# A Hierarchical Modeling and Virtual Prototyping Methodology for Functional Verification of RF Mixed-Signal SoCs

Von der Fakultät für Elektrotechnik und Informationstechnik
der Rheinisch-Westfälischen Technischen Hochschule Aachen
zur Erlangung des akademischen Grades eines Doktors der
Ingenieurwissenschaften genehmigte Dissertation

vorgelegt von

Diplom-Ingenieur

Yifan Wang

aus Wuhan

Berichter: Universitätsprofessor Dr.-Ing. Stefan Heinen
Universitätsprofessor Dr.-Ing. Gerd Ascheid

Tag der mündlichen Prüfung: 22.11.2013

**Bibliografische Information der Deutschen Nationalbibliothek**
Die Deutsche Nationalbibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über
**http://dnb.d-nb.de** abrufbar.

ISBN 978-3-8439-1837-4

D 82 (Diss. RWTH Aachen University, 2013)


Sternstr. 18, 80538 München
Tel.: 089/66060798
www.dr.hut-verlag.de



1. Auflage 2014

*To my parents*

# Acknowledgment

First and foremost, I would like to express my gratitude to Prof. Dr.-Ing. Stefan Heinen, my research supervisor, for providing me the opportunity to work on this exciting research topic and sharing his rich experience in RF system design and simulator theories with me. While allowing me considerable freedom to conduct research on my own, his valuable suggestions and guidance helped me to steer this work in the right direction.

I would also like to extend my grateful thanks to Dr.-Ing. Ralf Wunderlich, for the good advices and the great organizational support during my time at IAS.

Furthermore, I would like to express my appreciation to all the colleagues and students who helped me to test, enhance and carry out the proposed verification flow on the tapeout projects: Dirk Bormann for the first class EDA-tooling support, Niklas Zimmermann for leading the RF-DAC top-level design and tapeout, Lei Liao for leading the Low Power Bluetooth Transceiver top-level design, Martin Schleyer for his great contribution to the functional verification of the BT transceiver tapeout.

In addition, I would like to thank all colleagues and students at IAS for the good working atmosphere, in particular my office mates Junqing Guan, Ahmed Farouk Aref, Stefan Kählert, and Zhimiao Chen, for the great team work and countless fruitful technical discussions.

I would also like to thank Aytac Attac and Tobias D. Werth for proof reading parts of this work.

Last but not least, I am indebted to my family and friends for their support, encouragement, understanding, and patience through all these years of my study.

# Contents

# List of Figures

# List of Tables

# List of Abbreviations

| | |
|---|---|
| 3GPP | 3rd Generation Partnership Project |
| ADC | Analog-Digital Converter |
| AMS | Analog and Mixed-Signal |
| BDD | Binary Decision Diagram |
| BER | Bit Error Rate |
| CAD | Computer Aided Design |
| CDF | Component Description Format |
| CM | Connect Module |
| CMOS | Complementary Metal Oxide Semiconductor |
| CP | Charge Pump |
| DAC | Digital-to-Analog Converter |
| DAE | Differential Algebraic Equation |
| DRC | Design Rules Check |
| DUV | Design Under Verification |
| EDA | Electronic Design Automation |
| ESL | Element Selection Logic |
| FFT | Fast Fourier Transformation |
| FHSS | Frequency Hopping Spread Spectrum |
| FPGA | Field Programmable Gate Array |
| FSM | Finite State Machine |
| GFSK | Gaussian Frequency Shift Keying |
| GSM | Global System for Mobile Communications |
| GUI | Graphical User Interface |
| HB | Harmonic Balance |
| HDL | Hardware Description Language |
| HED | Hierarchy Editor |
| IC | Integrated Circuit |
| IEEE | Institute of Electrical and Electronics Engineers |
| IF | Intermediate Frequency |
| IIP3 | Input-referred third-order Intercept Point |
| IIR | Infinite Impulse Response |

| | |
|---|---|
| IP | Intellectual Property |
| ISM | Industrial, Scientific and Medical |
| KCL | Kirchhoff's Current Law |
| KVL | Kirchhoff's Voltage Law |
| LFSR | Linear Feedback Shift Register |
| LNA | Low Noise Amplifier |
| LO | Local Oscillator |
| LPF | Loop Filter |
| LSB | Least Significant Bit |
| LTE | Long Term Evolution |
| LvS | Layout vs. Schematic |
| MCC | Modulation Compensation Circuit |
| MNA | Modified Nodal Analysis |
| MoC | Model of Computation |
| MODEM | Modulation Demodulation |
| MOSFET | Metal-Oxide-Semiconductor Field-Effect Transistor |
| MSB | Most Significant Bit |
| NF | Noise Figure |
| OCEAN | Open Command Environment for Analysis |
| ODE | Ordinary Differential Equation |
| OFDM | Orthogonal Frequency Division Multiplex |
| OSCI | Open SystemC Initiative |
| OSR | Oversampling Ratio |
| PA | Power Amplifier |
| PDF | Probability Density Function |
| PFD | Phase Frequency Detector |
| PLL | Phase-Locked Loop |
| PRBS | Pseudo Random Bit Stream |
| PSL | Property Specification Language |
| PSS | Periodic Steady State |
| QAM | Quadrature Amplitude Modulation |
| RF | Radio Frequency |
| RFIC | Radio Frequency Integrated Circuit |
| RTL | Register Transfer Level |
| RVM | Real Valued Modeling |
| SNR | Signal to Noise Ratio |
| SoC | System on Chip |
| SPICE | Simulation Program with Integrated Circuit Emphasis |
| SVA | SystemVerilog Assertions |
| UVM | Universal Verification Methodology |
| VCO | Voltage Controlled Oscillator |
| VCVS | Voltage-Controlled Voltage Source |

| | |
|---|---|
| VHDL | VHSIC-HDL |
| VHSIC | Very-High-Speed Integrated Circuits |
| VP | Virtual Prototype |
| VPI | Verilog Procedural Interface |
| ZOH | Zero-Order Hold |

# 1

# Introduction

With almost 6 billion mobile cellular subscriptions world-wide in 2011 [ICT11], the impact of mobile wireless communication devices such as smartphones on our daily life has become more and more inevitable. Its commercial success has revolutionized the way of personal communication and information interchange through wireless freedom.

In general, the constant evolution of seamless wireless connectivity with increasing data rates and the constant development of low-cost, power efficient, fully integrated system solutions are the main reasons for their commercial success. On the one hand, the ultimate user experience, such as augmented reality, can only be achieved by simultaneously getting high volumes of data through different wireless communication channels simultaneously. On the other hand, the mass market demands reduced production costs. This mainly drives the trend towards higher integration and smaller die size. This can only be facilitated by exploiting the shrinking and integration potential of the key building elements inside every smartphones; namely, the integrated CMOS-based wireless SoCs including RF subsystems, at advanced technology nodes.

The RF subsystem plays a critical role by providing the interface between the end user and the data rate intensive communication within the whole system. Recent wireless SoCs incorporate GSM, UMTS with receiver diversity and GPS in their RF subsystems [HC+09]. Their future successor will come with an even higher wireless communication data rate, plus every imaginable wireless connectivity solution such as Bluetooth and WiFi in a single chip. To put it briefly, the only way to enhance the feature and flexibility of the smartphone while keeping them economically attractive is to cover a broad range of wireless

communication standards and eliminate off-chip components. This vision can be achieved by developing advanced nano-scale CMOS RF subsystem concepts such as a reconfigurable RFDAC-based multi standard transmitter [Zim11]; SAW-less receiver front-end with interference cancellation technique [Wer11]; and substantially enriching the functionalities of each RF front-end by incorporating digital control, calibration, and compensation algorithms, which tightly interact with the analog/RF front-ends. Following this development trend, more and more digital functionalities are merging into the former analog-only RF circuit blocks. Hence, functional complexity of today's wireless SoCs, especially in the analog/RF subsystems, is increasing drastically, while the technology nodes continue to shrink. Consequently, the rising functional complexity inside the RF subsystems make them prone to functional errors in the design. Unlike performance issues, functional errors considerably decrease the post silicon test and evaluation possibility of the chip and its software. Serious functional errors can even make a chip fail to start up! Such errors result in re-spins, which could be a bitter loss for the industry, considering the time and money spent on the design and fabrication. Recent studies [Meh10] have shown that in current mixed-signal SoC design flows the probability of producing a functional error due to design failures during the first tapeout lies around 45%. More than 50% of SoC design re-spins at 65 nm and below are due to functional errors, such as connectivity bugs, misinterpreted digital control bits, or wrongly inverted signals [Che09]. Hence, the overall system, including the analog/RF subsystems, demand rigorous functional verification prior to tapeout, which is one of the most critical and challenging tasks in the whole design flow [CK07] [KC09a].

Considering the two fundamental verification categories, both the formal and simulation-based verification methods are well established for pure digital designs [Mey04] [WGR05]. However, both of them are limited at the borderline of analog/RF blocks [KC09b] [HR04]. The reason lies in the basic nature of analog and digital signals; while digital signals only take a limited set of pre-defined logic values at discrete times, analog signals possess infinite states in a certain range at any point of time. Therefore, formal verification for analog/RF circuits is only applicable for small building blocks with a low number of elements. Otherwise, it has to face the state explosion problem [Pel09], which makes this method unfeasible for the large scale RF subsystem inside a SoC. The simulation-based method presupposes the description of the system behavior. The digital behavior is described by boolean relations and the analog by non-linear differential algebraic equation systems. A digital simulator has only to evaluate the logic relations at discrete time points, triggered by the change of logic states(event-driven). The analog simulator, such as SPICE [Nag75], has to solve a set of non-linear differential equations in an iterative fashion in order to meet the desired convergence criteria. Any slight change in signal or time will lead to a recalculation of the whole system. Hence, digital simulators possess

much higher simulation efficiency and coverage compared to the analog one. Simulation of a mixed-signal system requires not only the execution of both simulators at the same time, but also continuous synchronization with each other. Although the most straight forward approach for detecting functional bugs would be brute-force, simulating the whole design at the transistor-level using a mixed-signal simulator. The carrier frequencies in the RF subsystems which are usually decades higher than the bandwidth of the transferred data and therefore generate tremendous overhead for the simulator (s. Fig. 1.1). Plus, the convergence and small time-step issues due to the non-linearity, accompanied by the large transistor count of today's SoC, and the tough schedule due to the gradual shortening of time-to-market, make such simulation absolutely infeasible. With the increasing size and functional complexity of analog/RF subsystem

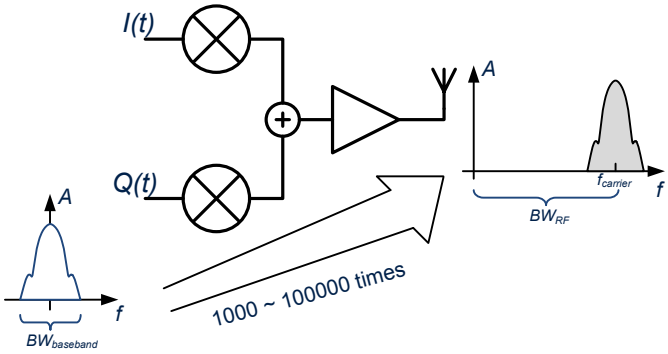



Figure 1.1: RF simulation problem: the RF signal bandwidth is often decades higher than the baseband data. Generating a large payload for the simulator.

designs, the analog simulator becomes the major bottleneck of the simulation-based functional verification!Over the last years, the EDA industry has made numbers of contributions through different ways to address this challenge. One way is the development of advanced SPICE-level and FAST-SPICE simulators through optimizing and parallelizing the solver in order to increase simulation efficiency of RF/analog subsystems [Cad] [Bda]. Still, advancements in these fields cannot keep up with the continuously growing size of the mixed-signal RF subsystem design.

Another contribution is the extension of the classical Hardware Description Languages (HDL) with mixed-signal capabilities, such as Verilog-AMS or

VHDL-AMS, and rising the level of abstraction by substituting the circuit at the transistor-level with their pin-compatible behavioral models. This significantly increases the simulation time step and thus reduces the simulation time. Behavioral models for the analog/RF blocks are mostly implemented manually. They are only good as long as they are kept in sync with the design changes. Without automated tooling, model refinement will occupy a lot of manpower. Hence, behavioral models require not only modeling expertise, but also extensive tooling support through the whole design flow. Despite the mentioned drawbacks, simulating the whole design in a pure digital simulator including pin-compatible event-driven behavioral models of the analog/RF part is the only feasible way of applying functional verification for complex, mixed-signal RF SoCs for the time being, since any analog construct used in the models invokes the analog solver and therefore slows down the digital simulation by orders of magnitude! A hierarchical modeling and verification methodology with consideration of different event-driven analog modeling techniques, accompanied by automated model validation and refinement approaches will be addressed in this work. In addition to proving the functionality, since the hardware itself only serves as part of the system, the Virtual Prototype of the whole chip, which captures the latest snapshot of the hardware part, is also necessary to ensure correct HW/SW interaction. Modern C-based modeling languages such as SystemC can effectively address higher levels of abstraction while being compatible with software design. However, it lacks a direct link to the analog/RF design environment. The approach to create a consistent link between circuits and models at different levels of abstraction, while hierarchically verifying the functionality of the whole system will be investigated and elaborated in this work.

## 1.1 Objective of this work

The goal of this work is to develop a functional verification methodology that is suited for the modern SoC systems including analog/RF subsystems prior to tapeout. First, an overview of different verification methods, modeling languages, and approaches, including their advantages and disadvantages has to be provided. Based on the investigation the most suitable method is defined. It should consist of efficient modeling methods that enable fast simulation at different levels of design hierarchy and different phases of the design process. In addition to being fast, methods maintaining the consistency between models and circuits, including the pin-compatibility and design parameter parity, also have to be given. Once the flow is defined, automation concepts will also be worked out.

An automated hierarchical SystemC virtual RF prototyping method targeting model creation that is based on the circuit design database will also be implemented, which seamlessly links the system level and block/circuit level design

by generating a pin accurate, specification oriented virtual prototype with high simulation performance. Finally, the feasibility of the proposed modeling and verification methodology has to be exemplified by the results of industry-oriented projects.

## 1.2 Structure of this work

This work is organized as follows:

- Chapter 2 introduces and compares the basic methods of verification, which generally can be categorized into two categories: formal and simulation-based(non-formal). Moreover, the elements and requirements of simulation-based verification for mixed-signal RF subsystems, especially the fundamental differences between functional and performance verification, are explained in detail. The general prerequisites of hierarchical verification flow, focusing on system partitioning and behavioral modeling, are given.
- Chapter 3 provides an overview of different HDLs regarding their simulation performance, reachable levels of abstraction, and model creation effort. Different modeling techniques from the functional verification point of view are shown. The issues and limitations of application of these mentioned modeling techniques are given and evaluated.
- Chapter 4 describes the proposed hierarchical functional verification concept in detail. The focus of this concept is maintaining the consistency between models and circuits within the design environment. Therefore, several techniques, including design partitioning, hierarchical model implementation, assertion coding, as well as automated model parameter extraction and update, are worked out. This chapter concludes with the influence and known limitations of the mentioned verification concepts on existing projects.
- Chapter 5 describes a new SystemC-based RF virtual prototyping Method based on insights gained in chapter 3 and chapter 4. First, the basic ideas and requirements of the RF VP implementation are given. Subsequently, two key points of the VP methodology, namely the automated hierarchical pin-accurate SystemC model frame generation based on design database and a unified SystemC RF building block library are pointed out. Finally, the corresponding implementations of the mentioned points are shown in the same chapter.
- The main target of this work is to develop modeling and VP methodology that focuses on fast and reliable function verification for state-of-the-art RF SoC design. The best way to demonstrate the effectiveness of the flow

is to apply it to the current project and design environment. This has been done in chapter 6.

- Chapter 7 concludes the thesis with a brief summary of the findings and achievements from this work. Further research directions are given.

# 2

# Fundamentals of Verification Techniques

According to the IEEE Standard Glossary of Software Engineering Terminology [IEE90], verification is defined as "The process of evaluating a system or component to determine whether the product of a given development phase satisfy the conditions imposed at the start of that phase." In electronic design automation (EDA), verification is defined as a process used to demonstrate the functional correctness of a design [Ber03].

Generally, the verification methods can be classified into two categories: formal and simulation-based(non-formal) verification [Lam08]. The simulation-based methods require an simulation environment. By applying input stimuli (or input test vector) to the design under verification (DUV), the DUV is verified by comparing its outputs to the specified/expected outputs. Formal methods do not need a simulation environment, the main focus of formal verification is the attempt to prove with mathematical certainty the correctness of a design [WGR05].

The substantial difference between the two categories is the existence of input stimuli (also called test vector). Simulation based methods are purely input stimuli driven and explore some of the possible behaviors and cases of a circuit system. Formal methods concentrate only on the output properties of the design [Lam08] and conduct an exhaustive exploration of all possible behaviors based on the formal specification. Fig. 2.1 gives a very general overview of the two mentioned verification methods according to [WGR05], which will be described in detail in the following.

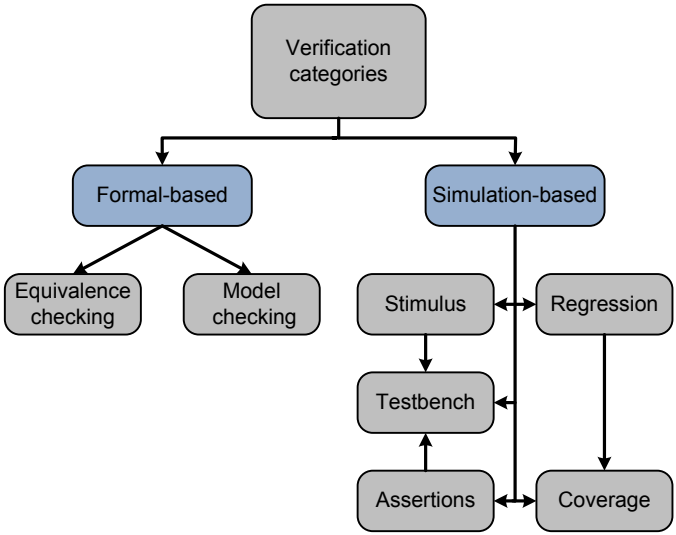



Figure 2.1: Overview of the two general verification categories.

