, and M.-M. Louerat. A 5mW, 100kHz bandwidth, current-mode continuous-time Sigma-Delta modulator with 84 dB dynamic range. *Proceedings of the Solid-State Circuits Conference*, pages 283 –286, 2002.

Aksin05. D. Aksin and F. Maloberti. Symbolic Small-Signal Analysis (SSA)Tool. In *IEEE International Symposium on Circuits and Systems*, pages 3007–3010. IEEE, 2005.

Ashenden02. P. J. Ashenden, G. D. Peterson, and D. A. Teegarden. *The System Designer's Guide to VHDL-AMS*. 2002.

Ashry11. A. Ashry and H. Aboushady. A 4th order subsampled RF Sigma-Delta ADC centered at 2.4GHz with a sine-shaped feedback DAC. In *Proceedings of the Solid-State Circuits Conference*, pages 263–266. IEEE, September 2011.

atm. 8-bit AVR Microcontroller with 128K bytes in-System programmable flash ATmega128 Datasheet. http://www.atmel.com/dyn/resources/prod%5C%255Fdocuments/doc2467.pdf.

Barnasconi10. M. Barnasconi. SystemC AMS Extensions: Solving the Need for Speed. *DAC Knowledge center*, 2010.

Beilleau09. N. Beilleau, H. Aboushady, F. Montaudon, and A. Cathelin. A 1.3V 26mW 3.2GS/s undersampled LC bandpass Sigma-Delta ADC for a SDR ISM-band receiver in 130nm CMOS. In *IEEE Radio Frequency Integrated Circuits Symposium*, pages 383–386. IEEE, June 2009.

Belfort09. Diomadson Belfort, Nicolas Beilleau, Hassan Aboushady, Marie-Minerve Louerat, and Sebastian Y. C. Catunda. A Q-enhanced LC bandpass filter using CAIRO+. In *International Conference on Electronics, Circuits and Systems*, pages 860–863, December 2009.

Beserra11. G.S. Beserra, J.E.G. de Medeiros, A.M Sampaio, and J.C. da Costa. System-Level Modeling of a Mixed-Signal System on Chip for Wireless Sensor Networks. *Design, Automation & Test in Europe Conference & Exhibition*, 2011.

Caluwaerts08. K. Caluwaerts, D. Galayko, and P. Basset. SystemC-AMS Heterogeneous Modeling of a Capacitive Harvester of Vibration Energy. In *IEEE Behavioral Modeling and Simulation Workshop*, pages 142–147. IEEE, September 2008.

Carloni02. L. P. Carloni, F. De Bernardinis, A. Sangiovanni-Vincentelli, and M. Sgroi. The Art and Science of Integrated Systems Design. *Proceedings of the European Solid-State Circuits Conference*, pages 25–36, 2002.

Cenni11a. F. Cenni, S. Scotti, and E. Simeu. Behavioral modeling of a CMOS video sensor platform using systemc AMS/TLM. *Forum on Specification and Design Languages*, 2011.

Cenni11b. F. Cenni, S. Scotti, and E. Simeu. SystemC AMS behavioral modeling of a CMOS video sensor. In *IEEE/IFIP International Conference on VLSI and System-on-Chip*, pages 380–385. IEEE, October 2011.

Chen05. Jesse E. Chen. Modeling RF Systems, 2005. http://www.designers-guide.org/Modeling/modeling-rf-systems.pdf.

Christen99. E. Christen and K. Bakalar. VHDL-AMS-a hardware description language for analog and mixed-signal applications. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 46(10):1263–1272, 1999.

cln. Cln - class library for numbers. http://www.ginac.de/CLN/.

Crombez07. P. Crombez, J. Craninckx, P. Wambacq, and M. Steyaert. Linearity guidelines for gm-C biquad filter design using architecture optimization with Volterra analysis. In *European Conference on Circuit Theory and Design*, volume 34, pages 216–219. IEEE, August 2007.

Daems03. W. Daems, G.G.E. Gielen, and W. Sansen. Simulation-based generation of posynomial performance models for the sizing of analog integrated circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(5):517–534, May 2003.

Degrauwe87. M.G.R. Degrauwe, O. Nys, E. Dijkstra, J. Rijmenants, S. Bitz, B.L.A.G. Goffart, E.A. Vittoz, S. Cserveny, C. Meixenberger, G. van der Stappen, and H.J. Oguey. IDAC: an interactive design tool for analog CMOS circuits. *IEEE Journal of Solid-State Circuits*, 22(6):1106–1116, December 1987.

del Mar Hershenson98. M. del Mar Hershenson, S.P. Boyd, and T.H. Lee. GPCAD: A Tool for CMOS Op-Amp Synthesis. In *Proceedings of the International Conference on Computer-aided design*, pages 296–303. ACM Press, 1998.

Einwich05. K. Einwich. Application of SystemC/SystemC-AMS for the Specification of a Complex Wired Telecommunication System. *Forum on Specification and Design Languages*, 2005.

El-Turky89. F. El-Turky and E.E. Perry. BLADES: an artificial intelligence approach to analog circuit design. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(6):680–692, June 1989.

Fernandez91. F.V. Fernandez, A. Rodriguez-Vazquez, and J.L. Huertas. Interactive AC Modeling and Characterization of Analog Circuits via Symbolic Analysis. *Analog Integrated Circuits and Signal Processing*, 1(3), 1991.

Ferrari99. A. Ferrari and A. Sangiovanni-Vincentelli. System design: traditional concepts and new paradigms. In *Proceedings IEEE International Conference on Computer Design: VLSI in Computers and Processors*, pages 2–12, 1999.

Frey00. P. Frey and D. O'Riordan. Verilog-AMS: Mixed-signal simulation and cross domain connect modules. In *Proceedings IEEE/ACM International Workshop on Behavioral Modeling and Simulation*, pages 103–108. IEEE Comput. Soc, 2000.

Fummi08. F Fummi, D Quaglia, and F Stefanni. A SystemC-based framework for modeling and simulation of networked embedded systems. In *Forum on Specification, Verification and Design Languages*, pages 49–54. IEEE, September 2008.

Gielen89. G.G.E. Gielen, H.C.C. Walscharts, and W.M.C. Sansen. ISAAC: a symbolic simulator for analog integrated circuits. *IEEE Journal of Solid-State Circuits*, 24(6):1587–1597, 1989.

Gielen94. G.G.E. Gielen, P. Wambacq, and W. Sansen. Symbolic analysis methods and applications for analog circuits: a tutorial overview. *Proceedings of the IEEE*, 82(2):287–304, 1994.

Gielen00. G.G.E. Gielen and R.A. Rutenbar. Computer-aided design of analog and mixed-signal integrated circuits. *Proceedings of the IEEE*, 88(12):1825–1854, 2000.

Gielen07. G.G.E. Gielen, T. Eeckelaert, E. Martens, and T. McConaghy. Automated synthesis of complex analog circuits. In *European Conference on Circuit Theory and Design*, pages 20–23. IEEE, August 2007.

gin. Ginac is not a cas. http://www.ginac.de/.

Habib10. A. Habib, F. Pecheux, and M.-M. Louerat. SystemC-AMS modeling of a PCR-CE lab-on-chip for multithreaded DNA analysis. In *2010 International Conference on Microelectronics*, number Icm, pages 483–486. IEEE, December 2010.

Hachtel75. G.D. Hachtel, M.R. Lightner, and H.J. Kelly. Application of the optimization program AOP to the design of memory circuits. *IEEE Transactions on Circuits and Systems*, 22(6):496–503, June 1975.

Hachtel80. G.D. Hachtel, T. Scott, and R. Zug. An Interactive Linear Programming Approach to Model Parameter Fitting and Worst Case Circuit Design. *IEEE Transactions on Circuits and Systems*, 27(10):871 – 881, 1980.

Hafez09. A.A. Hafez, M.A. Dessouky, and H.F. Ragai. Design of a low-power ZigBee receiver front-end for wireless sensors. *Microelectronics Journal*, 40(11):1561–1568, November 2009.

Haghighitalab10. D. Haghighitalab, M. Vasilevski, and H. Aboushady. LNA automatic synthesis and characterization for accurate RF system-level simulation. In *IEEE International Midwest Symposium on Circuits and Systems*, pages 938–941. IEEE, August 2010.