## 2.1 Formal verification methods

The main idea of formal verification for analog and mixed-signal (AMS) circuits and systems is to use mathematical formulation to rigorously prove the correctness of the design based on specification [Dre04]. Formal verification methods are widely used in the hardware design industry mainly targeting digital domain [Har03], since this is the only way to completely detect potential errors in the integrated circuits, which often results in costly re-spins[1]. According to [Lam08], two techniques of formal verification can be distinguished further: equivalence checking and model checking[2].

### 2.1.1 Equivalence checking

Equivalence checking proves or disproves the complete functional consistency between implementation and specification, or equivalence of two circuit descriptions,

[1] The term RE-SPIN in the mixed-signal design means a complete redesign and reproduction of the silicon due to bug/s in the previous design

[2] Also called property checking in the literature. A property checking program is called model checker [Lam08]. The notion MODEL CHECKING can be regarded as the established academic term for property checking

e. g. the register transfer level (RTL) model and synthesized gate-level model in the digital design [MM04]. Fig. 2.2 illustrates the general flow of equivalence checking.

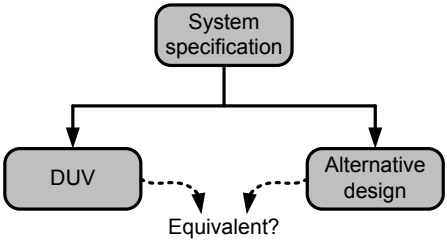



Figure 2.2: Overview of equivalence checking flow.

Equivalence checking is well established in the digital design due to the finite input and output state of digital circuits. Since the logic cones can be represented as Binary Decision Diagram (BDD) [Ake78], which has a canonical form, the equivalence of two digital designs can be verified by comparing if their respective canonical forms are identical. Additionally, the logical operations can be mapped to BDDs very efficiently with further order reduction possibilities [Bry86], which lead to the fact, that this method is widely acknowledged in the industry and widely applied to RTL and gate-level of the digital design. For equivalence checking of AMS circuits and systems, additional equivalence metrics are required, since there is no complete equivalence in the analog part of the circuits such as Boolean equivalence of digital systems [Ste11].

### 2.1.2 Model checking

Model checking [CGP00], illustrated in Fig. 2.3, is a verification technique that compare the functional properties with the models of circuits and systems. Therefore, the task of model checking requires the accurate modeling of the DUV and the formal description of the property to be verified. The models have to describe every possible system behavior of the DUV in a mathematically precise and unambiguous manner, so that all possible system states can be explored in a brute-force way [BKe08]. The result of the model checker, which is the software tool that performs the task of model checking, will report true if every state of the system model satisfies the property. Otherwise, it points out the property violations and provides debug information.

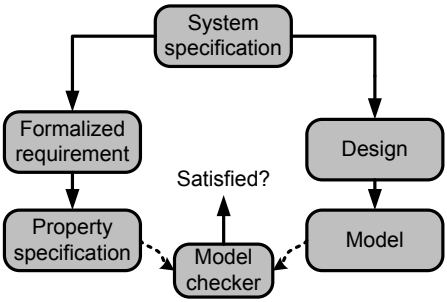



Figure 2.3: Overview of model checking flow.

For digital circuits, the model of the finite state system is described in state transition system (s. Fig. 2.4). The model checker verifies if the state transition system satisfies the given specification. This leads to the main issue of model

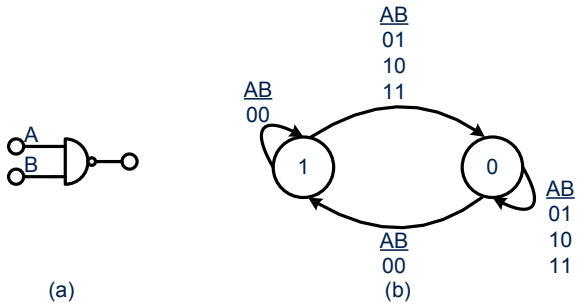



Figure 2.4: NAND gate (a) and the corresponding state transition graph (b).

checking technique: it is basically a reachability analysis, which suffers from the state-space explosion problem [Pel09]. Since the number of states, which has to be examined, increases exponentially with the number of state-space dimensions. The number of states required to accurately map the system can easily exceed the available memory and computational power. Especially in case of AMS systems, the state space of analog circuits can be regarded as infinite. One possibility is to map the transient response of an analog circuit under all possible input waveforms to an finite state machine (FSM) by numbers of repeated transient mixed-signal simulations, so that the continuous state space can be limited and discretized [DC05]. Still, the demand of exhaustive simulation in this case makes

the model checking process impracticable. Additionally, since model checking only verifies the model, but not the actual design, any verification result is therefore only depending on the quality of the model itself. Based on the two aforementioned facts, the model checking technique is applicable on pure digital systems with moderate size, but still not feasible for the verification of complex AMS systems.

## 2.2 Simulation-based verification methods

Although formal verification methods are the only way to prove the complete correctness of the system, it consists several drawbacks mentioned in section 2.1.1 and 2.1.2, the extensive usage of memory and long runtime for certain verification result makes them not feasible for the complexity and time-to-market requirement of today's RF mixed-signal System on Chips (SoC). Therefore, the simulation-based verification method still remains the most popular and widely established method in the industry [Ber03].

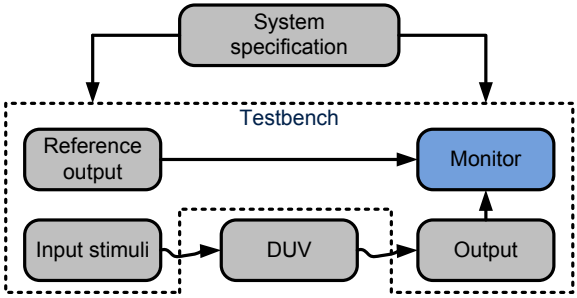



Figure 2.5: Overview of simulation-based verification flow.

As Fig. 2.5 illustrated, a typical simulation-based verification environment consists of four components:

- The **DUV**, which consists the implemented circuits based on system specification.
- The **input stimuli**, which consists the patterns and sets of the test vectors.
- The **reference**(or the expected) **output**, which has been derived from the system specification.
- The **monitor**, which is in charge of comparing the output of the DUV with the reference output.

Together, the input stimuli and monitor are considered as the testbench for the DUV [Ber03]. During the simulation, the set of input stimuli, which has been defined based on the specification, applies to the DUV, the output of the DUV will be checked against the reference output according to the specification. If any mismatch is found, further debug or design change process will be initiated by the verification engineer or the designer.

Normally, every design errors can be found with this approach, if the input of the DUV can cover all possible scenarios. However, this would lead to the demand of means of stimulus sets and simulations, which again makes the time required for the verification unreasonably long. Therefore, only a limited number of simulations will be executed by the designer. The designer has to choose the test cases based on their priority and usefulness according to the specification. E. g. the gain of a low noise amplifier(LNA) can only be determined correctly, if the input matching holds the specification. Additionally, the evaluation of simulation results based on the reference output, which is the task of the monitor shown in Fig. 2.5, is often a manual task. Especially in the analog verification environment, lots of simulation results have to be checked manually, or even post processed further by the designer, in order to determine the correctness of the DUV for certain test case. Moreover, the specification from the system level, e. g. signal to noise ratio (SNR), has to be converted into circuit level conforming specifications such as noise figure (NF) or noise voltage by the designer, which increases the possibility of misinterpreted specification and leads to wrong verification results.

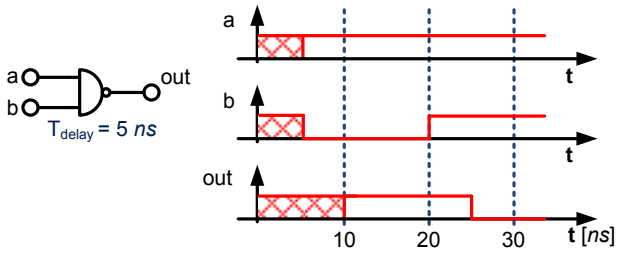
The human error made by manually examining the simulation results is also an unneglectable, but a daunting factor. While still practicable for the verification of circuit blocks with reasonable size, checking numbers of signals from different simulations of a RF front-end manually is clearly beyond the feasibility limit of a manual task. Thus, additional tools and methods are required to assist the simulation-based verification task. Before introducing the assistant tools, it is important to firstly distinguish available simulation algorithms for different systems, in order to highlight the limit and critical bottleneck of simulation-based verification methods for today's RF mixed-signal circuits and systems.

### 2.2.1 Simulation of digital circuits and systems

The digital system uses discrete values to represent signals and information. The signal level of a digital system contains two states, which can be considered as 0 and 1. Based on this binary property of the system, its behavior can be represented by Boolean algebra [Whi95], so that each part of the system can be striped down and mapped to the basic Boolean operators: conjunction $\wedge$, disjunction $\vee$ and negation $\neg$ [Boo54]. Thus, the digital systems can be implemented based on logic gate, which in general is the hardware representation

of the basic Boolean operators. In digital system, the output of the logic gate only changes if the input has been changed into a certain state (s. the state transition graph in section 2.1.2).

For the simulation-based verification of digital designs, the event-driven simulation is the broadly established simulation algorithm in the industry. During the simulation, every building blocks of a logic network are interconnected with each other [WGR05], serves as the netlist[3] for the digital simulator. The signals transport the simulation data from the outputs of the blocks to the adjacent, interconnected inputs of the gates. Each state change of the outputs could lead to the change of the internal states of the connected gates. This may lead to the transfer of new information through the whole logic network. Such transfer of information is regarded as event [WGR05]. During the simulation, if an event is generated, the information and its corresponding time will be put into an event list.




| sig. \ t | 0 | 5 | 10 | 20 | 25 | 30 |
|---|---|---|---|---|---|---|
| **a** | x | 1 | 1 | 1 | 1 | 1 |
| **b** | x | 0 | 0 | 1 | 1 | 1 |
| **out** | x | x | 1 | 1 | 0 | 0 |

| **Eventlist** | t=5, a, b=1<br>t=20, b=0 | t=10, out=1<br>t=20, b=1 | t=20, b=1 | t=25, out=0 | … | … |
|---|---|---|---|---|---|---|

Figure 2.6: Example of simulating a NAND logic gate.

The events traverse through the network, activating only blocks triggered by the signal flow. When the simulator reaches a certain time point, it will only evaluate the events occurred at that time point, all events occurred before that time point must have been already evaluated. Fig. 2.6 gives an Example of simulating a NAND gate. The general flow chart of the event-driven simulation algorithm has been shown in Fig. 2.7.

[3]The term *netlist* is defined as the input of a circuit simulator, consisting the structural description of the overall testbench units and DUV.

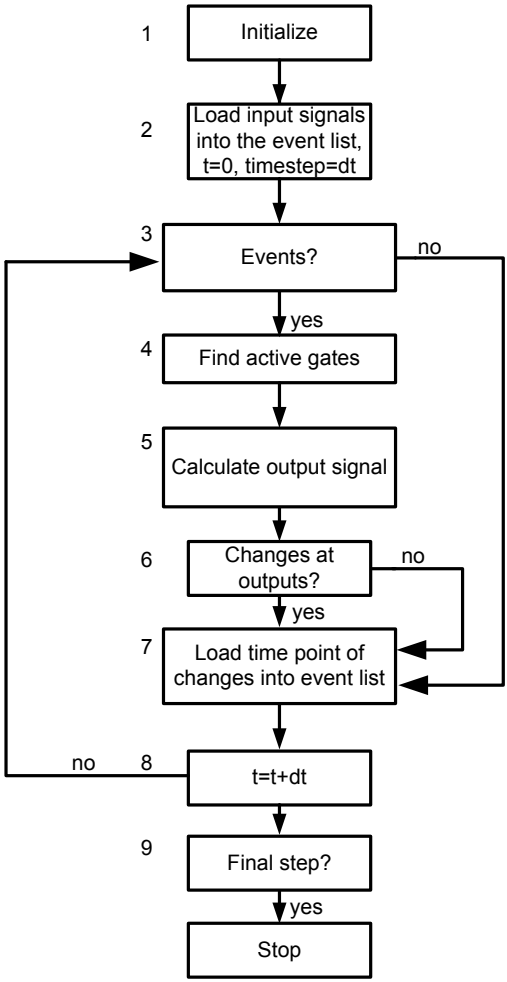



Figure 2.7: General algorithm for the transient simulation in a digital circuit simulator e. g. NCSIM.

### 2.2.2 Simulation of analog circuits and systems

Conservative systems are described using conservation laws at their connection points [FM98]. Such system always consists two components within each nods and signals: potential and flow. In electrical systems, the equivalent description for the potential and flow are voltage and current. The value of the mentioned components are continuous and any modification of the value can be characterized in the continuous time domain. This kind of system is defined as analog system. The relation between voltage and current can be described via Ohm's law [Ohm27], which states that the current through a conductor between two electrical nodes is directly proportional to the potential difference across the nodes.

Analog integrated circuits (ICs) consists linear (passive electronic device such as resistor, capacitor and inductor) and nonlinear (active device such as metal-oxide-semiconductor field-effect transistor: MOSFET) components. The overall system is defined by the network of the interconnected analog devices. The classical method of CAD (computer aided design) based analog circuit simulation is due to the fact that all connection-nodes are following the Kirchhoff's laws [Kir47]. Each active and passive device has its corresponding physical device model, described by ordinary differential equations (ODEs). The netlist of an analog system consists every interconnect-nodes of the devices, serving as the input for the analog simulator. The device model of the netlist elements forms the differential algebraic equation (DAE) system by obeying the Kirchhoff's voltage (KVL) and current(KCL) laws.

Modified Nodal Analysis (MNA) [HRB75] is one of the most used network analysis techniques by analog circuit analysis tools to set up the DAE system of the circuit. The nonlinear DAE system are then converted to a set of nonlinear difference equations by a multi step integration method [KG95] such as the backward Euler [JNT90] [BW03]. By having the DAE system representation of the circuit, nonlinear difference equations are solved simultaneously by using the numerical method based on the Newton-Raphson algorithm [Ypm95], which involves the conversion of non-linear equations into linear ones and their solutions by applying the sparse Gaussian elimination [4] to them [KG95] [TS94] [Nag75].

Fig. 2.8 illustrates a typical flow chart of a transient simulation by using the analog simulator. Here, the simulator generates the initial operating value of conservative components for all nodes in the circuit. At the second step, the equivalent linear model of the nonlinear device are generated, mapping the small signal behavior [TS94] of the device at its operating point. The solution of the DAE system can be found based on the inner loop (2-6), if every node of the system fulfills the error tolerance (convergence). The outer loop (2, 7-9) is

[4] Since the DAEs are presented in the matrix form, the LU factorization are applied to the nodal matrix.

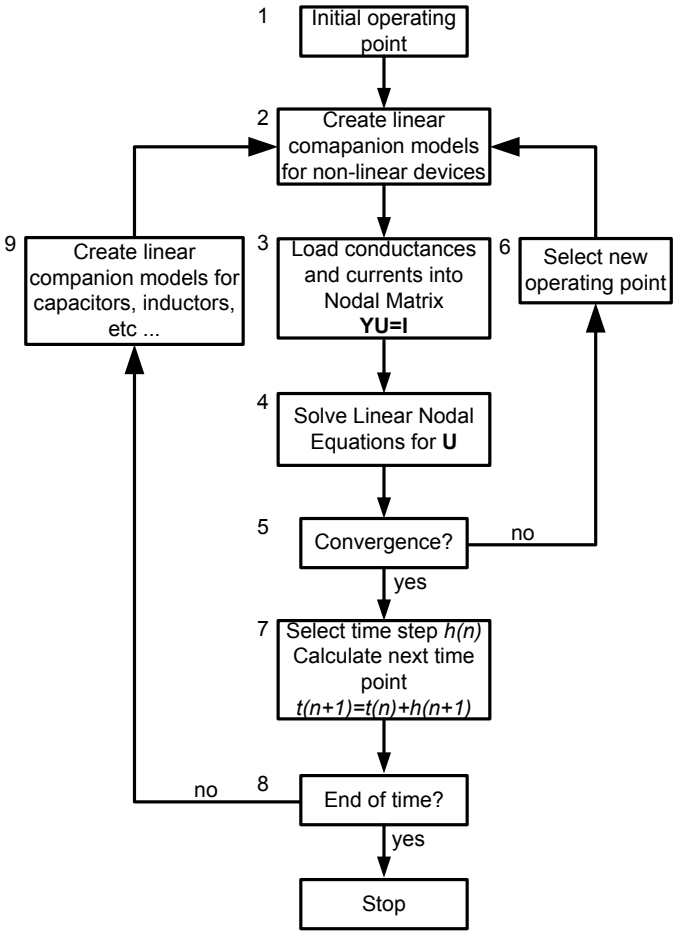



Figure 2.8: General algorithm for the transient simulation in an analog circuit simulator [HW10] e. g. SPICE [Nag75].

in charge of controlling the simulation time step, depending on the degree of the nonlinear system, the time step $h(n)$ will be adjusted dynamically by the simulator.

For highly nonlinear circuit system such as a switch-mode CMOS(Complementary Metal Oxide Semiconductor) RF power amplifier(PA), the step size of $h(n)$ depends not only on the nonlinearity but also strongly related to the maximal frequency in the system, resulting in very small time steps and slows down the overall simulation speed.

### 2.2.3 Simulation of mixed-signal systems

Systems containing both discrete event and continuous time parts are considered as mixed-signal system, where logic components closely interact with analog devices. Due to the different simulation algorithms for digital and analog systems (s. table 2.1), a mixed-signal system consisting analog and digital system in one silicon, has resulted in one of the main challenges for the simulation-based verification approach.

A simulator, which is compatible with both analog and digital netlist and their simulation domains, can be regarded as a mixed-mode simulator [AD+90] [New78]. As table 2.1 shows, numbers of substantial differences exist in both simulation domains. First, the analog system is conservative, the simulator has

| **Analog simulator** | **Digital simulator** |
|---|---|
| solves DAE system | calculates discrete events |
| solves implicit functions[a] | solves explicit functions |
| continuous time & value | discrete time & value[b] |
| solves whole DAE system iteratively | data flow simulation |
| **Simulation time step:** | |
| depends on the accuracy setup controlled by the simulator | predefined time scale & resolution already inherently specified |

[a] Explicit function is also possible, but will lead to stability issues according to [KG95].
[b] Continuous value, represented as 64-bit real number, can also exist in digital simulator.

Table 2.1: Comparison of the solver characteristics between analog and digital simulator.

to solve the DAE system based on the analog netlist in an iterative fashion, trying to meet the desired convergence criteria (accuracy). Each change in a single node of the system leads to the recalculation of the whole DAE system. In contrary, simulation of digital system requires only computing of the triggered

logic gates/networks, but not the whole system. This often leads to a speed up of 10 to 100 times compare to analog simulator [SN90]. Secondly, an electrical analog signal consists two components: voltage and current, in a continuous value representation. The digital signal contains only finite states (1, 0, X and Z). During the transient simulation, the time step of the analog simulator are depending on the accuracy setup of the simulator and the linearity of the circuit, while digital simulator only processes the existing events at each time point, which has been defined by the time scale and resolution.

Based on these facts, a mixed-mode simulator has to fulfill the following criteria, in order to overcome the basic issues resulting from the mentioned difference between the simulators:

- *Unified netlist:* A netlist serves as the input of the simulator, due to the different component representation and simulation algorithms of analog and digital systems, a mixed-mode simulator require an unified structural representation of the overall system. Fig. 2.9 shows a simple example, where the output of an digital inverter serves as the input of an analog inverter.

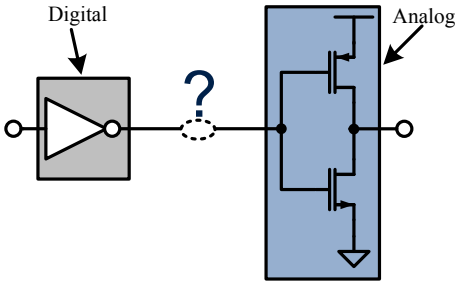



Figure 2.9: Simple mixed-signal system example.

Both analog and digital netlist are shown in Fig. 2.10. The mixed-signal netlist shown in the same figure contains following modification. First, it introduces the term *hierarchy* for the whole system. As one can observe, the analog netlist is a "flat" description of the components' interconnection, while the digital netlist consists both components' interconnection and their hierarchical structure. Second, the mixed-signal netlist introduces a more formalized language structure[5]. For the netlist shown in Fig. 2.10,

[5]The mixed-signal netlist shown in Fig. 2.10 uses the Verilog hardware description language format, which is the default netlist format for the mixed-signal design in Cadence design

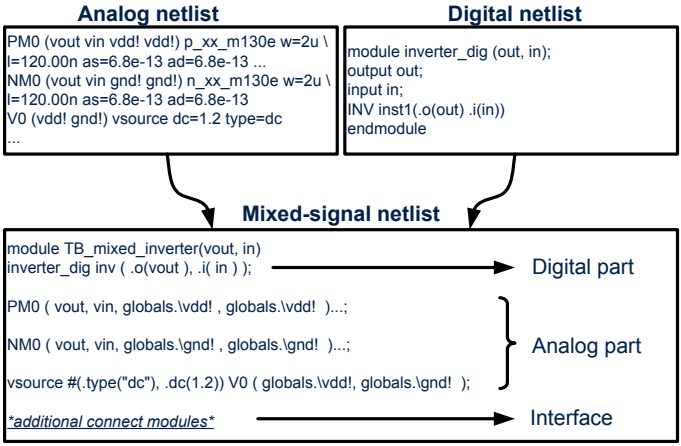



Figure 2.10: Mixed-signal netlist of the system shown in Fig. 2.9.

the analog devices are hierarchically embedded into a netlist format similar to the digital one. An more advanced feature of the mixed-signal netlist is, that it can also help the designer to describe different part of the system at different levels of abstraction.

- *Uniformed signal representation:* An uniform signal representation in both simulation domains is important for accurate simulation results [AD+90]. Analog simulator uses 64-bit double precision to present the value of voltage and currents, while digital simulator basically only using "0" and "1" to describe the logic signal. Thus a proper conversion between analog and logic signal has to be considered during the simulation. One possibility is to consistently use double precision for both signal forms according to [AD+90], [SN90] and [New78]. Other approaches such as converting the analog value into a 64-bit logic representation [Che09] or adding double precision support in the logic simulator [HC09] involve the application of HDL, which will be described in detail in chapter 3.
- *Seamlessly linking the simulation modes* (or interfacing the signal domains): When both analog and digital nodes are connected together, special interface elements have to be applied in order to convert digital states into analog/electrical values (voltage[6] or current) and vice versa. Basically,

tools, please refer the chapter3 for more detailed description.

[6]By default, digital states are converted to specific analog voltage levels, except for current

two cases have to be considered: electrical-to-logic and logic-to-electrical, the order of the names indicates the direction of signal conversion. Logic values are considered as more abstract compare to electrical value [JH06], a conversion from electrical to logic signal implies converting analog values to a higher abstraction level with less details, e. g. removing the current information. In contrary, converting from logic to electrical means a conversion from higher levels of abstraction to lower levels [SN90]. The major issue, which has to be addressed here, is that the logic states don't possess enough information to be handled in the analog simulator.

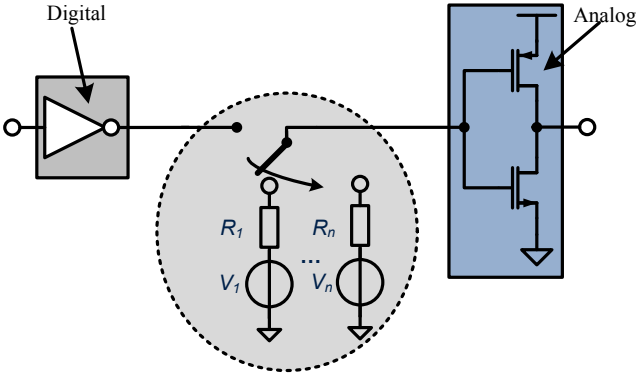



Figure 2.11: Details of a logic to electrical interface element.

Fig. 2.11 illustrates the interface element for the example shown in Fig. 2.9. The output of the logic gate is directly connected to the input of the analog inverter. In order to overcome the mentioned issue, the interface elements acts as an array of switchable voltage source, each alteration of the logic state leads to the switching to specific voltage source in the array. Higher number of the array elements in the interface will result in higher accuracy, but will also slow down the simulation [SN90]. Similar approaches such as the automatic insertion of HDL connect modules (CM) during the simulation [Mey03] will be discussed in detail in chapter 3.

- *Synchronization of simulation time:* During the simulation, time is expressed as a real number in analog simulator and as an integer multiple of the time resolution (minimum time unit) in digital simulator. Thus the mixed-modes simulator should be capable of handling both time repre-

mode logics.

sentations. One of the existing approaches is to unify the minimum time resolution for both simulator [New78]. Besides unifying the simulators'

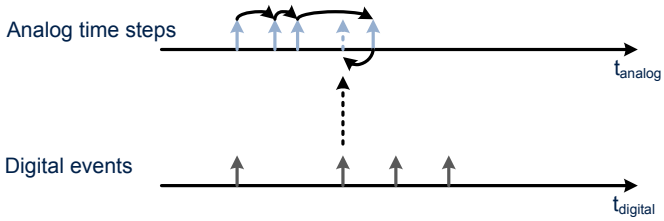



Figure 2.12: Time step synchronization between analog and digital solver during a mixed-mode simulation. Here, backtracking algorithm.

time resolution, there are essentially two methods for the synchronization of the simulation time steps [ZG+95]:

1.) The *backtracking* algorithm, illustrated in Fig. 2.12, which allows both simulators to progress in its own time step until its internal activity has ceased [ZG+95]. If e. g. a digital event has triggered the analog simulator before the end of the analog time interval, all results generated after the digital event are discarded. The analog simulator has to track back to the time point of the event and re-simulate with a smaller time step, in order to get accurate results. This technique has been used in [Vla91] and [AD+90].

2.) The *lockstep* algorithm, in which both simulator progress with same predefined time step. The analog simulator can reduce the size of the time step further, when convergence issues occur. If any digital events happens between two adjacent time points, the unified time steps will be refined.

Simulators exhibit the mentioned characteristics are capable of assisting the simulation-based verification approach for AMS circuits and systems.

### 2.2.4 Co-Simulation approaches

Besides simulating both analog and digital domain with an unified simulator, some tools also offers the support for the interaction with external simulators. Co-simulation require not only the proper synchronization between the analog and digital systems, but also additional data and time step alignment between the internal and external simulators, which leads to higher demand for computational power.

Most modern mixed-signal systems are designed with a mixture of hardware and software algorithm components. The specification and implementation of the algorithm can be done with system simulator or tools at higher abstraction level. Verification of the complete system require sometimes the interaction between hardware and the specified algorithm, especially for the case of RF MS SoC, where the algorithm, software and hardware are tightly meshed.

Typically, co-simulation is only feasible, if part of the system is extremely optimized and designed with the help of the external simulator. In that way, the overall simulation performance and the time savings compare to merging the design into the main simulator are still high, despite of all additional data and time step synchronization difficulties. Besides the mentioned case, the acquisition of external intellectual property (IP) may also lead to the application of co-simulation approach for the verification of the whole system. Part of existing co-simulation approaches are: MATLAB® simulink with Cadence® AMS-Designer [Ams], VERILOG/ VHDL-AMS with SystemC-AMS [ZGH10], SMASH with simulink [Sma] and Ptolemy [BH+94].

### 2.2.5 Assertions

A typical simulation-based verification scenario comprises a set of test cases with their corresponding input stimuli for the DUV, the results are checked according to the specification. Such kind of tasks are often done manually by the designer/verification engineer. Especially for the analog subsystem, the most common verification task is to manually inspect the output waveform of the DUVs.

Assertions are statements that unambiguously express the property of a model to be true [CD+10], if one of the statements is false by any reason, it automatically indicates an error. With the help of assertion generated error reports, the verification engineer can debug and analyze the design without spending too much time to inspect the output waveforms of the DUV. Each statements may contain logical and temporal conditions, which have to describe design properties unambiguously and precisely [CD+10]. For digital design, assertions are comment-like statements coded within the DUV [WGR05]. For AMS Design, assertions can be implemented either as post processing evaluation expressions or as macro models placed in the netlist.

#### Assertion-based verification

Assertion-based verification, as shown in Fig. 2.13, returns true, when every conditions in the statements has to be fulfilled during the simulation [Mey04]. While originated in software design and thus well supported in digital design and verification flows, assertion-based verification is a rather new concept for

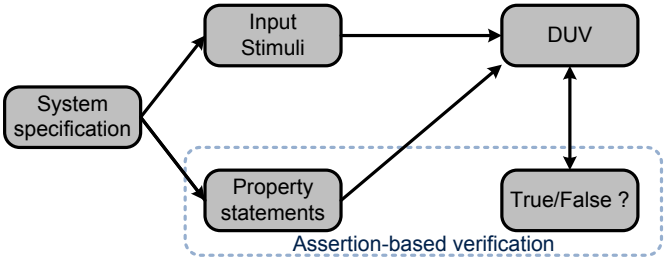



Figure 2.13: Assertion-based verification flow.

AMS verification environment [JPB10]. Currently, the idea of putting assertions into AMS design and verification flow are partly supported during mixed-signal simulations [Cad10] [JPB10]. Still, they suffer from various implementation restrictions due to the nature of analog signals, most AMS assertions still require model representation of analog circuits at higher abstraction level (s. chapter 3).

Assertion-based verification approach cannot cover all possible states of the system, due to the incomplete state space coverage of the simulation-based verification approach. However, implementing the statements leads to better formalization of the system specification.

### 2.2.6 Verification coverage

An AMS design cannot be verified exhaustively using simulation-based verification approach, due to its almost infinite inputs and states [WGR05] [Ste11]. However, the verification team still needs metrics to determine the quality and completeness of the verification, in order to deliver the design to tapeout with sufficient confidence. Verification coverage can be regarded as a measure of the mentioned confidence [Lam08]. Unlike the coverage measurement approach for digital design, for which a bunch of mature tools [Cad11d], standardized languages [Ber06] and flows already exists, define a proper coverage for the analog circuit is still an ongoing research topic. Generally, the concept of analog verification coverage suffers from two aspects:

1. Analog circuit verification still relies too much on manual interaction. Firstly, manual inspection of the DUV's output waveform is still a major task of the analog design flow, analog designers stick to this tradition. Secondly, verifying some of the specified parameters requires additional post processing and calculation of the DUV'S output waveform, this kind of specification is difficult to implement as assertions.This is also part of the

reasons, why analog design is still not well formalized and require designers' working experience based assumptions in addition to the specification.

2. Lack support: neither the language nor the tool are mature enough to be integrated into existing design flow.

In order to overcome the mentioned difficulties for AMS verification, the simulation-based verification method for analog centric mixed-signal circuits has to be classified further, so that the motivation of the proposed AMS verification flow can be outlined.

## 2.3 Simulation-based mixed-signal verification methods

On a finer scope, the simulation-based verification for AMS circuits and systems can be further classified into two different types: performance verification and functional verification. Although the boundaries between the respective types partially overlap, it is important to distinguish the main goals of both performance and functional verification, especially for the development of today's nano-scale, highly complex integrated RF mixed-signal designs.

### 2.3.1 Performance verification

As the name already indicates, the performance verification of mixed-signal circuits is a set of tasks to assure the circuits and systems are meeting their designated performance specification [IEE90], such as speed, linearity, gain or noise. Performance of the circuit is still the main concern of analog designers, thus, analog circuit verification still relies on performance verification and a lot of time in the analog design flow are spent to verify whether the noise figure or linearity fulfill the specified performance requirements.

One of the most established tools for the performance verification of integrated analog circuits is the Simulation Program with Integrated Circuit Emphasis (SPICE)[Nag75]. Since its first release, SPICE has got various improvements in terms of the solve stability, simulation efficiency and additional algorithm for RF circuits. Recent developments and optimizations have increased simulation efficiency of RF/analog subsystems by parallelizing and eliminating the need of solving large sets of differential equations at high speed [Cad] [Bda]. Still, analog design is a difficult task, complexity and chip size of todays RF SoCs are increasing drastically while technology nodes continue shrinking. This leas to the down scaling of supply voltage level and geometry size of the transistors, which makes the design of analog block even harder to meet the performance requirement.

Simulation of RF mixed-signal SoCs suffers generally from two aspects. One is the demand for small simulation time steps, arising from the high frequency

signals and analog nonlinearity, the other one is the purely tremendous size and complexity of today's circuits. While there are some enhanced simulation options for RF circuits such as periodic steady state analysis (PSS) [Cad11f] or harmonic balance (HB) [Cad11g] [Hei92] they all suffer from the fact, that they are not applicable to large and complex mixed-signal circuits [Kun05], which in general don't provide those basic periodic conditions. *The only simulation method that is currently implemented in an acceptable fashion for mixed-signal verification is transient!* Some other strategies, like Fast-Spice, try to fight the simulation requirements by using pre-compiled table models for the transistors, but in the praxis, these are not accurate enough for the desired simulations, when using aggressive simplification settings.

According to industry estimates in [Meh10], more than 50% of SoC design re-spins at 65 nm and below are due to mixed-signal and functional errors, such as connectivity bugs, misinterpreted digital control sequences or wrongly inverted signals [Che09]. This leads to the fact, that the complexity of analog blocks in mixed-signal design is increasing in a more functional way, performance verification of analog circuits using efficient analog solvers cannot cover all the functional state space of modern mixed-circuits. In order to achieve high confidence of the design, verification engineers have also to focus on ensuring the functionality of the system. The so called functional verification methods will be described in the following.

### 2.3.2 Functional verification

According to [Mey04], functional verification have to determine that the design will operate as intended based in the overall system specification. It is a structured and iterative process. Historically, functional verification was done when the hardware prototype has been built. If any issue is found, the prototype will be modified to fix the error, design change will also follow accordingly. This leads to following assumptions [Mey04], which has to be fulfilled:

1.) The internal operation of the prototype has to be visible for the verification engineer, in order to determine the source of the errors.

2.) The prototype has to be modifiable, in order to fix the error, so that the original design can be changed accordingly.

3.) Besides the quantity of the error has to be kept small, there should not be any severe errors in the prototype. If the prototype cannot power up, assumption 1 and 2 cannot be hold true.

For integrated mixed-signal systems, the prototyping causes significant costs. Additionally, the internal operation of a highly integrated SoC is barely visible for measurements, all measurements and performance evaluations are only possible

at the edge of the chip. A small functional error could mean that not any evaluation would be possible at all, which subsequently prevents the possibility to discover the source for the malfunction. Based on these facts, neither of the mentioned assumptions can be hold true. Therefore, it must be guaranteed that the chip itself is functional before a prototype is built(prior to tapeout), this leads to the purpose of functional verification for the state of art RF mixed-signal SoC.

What is functional verification of a typical design prior to tapeout about? For the chip's mask layout, there are plenty of tools available to do a physical verification [Cad09] [Men07] in order to ensure that the schematic and layout match each other. These methods are sufficient to guarantee that there are no further discrepancies between the schematic and the layout. Still, the physical verification cannot proof the functional correctness of top-level schematic itself down to each single building block. Functional verification is not solely about checking if all wires are connected on the top level, it's more about verifying whether timing requirements are met, if phase assignments are correct, if servo-loops are working and also if they are trimming the system into the right direction. In addition to this proof of functionality, it should be verified, that the correct version of a block has been implemented. To put it briefly, functional verification prior to tapeout conducts a set of simulations in order to prove the correct functionalities of the whole chip. It is a simulation-based verification, which requires high coverage in order to grant high confidence to the tapeout. While one might argue that the formal verification methods(s. section 2.1) are the only way to guarantee that the system is error free, but with the rising of the complexity level of the mixed-signal systems, the formal methods reach their limits considering the tremendous computational power required to solve and prove the exhaustive mathematical models.

Both formal and simulation-based verification techniques suffer from a common issue: neither there are tools capable of simulating the whole chip at transistor-level on time, nor there are enough memory or computational power to formally prove the functional correctness of the chip. The subsequent chapters and sections will introduce and illustrate the mentioned verification challenge and some ideas for its solutions in detail.