Harjani89. R. Harjani, R.A. Rutenbar, and L.R. Carley. OASYS: a framework for analog circuit synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 8(12):1247–1266, 1989.

Harvey92. J.P. Harvey, M.I. Elmasry, and B. Leung. STAIC: an interactive framework for synthesizing CMOS and BiCMOS analog circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(11):1402–1417, 1992.

Ho75. Chung-Wen Ho, A.E. Ruehli, and P.A. Brennan. The Modified Nodal Approach to Network Analysis. *Transactions on Circuits and Systems*, 22(6):504–509, June 1975.

Hormann11. L.B. Hormann, P.M. Glatz, C. Steger, and R. Weiss. A SystemC-AMS Simulation Environment for the Evaluation of Energy Harvesting Wireless Sensor Networks. *International Symposium on Performance Evaluation of Computer & Telecommunication Systems*, pages 247–252, 2011.

Iskander07. R. Iskander, D. Galayko, M.-M. Louerat, and A. Kaiser. Knowledge-aware synthesis using hierarchical graph-based sizing and biasing. *IEEE Northeast Workshop on Circuits and Systems*, pages 984–987, August 2007.

Javid09. F. Javid, R. Iskander, and M.-M. Louerat. Simulation-based hierarchical sizing and biasing of analog firm IPs. In *2009 IEEE Behavioral Modeling and Simulation Workshop*, pages 43–48. IEEE, September 2009.

Koh90. H.Y. Koh, C.H. Sequin, and P.R. Gray. OPASYN: a compiler for CMOS operational amplifiers. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 9(2):113–125, 1990.

Krasnicki99. M.J. Krasnicki, R. Phelps, R.A. Rutenbar, and L.R. Carley. MAELSTROM: efficient simulation-based synthesis for custom analog cells. In *Proceedings of Design Automation Conference*, pages 945–950. IEEE, 1999.

Kundert99. K.S. Kundert. Introduction to RF simulation and its application. *IEEE Journal of Solid-State Circuits*, 34(9):1298–1319, 1999.

Leenaerts01. D. Leenaerts, J. Van der Tang, and C. Vaucher. *Circuit design for RF transceivers*. 2001.

Leveque10. A. Leveque, F. Pecheux, M.-M. Louerat, H. Aboushady, and M. Vasilevski. SystemC-AMS Models for Low-Power Heterogeneous Designs: Ap- plication to a WSN for the Detection of Seismic Perturbations. *International Conference on Architecture of Computing Systems*, page 6, 2010.

Maehne09. T. Maehne and A. Vachoux. Supporting dimensional analysis in SystemC-AMS. In *IEEE Behavioral Modeling and Simulation Workshop*, pages 108–113. IEEE, September 2009.

Mahne11. T. Mahne. *Efficient Modelling and Simulation Methodology for the Design of Heterogeneous Mixed-Signal Systems on Chip*. PhD thesis, 2011.

Markert06. E. Markert, M. Dienel, G. Herrmann, D. Mueller, and U. Heinkel. Modeling of a new 2D Acceleration Sensor Array using SystemC-AMS. *Journal of Physics: Conference Series*, 34:253–257, April 2006.

Martens05. E. Martens and G.G.E. Gielen. Behavioral modeling and simulation of weakly nonlinear sampled-data systems. In *IEEE International Symposium on Circuits and Systems*, pages 2247–2250. IEEE, 2005.

Massouri10. A. Massouri, A. Leveque, L. Clavier, M. Vasilevski, A. Kaiser, and M.-M. Louerat. Baseband Fading Channel Simulator for Inter-Vehicle Communication using SystemC-AMS. In *IEEE International Behavioral Modeling and Simulation Conference*, 2010.

max. Maxima, a Computer Algebra System. http://maxima.sourceforge.net.

Medeiro94. F. Medeiro, F.V. Fernandez, R. Dominguez-Castro, and A. Rodriguez-Vazquez. A Statistical Optimization-based Approach For Automated Sizing Of Analog Cells. In *IEEE/ACM International Conference on Computer-Aided Design*, pages 594–597. IEEE, 1994.