## 2.4 Fundamentals of hierarchical verification approach for RF mixed-signal SoCs

In order to understand and meet the verification challenge mentioned above, it is necessary to introduce two terms, *hierarchy* and level of *abstraction*. Both terms are the foundation of a formalized design methodology for mixed-signal systems.

### 2.4.1 Design methodology for integrated mixed-signal circuits

According to [IEE90] ***Hierarchy*** is defined as "A structure in which components are ranked into levels of subordination; each component has zero, one, or more subordinates; and no component has more than one superordinate component." Fig. 2.14 shows the conventional mixed-signal design flow. it comprises both

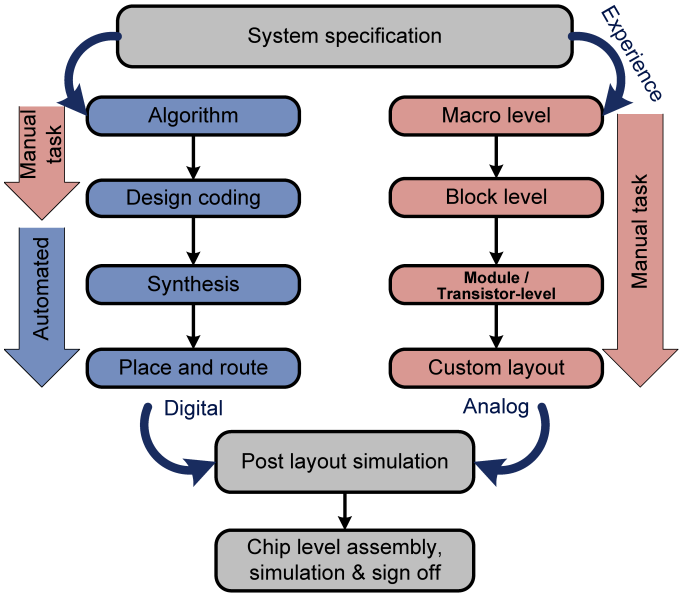



Figure 2.14: Conventional design flow for AMS circuits. The architecture of the design for analog and digital circuits are derived from the system specification. Unlike the formalized and automated digital design flow, most part of the analog design flow are manual tasks. The derivation of specification from the system down to each block are mostly based on the individual interpretation and experience of the designer.

analog and digital design as parallel paths and unifies both designs at the end of the design cycle. For the digital design flow, the algorithm for the system can be derived from the system specification. Based on the confirmed algorithm. The design of each blocks are done hierarchically using HDLs. A software controlled synthesis tool generates the proper hardware based on the HDL

design automatically. After the synthesis, the place & route tool generates the corresponding layout of the system and extracts the timing specification based on the transistor specification and routing parasitics. The overall digital design flow is therefore well formalized and automated.

In contrary to the digital flow, traditional analog design hierarchy is build up by interconnecting the devices such as transistors and passive components. In this way, small sub blocks are connected to larger functional units, design exploration, performance and functional verification are applied more or less together. This so called ***bottom-up*** approach has been standard for many years although it is hardly applicable for today's mixed-signal SoCs. On the one hand, the bottom-up approach could increase the probability of misinterpretation of the specification, which could lead to functional design errors. Since all the blocks are connected together at very late stage of the design flow, the chance to discover and correct the bug is low and the effort is high. On the other hand, most architecture problems of modern mixed-signal SoCs can only be disclosed, if all blocks, or at least a large subset, are merged and investigated together. If any error are found at this level, the cost and effort for re-design with the mentioned bottom-up approach will consume a significant part of the project resource. Still, the bottom-up approach remains the dominant design flow for small performance-critical AMS system [KC+00].

The ***top-down*** approach, proposed from [CM+92], [DS+94] and [CE+97], tries to address the mentioned issues and has been widely accepted in the industry. It is now the standard method to design complex mixed-signal SoCs. Here, the architecture is firstly defined and optimized at system level. The system is proven to work by system simulation tools [FH+05] (e. g. MATLAB® simulink). The requirement for the subsequent blocks and corresponding circuits are derived right after. Still, designers have to derive the specification of their blocks manually based on their own interpretation and experience. Commonly, the specification of an analog system is done in a hierarchical way. In the literature, this method is referred as ***Hierarchical decomposition*** [IEE90]: "A type of modular decomposition in which a system is broken down into a hierarchy of components through a series of top-down refinements."

For example, in a state of the art wireless communication SoC, shown in Fig. 2.15, the RF transceiver front-end is specified in 4 hierarchy levels:

- **System level** At this level, the overall system requirement is defined based on the specific communication standards. The standards dictates system parameters such as the allowed frequency range, the data rate, the signal dynamic range, the modulation scheme and the maximum allowed bit error rate (BER). Generally, the standard itself cannot explicitly define the system architecture, as along as the architecture derived from the standards fulfill the mentioned specification and requirements. For a

communication system capable of handling multiple standards, personal experience and interpretation of the system designer plays an important role at the construction of the architecture.

- **Macro level** As the system architecture is fixed, the basic requirements of the RF front-end can be firstly derived. Here, the transceiver architecture within its corresponding parameter, such as reference sensitivity level, interference level, dynamic range and overall noise figure, will be specified in detail.
- **Block level** Based on the architectural specification of the system, requirements for each block at the signal path can be derived subsequently. For Example, in Fig. 2.15, the frequency synthesizer of the receiver has to fulfill the specified phase noise and settling time requirements, these requirements can be used for the dimensioning of the charge pump current, loop filter band width and VCO (voltage controlled oscillator) circuit topology.
- **Module level** At the module level, the detailed circuit topology and parameters are decided. The circuit designer uses the specification here to implement and verify their design.

One essential issue during the hierarchical specification of the system exists in the fact, that the way to specify the analog part from system to module level is not explicit without ambiguity. E. g. there are several receiver architectures, which can fulfill the communication requirement, different receiver architectures leads also to different requirements for the subsequent blocks at lower hierarchy levels. Therefore, it is important to ensure the chosen architecture down to each module is the most optimized one for the application. Additionally, an adequate verification flow have to be established, in order to accompany the design flow at every hierarchy level. Directly verifying the whole system can be difficult for a numerous reasons: long simulation time, verification software limitation and reduced observability can clearly decrease the chance of finding bugs. Therefore, a large system has to be hierarchically decomposed into small subsystems or blocks recursively, until their sizes can be handled by verification tools [Lam08] with acceptable simulation performance.

### 2.4.2 Levels of abstraction

Today's wireless SoC contain several RF front-end in different operation modes for diversity, connectivity and communication. Each front-end operates over different channels and various gain states. Additionally, there are numbers of calibration loops and power saving modes controlled by the digital back-ends. Extensive simulation-based verification with reasonable coverage therefore can span large numbers of simulation-runs. Even the latest mixed-signal simulators

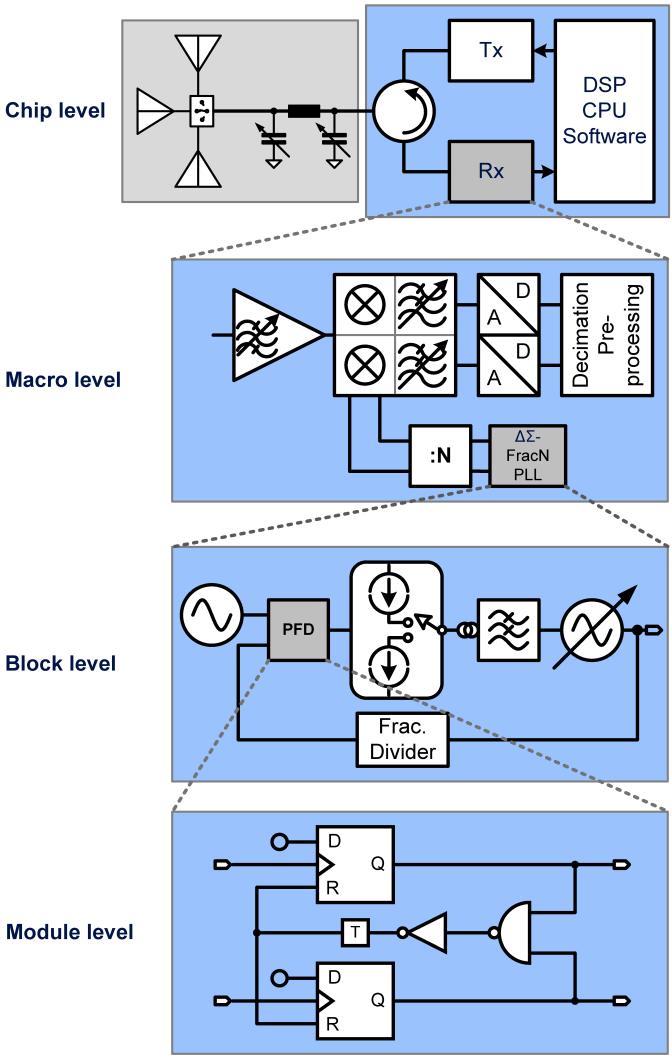



Figure 2.15: Design hierarchy for the RF front-end of a state of the art wireless communication SoC.

cannot handle a millisecond of chip level simulation [Che09]. Even the system can be hierarchically decomposed into small subsystems, the mixed-signal simulation at transistor-level can still take days or weeks to finish. Therefore, abstract representations of the system are required to increase the simulation performance.

According to [IEE90], ***abstraction*** is "A view of an object that focuses on the information relevant to a particular purpose and ignores the remainder of the information." Respectively, ***level of abstraction*** is the degree to which inforamtion about non-ideal effect or structure of the object is neglected compare to the dominant behavior of the entire system [MG08]. For digital design, the idea of using abstraction is clearly defined through the whole design flow There are 4 abstraction levels in general:

- *Behavioral* - At behavioral level (also referred as functional level), designer defines the arbitrary functional blocks, so that they can be used for system level simulation. A behavioral of the block contains the relationship between its input and output. Normally, only I/O behavior is specified, emphasizing the systems operation principles and there is no information about the internal structure of the system.
- *RTL* - Register transfer level description consists of logic combinational (multiplexer, adder) and sequential (latch, counter) components. RTL level is referred as an implementational view [Lam08], it is used for the data path design and evaluation of alternative architectures. It doesn't provide inforamtion about race condition or timing issues.
- *Gate-level* - At gate-level of the digital design, the blocks are grouped into basic logic gates, which describes the transistor-level behavior in discrete logic states and simple boolean operations. It can provide timing information of each gates and the overlying blocks and information about hazards, glitches and race conditions. Normally, the gate-level description is automatically synthesized based on the RTL level. After the layout generation with place and route tool, more detailed timing information include path delay can be provided at gate-level.
- *Electrical* - At electrical level, the digital design can be simulated with SPICE-like simulator. Although it is not common to perform an electrical level simulation for conventional digital design, it is required in some timing critical applications. Additionally, electrical level is also required for detailed physical level verification such as mixed-signal layout versus schematic.

As one can observe, the implementation of a digital system passes through multiple abstraction levels, from functional all the way down to the electrical level. From the RTL down to layout, each intermediate abstraction levels can

be subsequently generated using synthesis tools. Additionally, it is evident that the higher level of abstraction of the system, the faster it will simulate. Since digital circuits dominate most of the mixed-signal chip area, its design can profit significantly from the multi abstraction level structure for the simulation-based verification.

For the analog side, there are also corresponding levels of abstraction, shown in Fig. 2.16. The description of the various levels of abstraction in the analog design are based on the example of a continuous time filter shown in the same figure.

- *Pure functional* - This level is considered as the highest level of abstraction [MG08], the behavior of the analog system are described by simple mathematical equations consisting only the relation between input and output signal. In Fig. 2.16, the filter is described by its Laplace transfer function $H(s)$ in frequency domain.
- *Ideal behavioral* - At this level, circuit components, such as ideal opamps, capacitors and voltage-controlled voltage sources (VCVS) are connected together to realize the transfer function mentioned above. Similar to RTL level, it allows the designer to quickly check to architecture before the design goes more into detail.
- *Non-ideal behavioral* - Similar to the ideal behavioral description, this level consists additionally first order and second order non-ideal effects, such as nonlinear properties or dynamic behavior are included. E. g. in Fig. 2.16, the I/O resistance and basic dynamic behavior of the opamp has been included in the description.
- *Transistor-level* - Also referred as circuit level. Here, the opamp is represented in terms of the MOSFETS with detailed physical models provided by the technology foundry. All the performance parameter of the opamp can be simulated and evaluated. This is also the abstraction level, at which the analog designer are mainly focused during the whole design flow.

Although similar levels of abstraction in analog design are available, the possibility to automatically generate the corresponding lower levels of abstraction are very limited. Especially for the transistor-level design, there are no general tools for analog circuit synthesis [CR88]. Thus, the creation of the abstraction above the transistor-level has to be done manually, it is also one of the most important task for the verification engineers.

### 2.4.3 Hierarchical Verification

As mentioned in section 2.4.1, the design flow of mixed-signal circuits is organized in a hierarchical fashion. If the design process steps further, more detailed

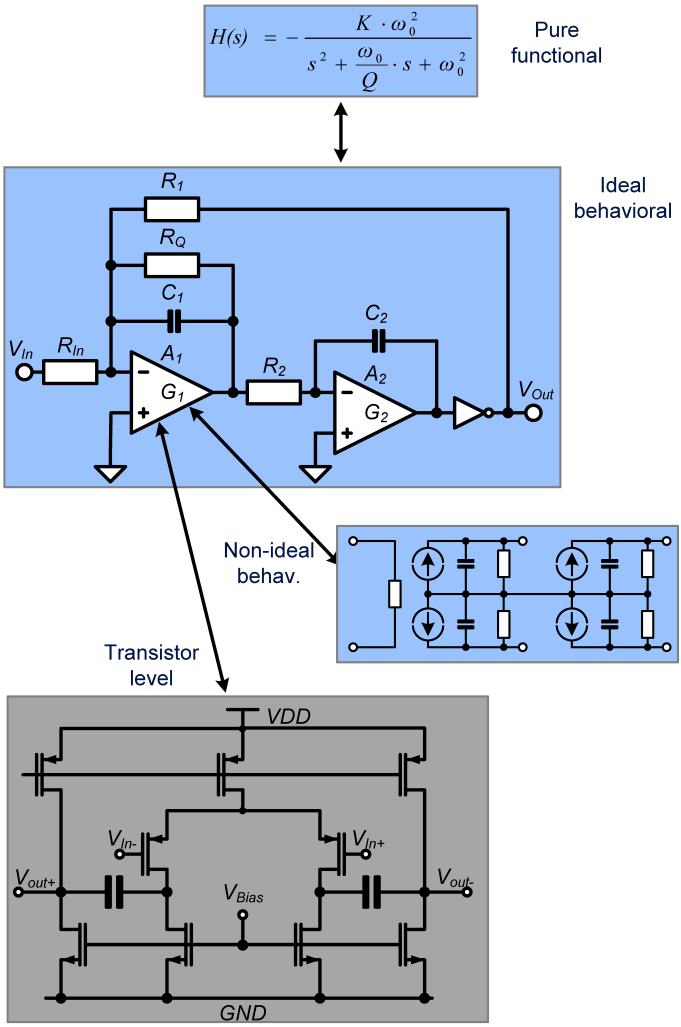



Figure 2.16: Levels of abstraction in analog design.

specification for the blocks at lower level will be given. At the end of the design process, the whole system has to be verified as a single unit, in order to deliver the silicon with enough functional confidence. How can this requirement to be met without consuming too much project time? One way is to seamlessly embed the mentioned levels of abstraction (s. section2.4.2) into the whole design flow.

### Hierarchical modeling

During the top-down design process, depending on the concrete flow, the abstraction representation of the system down to block level has to be implemented either by the designer or verification engineer. This allows a very fast architecture exploration, so that the structure of the system can be determined quickly by benefiting form the high simulation efficiency of the highly abstract system representation. This requires the work of model generation or behavioral modeling. Here, the term ***modeling*** is referred as a method to abstract from the detailed description of the system or block and replace them by simplified equation or I/O relationship.

For the digital design, the modeling and design work are unified, both abstract model at RTL level and detailed model at gate-level can be described with a single modeling language in a consistent way. E. g. using VERILOG HDL or VHDL (VHSIC-HDL: very-high-speed integrated circuits HDL). In this way, mixed-level simulation [MG96][7] of digital circuit at different abstraction levels can be done in a single simulator.

In contrary to the digital design, there is no automatic way to translate the abstract analog models into its transistor-level representations. Thus, the verification engineers have to transform the specifications into analog models in a language different from the design language[8]. This leads to the manual model implementation of the analog system though the whole hierarchy. According to [IEE90], this task is also called ***hierarchical modeling***: "A technique used in computer performance evaluation, in which a computer system is represented as a hierarchy of subsystems, the subsystems are analyzed to determine their performance characteristics, and the results are used to evaluate the performance of the overall system." Still, the analog models implemented according to the definition are performance oriented, which targets the main concern of the designer but might not consist enough details for the functional verification. Models for functional verification have to catch the latest snapshot of the actual

[7] Also referred as multi-level simulation [SN90]. Here, the term ***mixed-level simulation*** refers to the possibility of simulating sets of blocks at lower abstraction level, such as transistor-level, with the rest of the system described in higher abstraction levels [KZ04].

[8] Although some simple analog models can be described in SPICE macro modeling language, which is compatible to SPICE netlist, the model implementation for analog systems uses mixed-signal HDLs (s. chapter 3). Since the SPICE netlist language set is not capable of handling higher abstractions.

design specification down to each block in the system. Therefore, additional requirements have to be made for the modeling task, in order to achieve reasonable coverage of functional verification.

#### Requirements for hierarchical modeling and verification

For a state of the art SoC, verification engineer has to face 4 major challenges during the modeling and verification process:

- Enormous state space size yields lack of comprehensive coverage. Simulation-based verification requires to simulate each possible current state and input combination, in order to verify exhaustively that a chip is full functional.
- Finding source of incorrect behavior. finding source of the bugs, especially due to mixed-signal functional error out of thousands of signals, is time consuming and messy.
- No golden reference model. Especially for the analog side, since all the models are manually implemented, there is no models that can be served as reference.
- Absence of effective automation approach for analog modeling. As already mentioned, analog models are primarily targeting the performance issue of the analog design, there is no automatic way to generate additional functional details in the model for functional verification.

In order to meet the challenge and deliver a functional design, following requirements has to be fulfilled.

1.) Rather then verifying the entire system at once, tackle down the system into subsystems and verify them separately. Once the smaller, but more manageable subsystems are verified, the whole system can be assembled together and the verification engineer can simulate the more abstract version of them to ensure the functionality of the whole system.

2.) Determine the intent and the function of the DUV. Define the verification goals and make verification plans in advance. Use assertion instead manual inspection to disclose the bugs.

3.) Since the circuitry is organized in a hierarchical fashion, the models developed during the top-down design can never cover the required functional details for the functional verification. Thus, model refinement has to be done at certain stage of the design flow. ***Refinement*** is the process of turning a more abstract description into a more concrete description [Lam08]. The model refinement is typically processed bottom-up wise: using SPICE level simulation to accurately extract the blocks' properties at transistor-level, so that the model based on these properties can correctly map the

required functional and performance parameters for higher abstraction levels.

4.) Improve the design and verification flow. Although there is no automatic way to generate mixed-models, a more formalized design and verification flow can certainly reduce the manual interaction and yields less functional errors in the design.

In addition to the above mentioned requirements, the design and verification flow has to consider their different goals, namely, the different focus of designer and verification engineer.

For a typical top-down design approach, designer starts developing their models and testbenches at top-level[9] using very abstract and fast behavioral models. After the architecture exploration at top-level is done, the design of subsystems at transistor-level will follow piece by piece. The abstract behavioral models will be subsequently replaced by the transistor-level design. Since simulating the whole system at top-level is too time-consuming, the verification team has to start the model refinement and validation work as soon as the circuit design is partially finished. In this way, depending on the accuracy requirement of the test case, the top-level verification can be done by simulating circuits and refined models at different levels of abstraction in a reasonable time frame.

Basically, each model has to be simulated side by side with their transistor-level representation, during the model refinement flow, in order to check if they map the essential circuit behavior, if the pins of the I/O between circuit and model are match and if the simulation performance of the model is high enough for the top-level functional verification. Additionally, it has to be kept in mind, that the functional verification doesn't lend itself to performance verification. So the designer will still need to run SPICE simulation for their design. The difference between both teams are the input stimuli of the testbench: designer focuses more on a specific operation mode of the circuit and measure its performance, verification engineer, in contract, has to focus on the controls, switches of different operation modes and configurations. Only models with functional correctness and accuracy can find out if the bias current is wrong, or the control is inverted from what the specification requires.

Furthermore, the choice of modeling languages and techniques though the whole design hierarchy also plays an important role in the design and verification flow. Modeling targets between designer and verification engineer are different, which demands a modeling language capable of handling both requirements. The modeling language has also to be compatible with mixed-signal simulators, so that the mixed-level and mixed-domain can be supported without considering to co-simulate with an external simulator. Randomly choosing a temporary

[9]The term ***top-level*** is referred as the highest hierarchy level of the chip in the design flow.

suitable modeling language could lead to usage of different HDLs, which can lead to "HDL chaos" in the whole verification flow. In worst case, it can cause netlist or elaboration error, which prevents the simulator to execute or leads to wrong simulation result. Part of the available HDLs, their suitability for the functional verification of mixed-signal system and different modeling techniques at various levels of abstraction will be addressed in detail in the next chapter.

# 3

# Hardware Description Languages and Modeling Techniques for Mixed-Signal Verification

Behavioral modeling of RF/mixed-signal blocks at different levels of abstraction is the core part in the state of the art SoC design. At higher level of abstraction, unnecessary implementation details will be reduced, the model consists only essential functional details. Hence, the system behavioral becomes more understandable and the simulation performance will also increase. It is a method consistently getting more usage in the industry as a part of the design and verification flow for highly complex integrated systems. This chapter will provide an overview of the HDLs and modeling techniques used in this work.

## 3.1 Overview of available HDLs

Standardized mixed-signal HDLs, which are implemented by several companies and open source initiatives, offer a possibility to model the behavioral of the circuits and systems. They reduce the number of equations, by substituting the abstract coherences between the components with simplified mathematical descriptions. Modeling techniques and HDLs are consistently developing, in order to challenge the escalating complexity of today's SoCs. Fig. 3.1 shows part of the current available HDLs with their optimized task fields in the SoC design and verification flow.

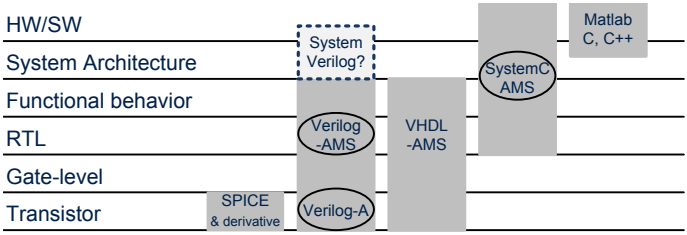



Figure 3.1: Part of the available modeling language and their reachable design hierarchy. Deduced and adapted from [BD+09]. The languages highlighted in ellipses are in the focus of this work.

As one can observe, there is no HDL currently, which is capable to handle the complete design hierarchy. For design focusing on chip level, Verilog and VHDL are most suitable. As the design hierarchy moves up, most HDLs are C-based. Needlessly to mention, that a seamless link between the levels of hierarchy and the corresponding HDLs is strongly required for modern SoC design.

### 3.1.1 Classical HDL

From the prospective of reachable abstraction levels within the design hierarchy and due to historical reasons, the current available HDLs can be segmented into two groups: classical and modern[1]. The classical ones are widely established and fully supported by most of the simulators in a state of the art design flow. The modern HDLs are more flexible and located at a higher abstraction level from the very beginning. Although the boundary between these to groups overlap at some points, the development of HDLs clearly shows a trend toward compatibility to software programming languages, such as C++. Still, classical HDLs dominates the lower level of design hierarchy, which is more hardware-related.

#### Verilog-AMS and VHDL-AMS

In the 80's the design of integrated circuits was netlist-based: designer creates the netlist of corresponding circuits manually, the resulting netlist served as the input for the simulator and physical verification tools. Classical HDLs such as Verilog [IEE01] and VHDL [IEE08] are the main languages for digital

[1] This kind of classification may not be very accurate. The intention of the author here is to show the HDL development trend, which is moving toward higher abstraction level and more capable of handling a wide range of design hierarchies.

design. The emergence of complex SoCs has created a vital demand for mixed-signal modeling and simulation, both HDLs became their AMS extensions (i. e. Verilog-AMS [Ver09]; [Cad11c] and VHDL-AMS [IEE07]). Although different standards, both HDL extensions offer additional capabilities to support the mxied-signal design and verification flow. The usage of individual HDL is strongly flow dependent, which is in turn rely on the tools used for the design. For the work presented in this thesis, design and verification tools with more focus on the Verilog HDL are used. Therefore, only the Verilog language will be briefly introduced here.

Verilog is a hardware description language that is very similar to the programming language C. However, in contrast to C, Verilog supports the important concept of time, which is crucial for the desired modeling of time dependent behavioral blocks. The initial goal in the development of this language standard was to provide a HDL, that supports digital as well as analog constructs. Since this is a high aim for two commonly completely different simulation strategies, several subsets have been developed over time: Verilog supporting only discrete time events and Verilog-A as the all-analog subset. Since a few years, Verilog-AMS is now available as a derivative of the original language. It provides both continuous-time and event-driven modeling semantics, and is therefore suitable for analog, digital, and mixed-signal circuits. Current simulators are able to split the mixed-signal system into an analog (continuous time) and event-driven (discrete time) partition, each having its own solving approach[2] using a synchronization procedure to couple both simulation domains. For electrical systems, Verilog-AMS supports the predefined continuous time discipline `electrical` (consisting of potential and flow) and the data type `wreal`, which is handled by the event-driven engine and commonly interpreted as analog voltage value with double precision(although it could also be treated as current in the same way). Furthermore, common digital logic types are also available.

It also has to be mentioned that in addition to Verilog, SystemVerilog [IEE09] and its mixed-signal extension SystemVerilog-AMS is a new standardization approach targeting to gain more market share. Compared to Verilog, SystemVerilog especially introduces structures, pointers and recursive subroutine capabilities. At the moment, library extensions such as the Universal Verification Methodology for SystemVerilog (UVM) provide reusable building blocks for architectural modeling and verification environments [Acc11]. SystemVerilog can be considered as the advanced development of Verilog with object-oriented feature, which could come in handy for the baseband modeling approaches described in section 3.2.3. Still, SystemVerilog itself focuses more on digital design at the moment and its mixed-signal subset is still under standardization process. Thus, it is a maturing and promising HDL, but not all

[2] Please refer section 2.2.1, 2.2.2 and 2.2.3 for detailed description.

simulators support the full capabilities of the language.

In a similar way as the relationship between VERILOG and VERILOG-AMS, VHDL is a subset of the all-in-one language definition VHDL-AMS. Due to historical and regional reasons mentioned in [RS09], VHDL is commonly used for digital designs and rarely for analog modeling, which maybe because the language is less intuitive for analog designers than VERILOG. Although its possibilities to define new data type constructs outperform VERILOG, most state of the art layout and design software only support VERILOG for top level netlisting, since the VERILOG-AMS language construct is closer to the circuit design.

### 3.1.2 Modern HDL

As already mentioned, the trend of HDL development is moving toward higher level of abstraction, since this is the only way to handle the rising complexity of today's SoC design. Additionally, expanding through the whole design hierarchy shown in Fig. 3.1 is also the focus of modern HDLs such as SYSTEMVERILOG and SYSTEMC AMS, in order to meet the requirement of software/hardware co-design. This is also the reason, why most of the modern HDLs are C-based. In the following, a brief introduction of SYSTEMC AMS will be given.

#### SystemC and its AMS extension

SYSTEMC [AC06] [GL+02] [BD+09] is an standardized open-source C-based HDL for modeling both hardware and software at different abstraction levels. It is a modeling platform based on special C/C++ classes, libraries and a simulation kernel for designs from the system-level, behavioral-level to register transfer level. SYSTEMC has enjoyed huge popularity since its introduction in 1999. Since then, it has gained a wide acceptance and support from the industry side. Its IEEE standardization in 2006 [AC06] has been coordinated by the Open SystemC Initiative(OSCI) [Ope]. For hardware design, SYSTEMC offers the Discrete-event (DE) model of computation (MoC) [Gro02].

A typical SYSTEMC module contains at least three parts [GL+02]. The first part comprises the definition of ports, signals and variables. Using SYSTEMC classes `sc_in`, `sc_out`, `sc_inout`, `sc_signal`, the designers can define the ports and signals used for this module. The second part consists both the initialization of internal data and variables. The third part is the declaration of SYSTEMC processes as the member functions of the module, which describes the module's functionality. Basically, there are two types of process in SYSTEMC, which are typically created statically, namely the **`SC_THREAD`** and **`SC_METHOD`**. Both types of process are created prior to the simulation execution. The `SC\_THREAD` process will be executed automatically at the start of the simulation. Commonly, it uses infinite loops to model hardware logic. The **`SC_METHOD`** process is similar to the

VERILOG `always@` block [BD+09], which can be used to model the event-driven behavior of the model. Since the main focus of standard SYSTEMC DE MoC are pure digital hardware, numerous approaches has been pursuit in parallel, in order to push the AMS extension of SYSTEMC to standardization.

The SYSTEMC-A approach presented in [AJK05] uses a similar approach as SPICE simulator. It supports systems variables, analog components and user-defined equations as a set of ordinary differential and algebraic equations for the analog solver. The analog solver uses the lock-step method for mixed-signal time step synchronization (s. section 2.2.3) with SYSTEMC and requires the modification of the standard SYSTEMC kernel.

The SYSTEMC-WMS approach in [OBC05] allows to model the analog behavior by implementing an wave theory-based analog interfaces(wave channel) and accounting the load effect at the interface, which is used to connect the subsequent analog blocks. This approach is fully compatible with the current version of SYSTEMC, but require the knowledge of the scatter parameter of each single block, which might not be easy to find out for an integrated mixed-signal system[3]. Additionally, SYSTEMC-WMS require small discrete time steps for it analog solver, which leads to the slow down of overall simulation in SYSTEMC.

In contrary to both the mentioned approaches, the standardized AMS extension for SYSTEMC, namely SYSTEMC-AMS[4] [Sys08b], uses a layer-based synchronization mechanism to synchronize with the desiccate event SYSTEMC simulation kernel [BS+11]. Fig. 3.2 shows the detailed architecture of the OSCI SYSTEMC AMS extension 1.0 standard based on the information from [Sys08a]; [BS+11]. In order to support AMS behavioral modeling at different levels of abstraction, it consists three modeling formalisms(also called MoCs: Models of Computation): Timed Data Flow (TDF), Linear Signal Flow (LSF) and Electrical Linear Network (ELN) MoCs [DH+08] synchronizing on top of the existing standard SYSTEMC kernel. In the following, the three mentioned MoCs will be briefly introduced.

#### SystemC AMS model abstractions

Fig. 3.3 shows the three current MoC implementations for the AMS extension of SYSTEMC.

[3] On chip $S$-parameter is hardly to estimate. Commonly the impedance between the blocks on chip is assumed to be high. Finding out the actual value and $S$-parameter, require either new testbench or on chip measurement, both are not feasible.

[4] During the standarization process, the Proof of Concept(PoC) implementaion of AMS extension to SYSTEMC, which has been implemented from Fraunhofer IIS/EAS, is called SYSTEMC-AMS. In the following, only the standard name SYSTEMC AMS will be used, although most simulation results are based on the PoC implementation.

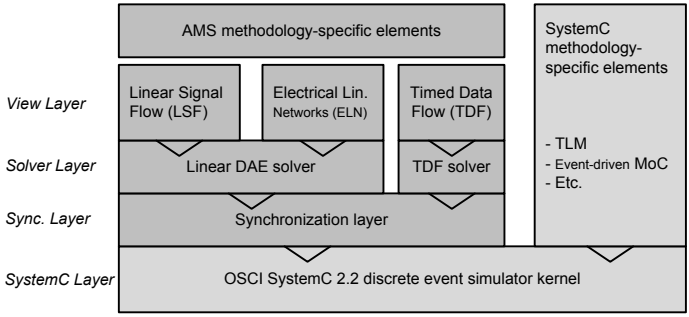



Figure 3.2: Standard SystemC AMS MoCs and the synchronization approach with the current SystemC kernel according to [DH+08]; [Sys08a] and adapted from [BS+11]. The standardized synchronization layer will offer the freedom to integrate used-defined MoCs and solvers built upon it.

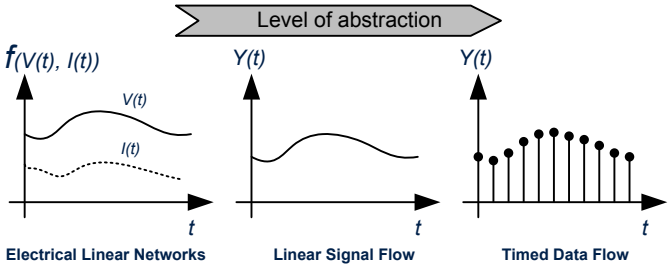



Figure 3.3: Currently available SystemC AMS formalisms according to [Sys08a]. As one can observe, the TDF MoC possesses the highest abstraction level in the current implementation of the AMS extension.

**Electrical linear networks (ELN)** define the basic behavior and components of electrical networks. The MoC can instantiate predefined linear network primitives such as resistors, capacitors, inductors and controlled source. Similar to the SPICE netlist, the modeled network of electrical components in ELN MoC require the simulator to solve the corresponding DAE(s. chapter 2). Due to its similarity to SPICE, the ELN MoC can describe the continuous-time relationships between voltage and current in a conservative system, thus it can be considered as the lowest abstraction level out of the three. Needlessly to mention, that the simulation time and modeling effort for ELN are high. Additionally, in the current PoC implementation, only linear DAE solver is realized, which restricts the modeling capability to linear component only.

**Linear signal flow models (LSF)** supports the modeling of analog behavior by using a set of basic mathematic operations such as addition, multiplication or first-order time derivation. Similar to ELN, a LSF model is built up on a network of such primitives represent the time or frequency domain behavior of the system, which is also solved by the build-in linear DAE solver. However, since the LSF models are generally non-conservative, it can be regarded as a higher level of abstraction compare to ELN. Still, its capability to model a complex system is limited.

**Timed dataflow models (TDF)** is used to simulate the sampled, discrete-time signals, which can represent any C++ type. TDF is implemented to avoid the overhead of the dynamic discrete-event scheduling imposed by the SystemC execution semantics. There for, it can accelerate the simulation by defining a static schedule [Sys08a]. For model representation of electrical systems, TDF signal can represent voltage or current at the given time point. Both mentioned features come very handy for the modeling of a complex transceiver system, which will be described in section 5.3.2 and 6.3.2.

### 3.1.3 Comparison between mentioned HDLs

As Fig. 3.1 has been shown, both Verilog and VHDL HDLs are targeting the same design hierarchy. Both languages support the description of networks as conservative or signal flow based. Both are capable to describe AMS system with its mixed-signal extension. Small difference: VHDL is ADA [WWF87] like and therefore case insensitive, Verilog is C like and therefore case sensitive. Since both HDLs have very strong tool dependency, It is therefore a choice of used tools for design and verification, which will decide the usage of the HDL. The primary advantage of VHDL is clearly its capabilities to handle used defined data type constructs, which offers more modeling freedom but may leads to model-incompatibility to the circuit level counter part. E. g. if the signal is

defined as an array or matrix type, it is not compatible to the netlister for mixed-signal simulation. This leads direct to the primary advantage of Verilog: most of the AMS design is netlisted as a Verilog-AMS netlist, which gives both the designer and verification engineer all the features of Verilog [IEE01] as well as its AMS extensions [Ver09]. Additionally, Verilog-AMS also offers a vital feature for mixed-signal simulation, namely the automatic insertion of predefined connect modules[5](CM) at the interface between both analog and digital domains during the elaboration phase which will described in detail in chapter 4.