Nagel71. L.W. Nagel and R. Rohrer. Computer Analysis of Nonlinear Circuits, Excluding Radiation. *IEEE Journal of Solid-State Circuits*, 6(4):166–182, 1971.

Nagel75. L.W. Nagel. SPICE2: A Computer Program to Simulate Semiconductor Circuits. 1975.

Normark04. E. Normark, L. Yang, C. Wakayama, P. Nikitin, and R. Shi. VHDL-AMS behavioral modeling and simulation of a $\Pi/4$ DQPSK transceiver system. In *IEEE International Conference on Cluster Computing*, pages 119–124. IEEE, 2004.

Nuzzo05. P. Nuzzo, F. De Bernardinis, P. Terreni, and A. Sangiovanni-Vincentelli. Enriching an Analog Platform for Analog-to-Digital Converter Design. In *IEEE International Symposium on Circuits and Systems*, pages 1286–1289, 2005.

Nye88. W. Nye, D.C. Riley, A. Sangiovanni-Vincentelli, and A.L. Tits. DELIGHT. SPICE: an optimization based system for the design of integrated circuits. *IEEE Transactions on Computer-Aided Design*, 7(4):501–519, September 1988.

Ochotta96. E. S. Ochotta, R. A. Rutenbar, and L. R. Carley. Synthesis of High-Performance Analog Circuits in ASTRX/OBLX. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 15(3):273–294, 1996.

Onodera90. H. Onodera, H. Kanbara, and K. Tamaru. Operational-amplifier compilation with performance optimization. *IEEE Journal of Solid-State Circuits*, 25(2):466–473, April 1990.

Pecheux05. F. Pecheux, C. Lallement, and A. Vachoux. VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(2):204–225, February 2005.

Pena07. E. Pena, E. de la Torre, A. de Castro, and T. Riesgo. A digital system to emulate wireless networks. *IET Computers & Digital Techniques*, 1(5):444, 2007.

Phelps00. R. Phelps, M.J. Krasnicki, R.A. Rutenbar, L.R. Carley, and J.R. Hellums. Anaconda: Simulation-Based Synthesis of Analog Circuits Via Stochastic Pattern Search. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 19(6):703–717, June 2000.

Rabaey06. J. M. Rabaey, F. De Bernardinis, A. M. Niknejad, B. Nikolic, and A. Sangiovanni-Vincentelli. Embedding Mixed-Signal Design in Systems-on-Chip. *Proceedings of the IEEE*, 94(6):1070–1088, June 2006.

Ravatin02. J. Ravatin, J. Oudinot, S. Scotti, A. Le-clercq, and J. Lebrun. Full transceiver circuit simulation using VHDL-AMS. *Microwave Engineering*, (May):29–33, 2002.

Rutenbar07. R.A. Rutenbar, G.G.E. Gielen, and J. Roychowdhury. Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs. *Proceedings of the IEEE*, 95(3):640–669, March 2007.

Sangiovanni-Vincentelli04. A. Sangiovanni-Vincentelli, L. Carloni, F. De Bernardinis, and M. Sgroi. Benefits and challenges for platform-based design. In *Proceedings of Design Automation Conference*, page 409, 2004.

Sangiovanni-Vincentelli07. A. Sangiovanni-Vincentelli. Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design. *Proceedings of the IEEE*, 95(3):467–506, March 2007.

Smith96. S.L. Smith and E. Sanchez-Sinencio. Low voltage integrators for high-frequency CMOS filters using current mode techniques. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 43(1):39–48, 1996.

soc. Soclib. http://www.soclib.fr/trac/dev.

Stefanovic05. D. Stefanovic, M. Kayal, and M. Pastre. PAD: A New Interactive Knowledge-Based Analog Design Approach. *Analog Integrated Circuits and Signal Processing*, 42(3):291–299, March 2005.

Swings91. K. Swings and W. Sansen. DONALD: a workbench for interactive design space exploration and sizing of analog circuits. In *Proceedings of the European Conference on Design Automation.*, pages 475–479. IEEE Comput. Soc. Press, 1991.

sysa. SystemC-AMS. http://www.systemc-ams.org/.

sysb. SystemC AMS 1.0. http://www.accellera.org/downloads/standards/systemc.

sysc. SystemC AMS publications. http://www.systemc-ams.org/publications.php.

sysd. SystemC Members. http://www.systemc.org/about/gallery/.