For pure digital implementation, both Verilog and VHDL are more closed implementations, SystemC in contrary is open source. In SystemC, open source allows kernel-level extensions [Pat05] as well as layer-based synchronization approaches such as SystemC AMS. Furthermore, VHDL or Verilog provides mainly the RTL abstraction, but SystemC provides the simulation kernel suitable for designs from RTL level to System designs [Pat05]. This also allows the model reuse at different levels of abstraction within the design hierarchy [CA03]. Last but not least, SystemC AMS is targeted to model a heterogeneous system, and therefore supports software and hardware co-design [Pan01]; [CS+06]. The AMS extension of the classical HDLs are still focusing on the hardware level. Apart from the mentioned facts, the flexibility offered by SystemC AMS is one of the most important key points to use it for system integration. E. g., SystemC AMS allows users to define their own signal type, which will bring a lot of convenience to build the baseband equivalent models without changing of the structure of ports. However, both ELN and LSF MoCs provided by SystemC AMS will invoke the analog solver for the simulation, which will greatly degrade the simulation efficiency. Hence, only the TDF MoC and pure SystemC discrete time MoC will be used for the functional verification approaches provided in this work. Table 3.1 shows a comparison of the mentioned modeling languages.

Due to the tooling used in this work, the choice of modeling languages are Verilog-AMS and SystemC, targeting different design hierarchies and supporting the functional verification of the whole chip.

## 3.2 Modeling techniques for mixed-signal verification

Model generation, also called modeling, is the key components of today's mixed-signal design and verification. HDLs offer a possibility to model the behavioral of the circuits. They reduce the number of equations, by substituting the abstract coherences between the components with simplified mathematical descriptions.

[5]Please refer section 2.2.3 for the explanation of the mixed-domain interface.

| **Language** | MATLAB® *simulink* | VERILOG-A | VERILOG-AMS | SYSTEMC |
|---|---|---|---|---|
| **Simulation domain** | Var.[a] | Analog | Mixed-signal, event-driven | event-driven[b] |
| **Signal value** | Continuous, discrete | Continuous | Continuous, discrete | Continuous, discrete |
| **Simulation performance** | Fast[c] | Slow | Medium[d] | Fast |
| **Pin compatible ?** | No | Yes | Yes[e] | Yes |
| **Model validation effort** | Medium | Low | Medium | Medium - High |
| **Model creation effort** | Low | Medium | Medium | Medium - High |
| **Reachable abstraction** | Medium | Low | Low | High |

[a] Depending on the Simulink simulator definition of signal, in most case discrete time.
[b] The AMS 1.0 extension also offers linear analog solver. However, any case of solving DAE will slow down the simulation. Hence, only the TDF and DE MoC will be considered.
[c] Using only the Simulink discrete time simulator
[d] Depending on the usage of the analog subset. In most case, the analog subset will slow down the simulation due to the DAE solving process. Hence, use only wreal data type will speed up simulation.
[e] Both VERILOG-A and VERILOG-AMS offer only pin-compatibility for passband models. For baseband modeling approach, only VHDL and SYSTEMVERILOG offer the pin compatibility.

Table 3.1: Comparison of the mentioned modeling languages. VHDL and SYSTEMVERILOG are not shown here, due to similarities to the VERILOG HDL.

Fig. 3.4 shows the required models within the proposed mixed-signal design and verification flow.

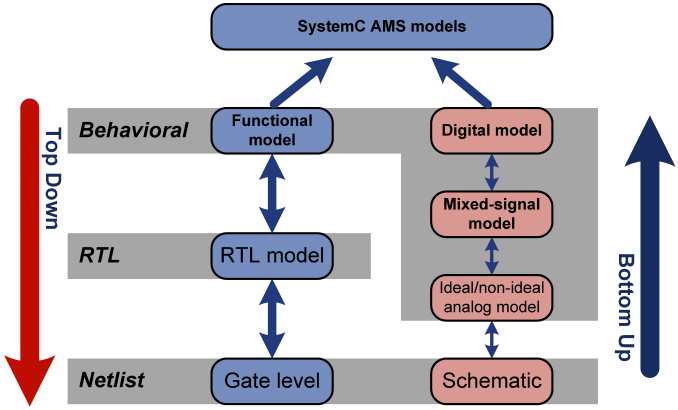



Figure 3.4: Models at different levels of abstraction within the mixed-signal design flow. Each model in digital design are compatible with the digital simulator. In contrary, for analog design, only analog models can be simulated in an pure analog simulator.

For pure digital design, the modeling and design task are tightly integrated into each other, since the design flow is well formalized and automated. This is the reason, why digital blocks require almost no additional modeling work. Furthermore, digital simulators work directly with any state of the actual design and models. Thus, for the mixed-signal verification flow discussed in section 2.4.1, the required levels of abstraction through the whole design hierarchy at chip level are available. In contrary to the modeling tasks in digital design flow, analog blocks require hand-crafted models at every state of the design. In the following, modeling techniques at different levels of abstraction for the analog blocks will be discussed.

### 3.2.1 Analog modeling

#### Analog modeling using Verilog-A

Verilog-A supports behavioral and structural descriptions of an analog block. Structural descriptions define the system by interconnecting components such as other modules or pre-defined components. In contrary, behavioral descriptions

are mathematical relations describing the I/O behavioral. Analog modeling can be categorized further in three approaches: conservative modeling, signal-flow modeling and event modeling.

**Conservative modeling** In conservative models, both potential and flow are required to describe the system's behavior. One useful concept for modeling of conservative systems within VERILOG-A is the concept of a branch [FM98]. It is defined as a path of flow between two nodes, where a node represents the interconnection of two or more branches [Cad03]. For an electrical system, every branch has an associated potential(voltage *V(p,n)*) and a flow(current *I(p, n)*). The reference directions for branch and flow are shown in figure 3.5.


Figure 3.5: Associated branch, potential and flow in VERILOG-A according to [FM98]. a.) Abstract representation. b.) Electrical voltage and current reference direction for a branch.

The contribution statement <+ assigns an expression to a branch (either potential or flow). The KVL and KCL are used to resolves the relationships between the interconnected nodes, resulting in the admittance matrix for the MNA. Using behavioral description, only abstract mathematical operations are necessary to define the blocks' I/O behavioral. Therefore, less nodes and equations are required to describe the same functionality. This also leads to the reduced of entries of the matrices to be solved, which is the major reason why simulation with analog behavioral models benefits from a significant speedup.

**Signal-flow modeling** In some cases, it is sufficient to describe the signal flow of the electrical system based on only one of the two mentioned quantities [KZ04]. E. g. use only voltage discipline to model the output of a voltage controlled oscillator(VCO). Here, the signal flow at the VCO output can be described in a more abstract way. The can be connected to conservative models, but exhibit higher simulation efficiency. So the *signal flow models* are frequently used during the analog top down design phase.

**Modeling analog event** In order to model event-based analog behavior, such as sample and hold, or relay, the event operators are used to evaluate the input

signals. Commonly the ***cross*** function is in charge to detect the clock signal or some threshold behavior. Analog events allow VERILOG-A models to execute code conditionally if a certain event has occurred. The conditional execution allows to detect e. g. zero crossings. Here the **`cross`** and **`timer`** are commonly used to model such digital, reactive behavior in analog models. In this way, mixed-signal circuit such as $\Delta\Sigma$ ADC can be modeled in pure analog fashion without using a mixed-signal environment. More modeling possibilities for events can be found in the current version of the VERILOG-A and VERILOG-AMS standard [Cad11c]. The output signal of the event-driven blocks will remain unchanged, as long as no input events occur. This leads to the relaxation of analog time steps and therefore even higher simulation performance compare to signal-flow models. Nevertheless, event-based analog models still remain in analog domain. For higher abstraction, it should be superseded by real valued modeling as described in section 3.2.2.

**Arithmetical and Analog Operators** Both mathematical and analog operators are available to support the behavioral modeling in VERILOG-A.

- *Mathematical operators* - VERILOG-A includes basic exponential and trigonometrical functions and operators for arithmetical and logical operations. Additionally, numbers of random distribution functions are available.
- *Analog operators* - maintain an internal state and thus are subject of several restrictions. They are not supported in functions or loop statements. If applied in a conditional statement, only static input expressions can be used. The analog operators include functions for time derivatives and integrals of signals, as well as various filter implementations.

### Analog modeling using SystemC AMS

Similar to the modeling techniques in VERILOG-A, the ELN and LSF MoCs in SYSTEMC AMS can be used for conservative and signal-flow modeling in the analog domain. Still there are three reasons showing that the analog modeling capability of SYSTEMC AMS is limited compare to the VERILOG-A models.

1.) **Limited option for analog operator**: Compare to VERILOG-A models, the ELN or LSF MoCs don't offer additional analog operator for event detection, signal limitation or transition of signals. Although this operators can be extended by user-defined module due to the flexibility of C++, it limits the usability of the HDL.

2.) **Limited support for analog analysis**: The current PoC implementation of SYSTEMC AMS only has a linear DAE solver for analog simulation,

which generally limits the implementation freedom of analog models. Besides that, only transient and AC, but no steady-state or periodic stability analysis are supported. Additionally, the simulation time step has to be pre-defined by the user, which may leads to convergence issue or inaccurate results.

3.) **Issues at mixed-domain interface**: VERILOG-AMS offers the automatic insertion of necessary connect modules at the mixed-domain interface for mixed-signal simulation, whereas the CMs have to be inserted manually at the interface between ELN and TDF MoCs [Sys08a].

Based on these facts, it can be seen that SYSTEMC AMS cannot show it true potential by only modeling analog behavioral at low abstraction level. The usage of analog MoCs in SYSTEMC AMS is only feasible, if a large digital or mixed-discipline system require certain dynamic behavior of the analog system, e. g. a sensor front-end in the electronic stability control system(ESC). For general analog model implementation in RF/analog design, VERILOG-A is still the first choice at the moment, due to the better but only commercially available analog solver support.

Furthermore, the analog simulation performance is also not sufficient enough for long term simulation of a whole mixed-signal SoC, especially if multiple RF front-end are involved. Even with optimized analog models which reduces the number of equations for the simulation at minimum and state of the art high-performance simulators, it can take days and weeks to simulation just a couple of millisecond chip-level action of a modern RF SoC [Che09]. The bottleneck of the simulation performance for mixed-signal systems originates from the solver mechanism of the analog simulator, which solves the whole system matrix at variable step size depending on the highest frequency and grade of nonlinearity in the system(s. section 2.2). In worst case, even with the models, the problem arising from the nonlinearity and high frequency will still force the simulator to solve the whole system for many small time steps and run into convergence issues. In order to speed up the simulation and hence to increase functional verification coverage, the only choice at the moment is to move the analog behavior into digital domain and exploit the speed and capacity of a purely event-driven simulator. In consequence, event-driven or equivalent baseband models of analog blocks have to be implemented for the top-level and chip-level functional verification.

### 3.2.2 Event-driven modeling

Analog behavioral model require SPICE-like simulator to solve the DAEs, which can be regarded as the main bottleneck in the simulation performance for the simulation-based functional verification. Especially for RF and nonlinear circuits,

the resulting small time step of analog simulator by solving the whole system matrix and additional mixed-domain synchronization process will greatly reduce or even wipe out the coverage of functional verification prior tapeout.

### Event-driven modeling using Verilog-AMS

Fundamentally different to analog simulation, the digital simulation uses an event-driven (also known as data flow) approach, which leads to a sensitivity list, where each signal change triggers new evaluation of logical expressions at the connected nodes in a sequential manner [HC09], but not for the whole system as the analog simulator. It is obvious, that the digital domain uses the aspect of time (data flow: one calculation after the other) and therefore does only provide transient simulation possibilities. One important aspect here is to mention, that digital solvers are not capable of iterating backward in time to the author's knowledge. Digital simulation concepts therefore can't handle iterative features as used in backward Euler and other formulas.

The Event-Driven modeling approach excludes the high frequency signal path from the analog domain, e.g. using the **wreal** data type introduced by Verilog-AMS [KZ04]; [JGH07]; [HC09]. It possesses continuous value(floating point numbers) in discrete time domain[6], so that analog values and behavior can be partially mapped into an digital simulator based on the principle of oversampling. In other literature [HC09], this modeling technique is also referred as *Real Valued Modeling*(RVM) or *real number modeling* [WVM+09]. Using the **wreal** data type, the analog signal path, especially the RF and nonlinear part, can be extracted from the analog matrix and put into the digital, event-driven domain. Although keeping the high frequency signals does not lead to less calculations per period (they even might be increased depending on the sampling procedure [JGH07]), but the number of equations that have to be solved for each of these time steps will be reduced dramatically, since only isolated portions are calculated but not the whole system matrix like the analog solver. The theoretical accuracy of **wreal** (64-bit double precision) is comparable to standard analog signals and offers even a better dynamic range as signals in the analog domain [Joe08]. Since the whole system is modeled in a data flow manner, issues such as convergence error coming from analog simulator can be avoided, resulting in reduced simulation overhead and hence in a significant simulation speedup. [JGH07]; [WJ+09] report a speed-up of over 25x compared to transistor-level simulation, and 16x compared to analog behavioral models.

[6] **wreal** in Verilog-AMS is similar to **real** in VHDL-AMS.

### Event-driven modeling using SystemC

As one can observe in Fig. 3.2, there are two options to implement event-driven model in SystemC. Either use the TDF MoC from the AMS extension or use the pure SystemC DE MoC. Although different in semantics between SystemC and Verilog-AMS, the principle to implement the models are almost the same. However, it has to be mentioned, that the TDF MoC is a discrete time modeling style, which only considers the signal in pre-defined and fixed sampling time steps [Sys08a]. This mechanism uses its own local time annotation and does not interact with the SystemC's discrete-event kernel [Sys08a]. Therefore, using the TDF MoC can reduce the overhead of the dynamic discrete-event scheduling imposed by the SystemC execution semantics. In general, the TDF MoC can be considered as a special case of event-driven modeling technique. Still, interaction between TDF and DE MoCs requires additional synchronization process, which can be done using additional I/O converter port [7].

### Sample-based representation of analog signals

The aforementioned event-driven modeling technique uses real value to represent analog signals in event-driven domain. Higher level of abstraction can be achieved by only characterizing either potential or flow of the conservative system in the discrete time simulator. The digital simulation kernel reconstructs the signal with the model of zero-order hold(ZOH) [BN+05]. Therefore, real valued representation of analog signals in event-driven domain are sample-based. In order to achieve reasonable accuracy with the sample-based representation of real valued signals, the sampling rate of the signal must come up with certain oversampling ratio.

Mathematically speaking, the minimum sampling frequency of an analog signal has to be twice higher than the highest frequency component, this is referred as the Nyquist sampling rate. However, as one can observe in Fig. 3.6, applying such low sampling rate in the practice cannot properly illustrate the signal in the time domain and therefore hinder further debug possibilities. To avoid this problem, a common rule of thumb in the practice is to set the sampling rate from 10 to 20 times the highest frequency of interest [Joe08]; [Wan08]; [Lyo10]. In general, an oversampling ratio(OSR) of $N_{OSR} \geq 10 \cdot N_{Nyquist}$ has to be used in order to accurately represent the analog signal in the discrete-event time domain.

Besides the mentioned OSR estimation, the corresponding time step has also to be defined in the digital simulator. Therefore, an event-driven model of the analog block requires either its own internal clock(also called virtual clock) for

[7] The detailed description of the synchronization process between TDF and DE is not the main focus of this work, please refer [Sys08a], section 2.4 for more information.

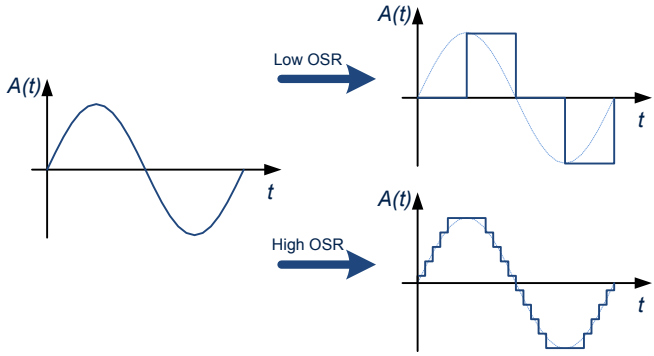



Figure 3.6: Time domain representation of sample-based analog signals. Here, a sinusoidal has been sampled with different OSRs, one can observe that the signal with low OSR cannot properly illustrate the original signal form in the time domain and thus reduce the debug possibilities.

determining the corresponding time step or a reactive behavior triggered by changes of the input signals. Sometimes, both requirements are necessary in order to accurately model the analog behavior in the event-driven domain.

### 3.2.3 Equivalent Baseband Model

While models at higher levels of abstraction in general can lower the number of equations to be solved, the signal, that is treated as the models' in- and outputs, can be implemented in several ways. Each signal, that is to be simulated, consists of data, typically modulated upon a carrier frequency. The real-world/physical representation of a signal value, which especially includes the carrier frequency as well as the modulated data packet itself is defined as ***Passband*** signal

A typical abstraction of this physical identity is to use a mathematical approach to strip the carrier frequency from the signal itself, leading to lower frequencies in simulation and therefore larger time steps. Especially for the simulation of RF mixed-signal SoCs, where the typical bandwidth of the actual data is several decades lower than the carrier frequency. Since the carrier frequency (assumed as a constant value) is of no additional information for the system and hence does not need to be taken into account during the simulation.

A modulated data $x(t)$ with the carrier $\omega_0$ can be denoted as

$$x(t) = A(t) \cdot \cos\left[\omega_0 t + \Phi\left(t\right)\right] = A(t) \cdot \Re\left\{e^{j(\omega_0 t + \Phi(t))}\right\}, \tag{3.1}$$

where $A(t)$ and $\Phi(t)$ represent the amplitude and phase information of the data packet. As shown in Fig. 3.7 and according to [JBS00], this can be shifted towards zero carrier frequency, so that the equivalent ***baseband*** signal can be described as:

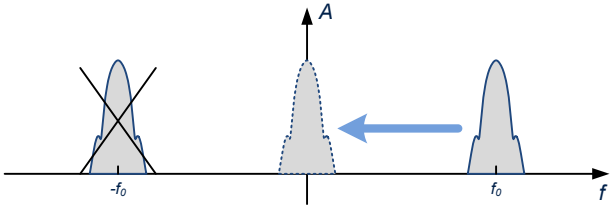



Figure 3.7: Obtain the equivalent baseband signal representation by shifting the passband signal towards zero in the frequency domain.

$$\begin{aligned} x_{BB}(t) &= A(t) \cdot e^{j(\omega_0 t + \Phi(t))} \cdot e^{-j\omega_0 t} = A(t) \cdot e^{j\Phi(t)} \\ &= A(t) \cdot [\cos(\Phi(t)) + j \cdot \sin(\Phi(t))] \end{aligned} \tag{3.2}$$

The complex equivalent baseband signal, which is independent to the carrier frequency $\omega_0$ can be denoted as:

$$x_{BB}(t) = I(t) + jQ(t)$$

The transformation back to its passband representation can be achieved using

$$x(t) = \Re\left\{x_{BB}(t) \cdot e^{j\omega_0 t}\right\}. \tag{3.3}$$

The information which is necessary to describe a passband signal using its equivalent baseband representation consists therefore at least of three parts: in-phase component $I(t)$, quadrature-phase component $Q(t)$ and the carrier frequency itself $\omega_0 = 2\pi f_0$. An equivalent representation using the polar form can be achieved by using Cartesian to polar transformation:

$$A(t) = \sqrt{I(t)^2 + Q(t)^2} \tag{3.4}$$

$$\Phi(t) = \arctan(\frac{Q(t)}{I(t)}) \tag{3.5}$$

Since the equivalent baseband representation is a further abstraction of the signal itself, the baseband modeling technique is not restricted to specific simulation/modeling domains. However, the obvious thing to do would be to combine both baseband and event-driven modeling techniques, in order to achieve even

higher simulation performance.

### Baseband representation of multiple signals

For simulations including multiple signals(e. g. blocker, nonlinearity effects, harmonics. see Fig. 3.8(a)), each single signal should be converted to its own baseband representation, which leads to a representation in the spectra as shown in Fig. 3.8(c) - otherwise the resulting bandwidth will soon become so large, that there is no further speedup (see Fig. 3.8(b)). The composed signal shown in Fig. 3.8(c) can be denoted using a matrix notation as shown in equation 3.6.

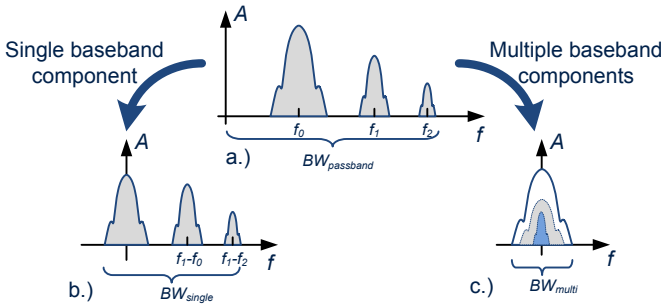



Figure 3.8: Equivalent baseband signal representations of a passband signal consisting higher order harmonics. a.) The passband representation. b.) Equivalent baseband representation using only one set of baseband component. c.) Equivalent baseband representation using multiple sets of baseband component.

$$x_{BB,multi}(t) = \begin{Bmatrix} I_0(t) & Q_0(t) & f_0 \\ I_1(t) & Q_1(t) & f_1 \\ \dots & \dots & \dots \\ I_n(t) & Q_n(t) & f_n \end{Bmatrix} \tag{3.6}$$

The number of necessary calculation packages in a specific time period for each representation, can be roughly estimated as (assuming $BW \ll f_n$ and an specified OSR)

$$\begin{aligned} \text{Passband Fig. 3.8(a)} &: OSR \cdot max(f_0, f_1, \dots f_N) \\ \text{Single basebandFig. 3.8(b)} &: 2 \cdot OSR \cdot max(f_0 - f_n, \dots) \\ \text{Mult. Baseband Fig. 3.8(c)} &: 2 \cdot OSR \cdot BW \cdot n, \end{aligned} \tag{3.7}$$

For $BW \ll f_1 \ll f_2$ the baseband representation with multiple sets of baseband signal components is for a reasonable number of simultaneous signals $n$ obviously more efficient than the others.

## 3.3 Issues and limits of presented modeling techniques for verification

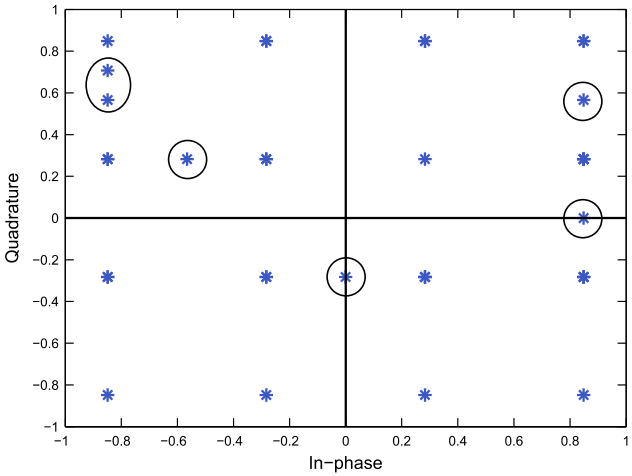
Although presented modeling techniques can achieve high simulation performance by increasing the levels of abstraction, several obstacles and limits of them and the resulting effects on the modeling approaches for functional verification have to be explained and discussed.

### 3.3.1 Sampling issues for event-driven models

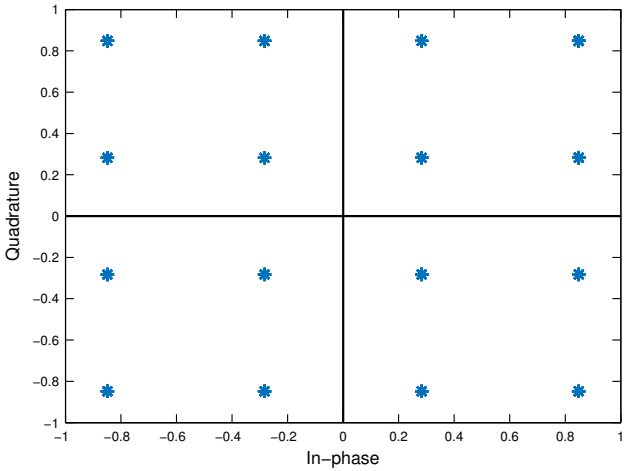
Event-driven models uses sample-based real value to represent analog signals. In order to reduce the approximation errors coming from the sampling process, the oversampling rate $N_{OSR}$ has to be around $N_{OSR} \geq 10 \cdot N_{Nyquist}$ according to section 3.2.2. This leads to the main issue of sample-based signal representation: the sampling rate of the RF signal for a communication system, e. g. $2.4GHz$ for Bluetooth, has to be set to $48GSamples/s$. If higher order harmonics have to be considered, the sampling rate has to be increase at least by the factor of three, in order to cover the third order nonlinear effects. Additionally, taking the frequency response of the RF filter into account demands even higher sampling rate. As a result, the computational effort required for the high oversampling ratio will significantly slow down the simulation and therefore exceeds the benefit of accuracy. For the functional verification, the details of the model including mentioned nonlinear and memory effect has to be chosen properly, in oder to trade the model accuracy against the simulation speed.

#### Post processing of sample-based signal

The event-driven simulator will only process and generate new data point, if the adjacent sample values are different. This mechanism results in less calculation and hence higher simulation performance. However, incorrect post processing and evaluation of the resulting data might lead to false results. As one can observe in Fig. 3.9(a), once the adjacent samples in In-phase path have the same values and in the Quadrature path not be the case, the interpolation mechanism during post processing will occur, so that the direct evaluation of the simulated 16-QAM signal without re-sampling results in unexpected/wrong constellation plot. Since this issue originates from the basic principle of event-driven simulation mechanism, it can only be avoided by re-sampling the signal under observation with its corresponding clock signal, as shown in Fig. 3.9(b).




(a) Constellation plot without re-sampling.




(b) Constellation plot after re-sampling the output signal.

Figure 3.9: Simulated constellation plots of a 16-QAM signal. The errors are highlighted in circles. The correct constellation can only be obtained by re-sampling the signal with the corresponding clock.

#### Discrete time amplitude noise

The standard deviation arising from the noise power that has to be realized using the white noise source can be calculated according to [Che05] as [8]

$$\sigma = \sqrt{4 \cdot k_B \cdot T \cdot R \cdot (10^{NF_{dB}/10} - 1) \cdot BW}, \tag{3.8}$$

which will end up with a white noise source having the defined voltage in the given bandwidth $BW$. The bandwidth $BW$ is used in the simulator to determine the update ratio of the random process that calculates the noise amplitude to be added to the input signal. Using an event-driven simulation approach, one is not necessarily interested in generating additional events due to this sampling process of the noise source, so the noise voltage is added at the sampling instances of the signal (assuming that the oversampling ratio is large enough). To keep identical noise voltages, the bandwidth $BW$ must be adjusted according the OSR of the input signal, which leads to

$$BW = \frac{1}{\Delta t}, \tag{3.9}$$

with $\Delta t$ being the time difference between two samples. However, there will always be a slight error in this bandwidth calculation due to the issue mentioned in section 3.3.1, since the time difference is a simple forward prediction. For the case of the verification approach, this can be neglected if a sufficient oversampling ratio is used.

#### Determine OSR for signals

A modern RF SoC has to operate at different frequency band with different communication standards. Therefore, the sampling rate of different signals during the simulation has to be calculated carefully. E. g. the event-driven passband model of a RF system operating at the Industrial, Scientific and Medical(ISM) band exhibits not only the oversampled RF signal in 2.4 $GHz$ range, but also baseband clocks and additional virtual clocks for the equivalent discrete-time filters. Here the OSRs of different signals must have a common integer denominator, otherwise, additional calculation due to the mismatch in the time step might occur, resulting in unwanted components in the frequency domain and hence in false simulation results.

#### Discrete time simulator limitations

The fixed OSR in a discrete time simulator leads to some problems concerning effects that require a high timing resolution, especially when simulating the phase

[8] $k_B$: Boltzmann constant $R$: Resistance

noise. The calculations from [HLL99]; [Abi06]; [Kun03] lead to the specification of the equivalent standard deviation $J$ of the time-domain jitter from the frequency domain phase noise $PN_{dB}$ as

$$J = \sigma_\tau = \sqrt{\sigma_\tau^2} = \sqrt{10^{\frac{PN_{dB}}{10}} \cdot \frac{\Delta f^2}{f_0^3}}. \tag{3.10}$$

While this only takes care of phase noise due to white noise sources, $1/f^3$ noise can be generated by adding a filter that adjusts the normalized delay $\Delta T_n$ between two transitions of an oscillating signal (assuming $f_c$ is the corner frequency of the $1/f^3$ noise) according to

$$\begin{aligned} j_n &= f(\sigma_\tau) \\ \Delta T_{n,1/f^3} &= j_n \cdot f_c \cdot \Delta T_{n-1} + \Delta T_{n-1,1/f^3} \\ \Delta T_n &= \frac{1}{2 \cdot f_0} + j_n + \Delta T_{n,1/f^3}, \end{aligned} \tag{3.11}$$

$j_n$ is the jitter results from $J$. $n-1$ and $n$ are the numbers of two sequent transitions. $f_0$ is the frequency of the noise free oscillating signal.

In both cases, the event-driven simulator must be capable to accurately display the time steps that occur due to the additional timing jitter. Assuming a probability density function(PDF) of

$$PDF_{Gauss}(\Delta\tau) = \frac{1}{\sqrt{2\pi} \cdot \sigma_\tau} \cdot e^{-\frac{\Delta\tau^2}{2\sigma_\tau^2}}, \tag{3.12}$$

the corresponding probability of the random timing jitter between $(i - \frac{1}{2}) \cdot \Delta t$ to $(i + \frac{1}{2}) \cdot \Delta t$:

$$P_i = \int_{(i-\frac{1}{2})\cdot\Delta t}^{(i+\frac{1}{2})\cdot\Delta t} PDF_{Gauss}(x)dx. \tag{3.13}$$

The resulting values at the discrete sampling points $i$ can now be interpreted as discrete distributed probability, so that the variance from $-N \cdot \sigma$ to $N \cdot \sigma$ can be calculated as

$$var_{\tau,Discrete}(\Delta t) = \sum_{i=\frac{-N\cdot\sigma}{\Delta t}}^{\frac{+N\cdot\sigma}{\Delta t}} i^2 \cdot P_i \cdot \Delta t^2. \tag{3.14}$$

The resulting phase noise can then be estimated from the quantized timing jitter

as

$$PN_{Discrete}(\Delta t, \Delta f) = 10 \cdot log\left(\frac{\sigma^2_{\tau,Discrete}(\Delta t) \cdot f_0^3}{\Delta f^2}\right)$$
$$= 20 \cdot log\left(\sigma_{\tau,Discrete}(\Delta t)\right) + 30 \cdot log(f_0) - 20 \cdot log(\Delta f). \quad (3.15)$$

Although mathematically proven, that for a very long simulation period, every mean value for the standard deviation of the timing jitter can be reached, it is obvious that there's a fundamental limit of usability for the time dependent simulations. In the author's experience, a jitter specification of up to 10 $fs$ can be modeled accurately enough with a quantized time step of 1 $fs$, which is the highest time resolution in current digital simulator [Cad11c]. Everything below this threshold is not feasible to simulate. E. g., GSM[9] has a phase noise specification of $-141dBC/Hz$ at 3 $MHz$ offset from its center frequency of 1.8 $GHz$ according to [3gp], resulting in an equivalent timing jitter of 3.5 $fs$ according to [WJ+09], which shows that the simulation limits are already reached. To permit the phase noise calculation from simulation to be in between a range of $\pm 3dB$, the jitter specifications shouldn't deviate more than

$$\frac{\Delta J}{J} = 0.414\ldots - 0.293 \quad (3.16)$$

which can be derived from equation 3.15. Still, it has to be mentioned, that verifying the non-ideal effect resulting from the timing jitter is only a part of the functional verification task. As the complexity and size of the system increases, alternative phase noise modeling approaches such as the one shown in [WCH10] have to be considered, in oder to overcome the limitation. In some cases, depending on the focus of the verification, the effect of phase noise has to be neglected for the sake of simulation speed.

#### Non-uniform sampling issue

As already mentioned in section 3.3.1 and 3.3.1, non-uniform sampling will occur, if an event-driven model is only trigged by its input signal. One way to capitalize this effect is the implementation of event-driven filter. Since the time in simulation of the discrete time kernel can only be increased (there's no iterative possibility like in the analog domain), the only possible method to do a discrete time filtering is using a Forward Euler approximation as mentioned in [Cot90]. E. g. a second order filter with the general transfer function:

$$H(s) = \frac{Y(s)}{X(s)} = \frac{b_0 + b_1 \cdot s + b_2 \cdot s^2}{a_0 + a_1 \cdot s + a_2 \cdot s^2} \quad (3.17)$$

[9] GSM stands for Global System for Mobile Communications

can be displayed in time domain with following differential equation:

$$a_0 y(t) + a_1 \dot{y}(t) + a_2 \ddot{y}(t) = b_0 x(t) + b_1 \dot{x}(t) + b_2 \ddot{x}(t). \tag{3.18}$$

Substituting $u(t) = \dot{y}(t)$, $v(t) = \dot{x}(t)$ and integrating on both sides leads to

$$a_0 u(t) + a_1 y(t) + a_2 \dot{y}(t) = b_0 v(t) + b_1 x(t) + b_2 \dot{x}(t). \tag{3.19}$$

With these auxiliary variables $u$ and $v$, an equivalent filter using a Forward Euler approximation can be realized as :

$$y_n = \frac{\dot{x_n} h_n b_2 + h_n x_n b_1 + h_n v_n b_0 + y_{n-1} a_2 - h_n u_n a_0}{h_n a_1 + a_2}. \tag{3.20}$$

Although this modeling approach for event-driven filter can provide sufficient accuracy and reduce reasonable amount of simulation time compare to discrete time filter implementation with fixed sampling rate, as shown in section 6.1.1. The Forward Euler method itself is an explicit integration method based on a truncated Taylor series expansion, which is very fast [KG95], but has very small numerical stability region [GST01]. So event-driven model implementation based on this method can only be applied to lower order filters with careful consideration of the maximum time step between two adjacent samples with different values during the whole simulation. Furthermore, according to [Joe08], not only the evaluation of non-uniform sampled simulation data is difficult due to the fact that the typical simulation tools don't offer high-level reconstruction algorithms, but also high computational power are required for such data to provide accurate Fourier transformations [Fre80]. Thus, the non-uniform sampling issue has to be handled with care for further model implementation. For reliable verification results, it's strongly recommended to avoid such kind of implementation.

### 3.3.2 Pin compatibility issue

For functional verification and consecutive debugging, it is important to be able to switch between different abstraction levels of the modeling "on the fly". This especially comes in handy, when being able to do a top level simulation with abstract models of the complete system, keeping only a very small portion of the design on transistor-level, for a fast verification prior to checking that part into the system database (Fig. 3.10).

The view switching itself is a well established method. For analog circuit design, designer often switch the views between the transistor- and layout/extracted[10]-level to verify the performance of their circuits. It can be achieved

[10]The term *extracted simulation* is often used for the circuit level simulation with back annotated layout parasitics. In this way, simulation can deliver more realistic results but also consumes more computational power.

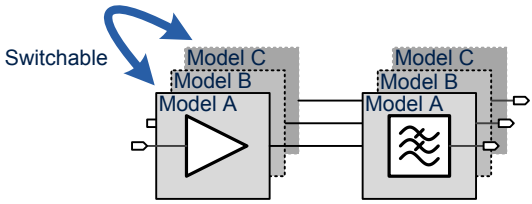



Figure 3.10: View switching as provided by the HED.

by using available tools in the design environment, such as Cadence® *hierarchy editor*(HED) [Cad12].

One of the main issues is to ensure pin compatibility while switching between different implementation variants of models. Using models with unmatched pin definitions or numbers leads to obscure and often false-friend verification results and is therefore strongly unrecommended. This is the main reason, that the baseband modeling approach is strongly limited by using VERILOG-AMS HDL. VERILOG-AMS doesn't provide any type constructs for the signal in its current language status. Fig. 3.11 illustrates this problem. While a passband signal uses the uses solely the values of $x(t)$ and $y(t)$ for the I/O, the baseband representation consists of at least three components to characterize the signal, each having a much lower frequency(consider the carrier frequency as a constant or slowly varying over time).