Tlelo-Cuautle07. E. Tlelo-Cuautle, C. Sanchez-Lopez, M. Fakhfakh, and M. Loulou. Designing SRCOs by symbolic-behavioral-modeling of unity-gain cells. *IEEE International Conference on Electronics, Circuits and Systems*, pages 1035–1038, December 2007.

Tulunay06. G. Tulunay and S. Balkir. Automatic synthesis of CMOS RF front-ends. In *IEEE International Symposium on Circuits and Systems*, page 4. IEEE, 2006.

Tulunay08. G. Tulunay and S. Balkir. Synthesis of RF CMOS Low Noise Amplifiers. In *IEEE International Symposium on Circuits and Systems*, pages 880–883. IEEE, May 2008.

Vachoux97. A. Vachoux, J.-M. Bergé, O. Levia, and J. Rouillard. *Analog and Mixed-Signal Hardware Description Languages*. Kluwer Academic Publishers, 1997.

Vachoux04. a. Vachoux, C. Grimm, and K. Einwich. Towards Analog and Mixed-Signal SOC Design with SystemC-AMS. *IEEE International Workshop on Electronic Design, Test and Applications*, pages 97–97, 2004.

Vasilevski07a. M. Vasilevski, H. Aboushady, F. Pecheux, and L. de Lamarre. Modeling Wireless Sensor Network nodes using SystemC-AMS. In *Internatonal Conference on Microelectronics*, pages 53–56. IEEE, December 2007.

Vasilevski07b. M. Vasilevski, F. Pecheux, H. Aboushady, and L. de Lamarre. Modeling heterogeneous systems using SystemC-AMS case study: A Wireless Sensor Network Node. In *IEEE International Behavioral Modeling and Simulation Workshop*, pages 11–16. IEEE, September 2007.

Vasilevski08a. M. Vasilevski, N. Beilleau, H. Aboushady, and F. Pecheux. Efficient and refined modeling of wireless sensor network nodes using SystemC-AMS. In *Ph.D. Research in Microelectronics and Electronics*, pages 81–84. IEEE, June 2008.

Vasilevski08b. M. Vasilevski, F. Pecheux, N. Beilleau, H. Aboushady, and K. Einwich. Modeling and Refining Heterogeneous Systems With SystemC-AMS: Application to WSN. *Design, Automation and Test in Europe*, pages 134–139, March 2008.

Vasilevski09. M. Vasilevski, H. Aboushady, and M.-M. Louerat. Automatic model refinement of GmC integrators for high-level simulation of continuous-time Sigma-Delta modulators. In *2009 IEEE International Symposium on Circuits and Systems*, pages 2769–2772. IEEE, May 2009.

Volterra59. V. Volterra. *Theory of Functional and of Integral and Integro-Differential Equations*. Dover Publications, 1959.

Wambacq95. P. Wambacq, F.V. Fernandez, G. Gielen, W.M.C. Sansen, and A. Rodriguez-Vazquez. Efficient symbolic computation of approximated small-signal characteristics of analog integrated circuits. *IEEE Journal of Solid-State Circuits*, 30(3):327–330, March 1995.

Yee01. D. G.-W. Yee. *A design methodology for highly-integrated low-power receivers for wireless communications*. PhD thesis, 2001.

Youssef11. S. Youssef, F. Javid, D. Dupuis, R. Iskander, and M.-M. Louerat. A Python-Based Layout-Aware Analog Design Methodology For Nanometric Technologies. In *Design and Test Workshop*, 2011.

Yu96. Q. Yu and C. Sechen. A unified approach to the approximate symbolic analysis of large analog integrated circuits. *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications*, 43(8):656–669, 1996.

Zele96. R. H. Zele and D. J. Allstot. Low-power CMOS continuous-time filters. *IEEE Journal of Solid-State Circuits*, 31(2):157–168, 1996.

Zhou11. M. Zhou and R. van Leuken. Systemc-AMS model of a dynamic large-scale satellite-based AIS-like network. *Specification and Design Languages*, 2011.