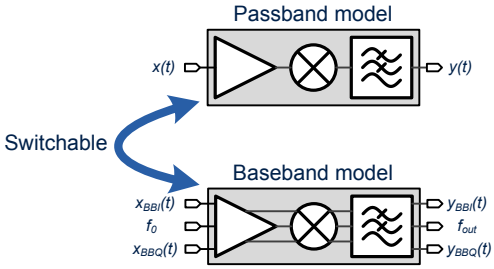



Figure 3.11: Connectivity representation for passband and baseband signals.

While one can overcome this issue by diverting differential signals from their intended use to carry I/Q signals [JH06], this method fails when trying to simulate with more than one signal at the same time as introduced in the section 3.2.3.

Additionally, this workaround is limited by the circuit design: once the circuit structure is single ended, the baseband modeling possibility by using available Verilog-AMS HDL feature will also end. Furthermore, using baseband models by diverting the original pin definition is often the primary source for unreliable verification results. This is where the need for new data type constructs and further extensions using pointer structures and dynamic arrays arises, in order to enhance simulation time while maintain the pin compatibility.

To the author's knowledge, there are currently two solutions to the aforementioned problem:

- **Extend current HDL:** The feature of Verilog HDL can be enhanced by implementing additional functions via the Verilog Procedural Interface(VPI) [DPR96]. In [Che09] and [Che10], the author implemented the aforementioned data type constructs for baseband signal via the VPI, it combines the real number and baseband modeling techniques in order to get the highest simulation performance. Still, such extension of the current HDL is often close sourced and only available for commercial usage. In the author's opinion, this kind of implementation doesn't offer enough transparency for reliable verification results. Moreover, it could increase the verification overhead in worst case, since the verification engineer has also to spend time to debug the HDL.

- **Use alternative HDLs:** Both SystemVerilog and VHDL offer the possibilities to implement new data type constructs as mentioned in section 3.1. However, the usage of the HDL is strongly depending on the design and verification flow. SystemVerilog is a still-maturing language and VHDL-AMS offers less compatibility for mixed-signal simulation compare to Verilog-AMS, which makes the overhead for using the language exceeds the benefit of its feature. Furthermore, recalling Fig. 3.1 and 3.4, even with the help of baseband modeling technique, the reachable abstraction level of classical HDLs are still limited. In contrary to the classical HDLs, SystemC fulfills all the mentioned requirements for baseband modeling with more implementation transparency and possibilities to reach higher abstraction level, but no direct link to the circuit level design. In order to benefit the feature offered by SystemC without losing the link to circuit level design, the consistency between models at different levels of abstraction has to be maintained. This topic will be discussed in detail in chapter 5

### 3.3.3 Connectivity and synchronization issues

#### Event-driven model in the presence of multiple drivers

This issue mainly concerns the event-driven models using `wreal` data type. The mentioned `wreal` is a discrete time data type, which is optimized for event-driven data flow simulations. However, it is not designated to model analog effects such as current summation at a single node, e. g. if two active `wreal` outputs are connected together, discipline resolution for `wreal` has to be determined in advance, in order to describe the signal condition in the presence of multiple drivers[11]. Otherwise, the simulation won't deliver proper results. It has to be kept in mind, that depending on the modeled output(voltage or current), different corresponding resolution functions has to be applied. There are two ways to define the discipline at one node, either globally or locally. Global definition is more well-arranged, but it will only accept one certain behavior for all `wreal` nets consisting multiple drivers. Local definitions can handle different model behavior at different node, but it will also cause more debug overhead due to multiple resolution functions.

#### Synchronization overhead between TDF and DE MoCs

As already presented in section 3.2.2, although both MoCs uses discrete time simulation mechanism, converter ports and exact synchronization semantics are required to connect both TDF and DE MoCs. This is due to the facts, that TDF MoCs run independently from the SystemC kernel and SystemC-AMS uses a layered synchronization concept to interact with SystemC's simulation kernel. Still, with fixed time step, discrete-time event from SystemC between the delta cycle will be neglected. Reducing the time step and hence increase the chance to catch the discrete event will again resulting in the slow down of simulation. A detailed evaluation of this issue can be found in section 6.1.1 and 6.1.3.

#### Cross domain connect module issue

Unlike the analog model, the event-driven model can't characterize frequency dependent load using impedance or Laplace transfer functions. An appropriate CM therefore has to be inserted to join the two different modeling domains. Currently available CMs use pure resistance and apply predefined delays to convert analog signals into discrete event and vice versa [JGH07], which is insufficient in some case. Assume the impedance is matched at the center frequency, gives us the possibility to do a sufficient system simulation in the band of interest. However, this doesn't take the critical impact of RF impedance matching into consideration and is therefore not sufficient for simulation with

[11] Please refer [Cad11e] for the supported resolution functions.

blockers or wide band signals. Fig. 3.12 illustrates the mentioned issue. Although VERILOG-AMS provides the possibility to automatically resolve the different discipline and insert CMs [FO00], the frequency dependency of the I/O impedance has to be considered in following cases.

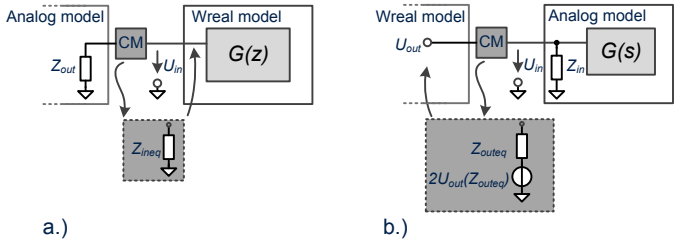



Figure 3.12: Proposed parameterizable CM with proper impedance. a.) From analog to event-driven domain. b.) From event-driven to analog domain.

From analog to digital domain (Fig. 3.12 a.)), the input of event-driven model works like an open clamp, thus it appears that the input voltage of the event-driven model is $U_{in} = 2U_{out}$, if both models are connected directly. The connect module should provide an equivalent load impedance $Z_{ineq}$ at the input wire of event-driven model. The built-in CM from Cadence® provide only a resistance $Z_{in}$ of 50 Ω per default, which is not sufficient to cover all the analog to event-driven cross domain connection possibilities(mostly, $Z_{in}$ and $Z_{out}$ are frequency dependent). E. g. A **`wreal`** wire can only map the value of voltage or current from an analog model, resulting in information loss, so that a proper input or output impedance cannot be clearly defined in the event-driven model. If there is a cross domain connection between an event-driven and an analog model(Fig. 3.12 b.)), the output of an event-driven model works like an ideal voltage source. In this case $U_{out}$ and $U_{in}$ at the interface of the adjacent models are always the same. A change in the voltage characteristic cannot be detected. The CM hereby have to convert the output voltage of the event-driven model in an equivalent, non-ideal voltage source as shown in Fig. 3.12 b.). For a parameterizable CM, the output voltage of the event-driven model should be converted to $2 \cdot U_{out}(Z_{outeq})$. At the moment, two steps are required for implementing a proper CM: first, the CM it self has to be realized as shown in listing 3.1. Second, the connect rules has to be defined as shown in listing 3.2.

```
1 discipline electrical_zin
2 domain continuous;
```

```
potential Voltage;
flow Current;
enddiscipline

connectmodule E2wreal(Ain, Dout);
input Ain;
electrical_zin Ain;
wreal Dout;
real Dreg;
assign Dout = Dreg;
...
always @(absdelta(V(Ain), vdelta, ttol, vtol))
    Dreg = V(Ain);
analog begin
//parameterizable impedance: Zin_num and Zin_denom are based on block specs!
V(Ain)<+laplace_nd(I(Ain),{Zin_num}, {Zin_denom});
end
endmodule
```

Listing 3.1: User-defined electrical to `wreal` connect module example.

Clearly, this method is limited by the numbers of cross domain interfaces with different complex impedances at the top level schematic and their complexities. Therefore, such method is not feasible for top-level verification due to its high time expense and error-proneness. It should only be used, if the impedance effect at the cross domain interface has to be considered during the verification, e. g. during a mixed-level simulation.

```
connectrules user_def_wreal;
electrical,electrical_zin resolveto electrical;
endconnectrules
```

Listing 3.2: User-defined electrical to `wreal` connect rule example.

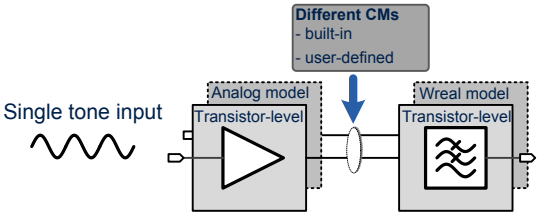



Figure 3.13: Testbench with switchable levels of abstraction and different CMs.

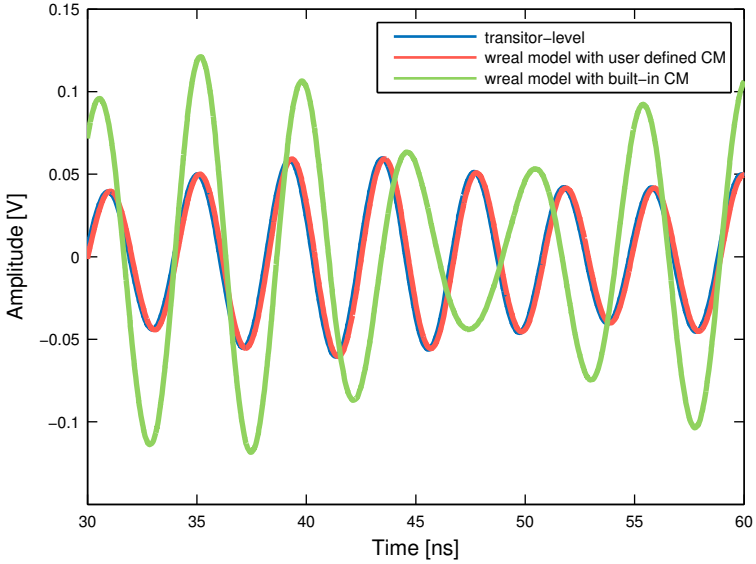



Figure 3.14: Transient simulation results using different cross domain CMs.

In order to demonstrate the mentioned issue resulting from CM, a simple testbench consisting only of an amplifier and a matching network has been build up, as shown in Fig. 3.13. Firstly, the whole testbench is simulated at transistor-level in order to get the reference result. After that, the amplifier is kept at the transistor-level, the filter however, has been switched to its event-driven model implementation. Here, both user-defined and built-in CMs has been applied to the cross domain interface subsequently for the comparison. The user-defined CM uses a Laplace transfer function to model the impedance effect at the filter input, while the built-in CM consisting only a 50 Ω resistance.

Fig. 3.14 shows the clipped result of the transient simulations by applying a sinusoidal signal at the input of the amplifier. The transient responses show significant difference between different CMs: While the user-defined CM with the correct impedance implementation showing good agreement with the reference result at transistor-level. The result with built-in CM shows a completely different response, which clearly will lead to false verification results.

Furthermore, applying cross domain CMs during the simulation will definitely lead to higher mixed-signal synchronization overhead and hence lower simulation performance as shown in section 2.2.3. It is therefore important to firstly partition the whole system properly and build up a corresponding modeling plan, so that the number of CMs can be reduced at minimum, or even avoided completely.

# 4

# A Verilog-AMS Based Modeling and Verification Methodology for RF Mixed-Signal SoCs

In order to meet performance specification in the design of integrated RF/MS blocks in a RF MS SoC, high degree of skills and expertise gained though experience are required. Since the chip size of today's wireless SoCs are increasing drastically while technology nodes continue shrinking, the challenge of analog RF circuit design is not only the performance requirement, but also the increasing functional complexity within of them. Moreover, severe functional failures might result in silicons, that might cause the chip not to be powered up. Such errors have to be detected and corrected prior to tapeout, otherwise, no performance evaluation is possible, forcing the whole project time frame to extend, which will lead to significant amount loss of financials. Thus, advanced verification methodologies which closely interacts with the design flow and seamlessly integrates into the existing tool link, are critical in successfully delivering functional silicons, while meeting the increasingly tight time-to-market constraints.

## 4.1 Model implementation and functional verification flow

### 4.1.1 Available methods

However, applying modeling technique alone is not enough to overcome the mentioned issue. Besides running fast, a more formalized verification flow has to be developed, part of this flow is in charge to control the model implementation and refinement process. Fig. 4.1 depicts the verification flow with strong focus

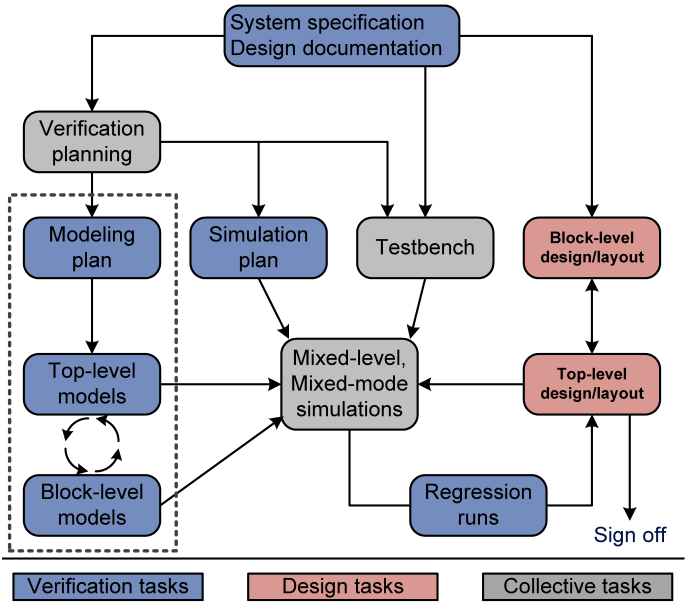



Figure 4.1: Overview of the verification flow with strong focus on analog/RF subsystems, adapted from [CK07]. The dashed line highlights the critical path for model generation.

on analog/RF subsystems proposed from [CK07]. The functional verification flow described here serves as an addition to the existing verification techniques such as timing analysis, DRC, LvS and other advanced digital verifications.

Based on the system specification and design documentation, the verification team will start to develop the corresponding top-down models based on the proper verification plan, while the analog team starts to design the system at block level. Both teams have to cooperate at very early design stage to develop the verification plan and testbench according to the design documentation. At this point, the testbench will still be kept very general with small discrepancies between both teams due to different focuses: analog design team will focus on the performance evaluation, while the verification focusing on the stimuli development and functional check for the circuits. Later on, as the design hierarchy moves to the top-level, different simulation tasks based on the simulation plan are executed. Regression runs will help both teams to increase the verification coverage and

hence to keep the project proceed as scheduled and to avoid any regressions. At the end, sign-off runs consisting top-level model are required in order to achieve a reasonable degree of functional confidence prior tapeout.

Although the tasks shown in Fig. 4.1 of both teams are clearly defined and described in [CK07] and the flow seems to be straight forward. Certain details and the resulting additional overheads of the flow has to be discussed further. Especially the model generation path, which is substantial and critical for the verification in the author's opinion.

**Model implementation flow:** Traditional top-down analog/RF design applying modeling techniques to speed up the whole design process. However, there is still no "formal" model refinement method for the functional verification. This is due to the fact, that there is no general or mature synthesize method for analog circuits based on higher levels of description. So there is no automated way either to generate analog circuits based on higher levels of description nor to refine the models based on the circuit. Hence, models for analog/RF blocks has to be hand crafted. there is always a point in the verification flow, where the top-down models has to be refined in order to map more circuit-level information such as pin description, I/O extension or functional details. One might argue, that bottom-up model generation can avoid such extensive model refinement process. However, if models are created firstly after design process is advanced in certain stages, the model implementation overhead will be much larger, so that it exceeds the time benefit of using models for block integration. Additionally, even the bottom-up modeling cannot prevent inevitable design changes at some point. Thus, model implementation has to be considered as an iterative process with large amount of refinement overhead.

In contrary to the statement in [CK07], where only ***one*** top-level model is required, which

> *...serves as the "sign-off" quality executable specification delivered to the integrator of the functional unit.*

Top-level models with different abstraction has to be implemented to support the functional verification. Due to the increasing complexity of the RF front-end, one model can neither cover all the required functional details nor the certain performance metrics. Thus, top-level models at different levels of abstraction are required for the sign-off simulations containing different verification aspects such as start sequence, connectivity check, demodulation algorithm or just data integrity.

**Testbench and assertion implementation** Testbench is the essential part for simulation-based verification, its components has been described in section 2.2.

In addition to the testbench, assertions have to be implemented for the stimuli and output monitor, so that part of the functional errors or even errors in the testbench due to misinterpreted specifications can be detected without large amount of visual inspections. Furthermore, model validation and refinement rely on the testbench, a significant amount of model parameters can only be extracted from extensive simulation results at transistor-level. Besides that, regression tests require automated execution of testbenches.

**Model refinement process** As states above, analog/RF models are mostly handcrafted. Although the modeling and the respective refinement process is an iterative and time-consuming process, changing models manually will lead to more chance of coding error and hence to false verification results. Thus, an automated model refinement flow is inevitable to keep the manual interactions to a minimum.

Based on the mentioned details, it can be seen that the functional verification for RF MS SoC is neither a stand alone process nor a pure model generation flow. It is a process consistently accompanying and supporting the design, which has to be seamlessly integrated into the current mixed-signal design tool link and flow. In order to complete and improve the available method, a hierarchical verification methodology is proposed.

### 4.1.2 Proposed hierarchical verification methodology

Both terms ***level of abstraction*** and ***hierarchy*** are in focus of the proposed methodology. In addition to the fundamental requirements of the methodology described in section 2.4, it must be mentioned here, that the intention of proposed method and flow is not trying to completely neglect manual interactions. On the contrary, it requires even higher workload of the verification team at the beginning of the design process, in order to keep the flow as automated and formalized as possible. Fig. 4.2 illustrates the proposed flow as an extension of the critical model generation path pointed out in Fig. 4.1. The key concepts of the proposed methodology will be described in the following sections.

## 4.2 Key concepts for hierarchical verification

### 4.2.1 Verification planning

A comprehensive verification plan contains a wide range of topics based on the system specification and design documentation shown in Fig. 4.1. It serves as the initial point of the verification flow. For analog/RF part of the system, besides meeting the tapeout schedule, a verification plan has to account for all

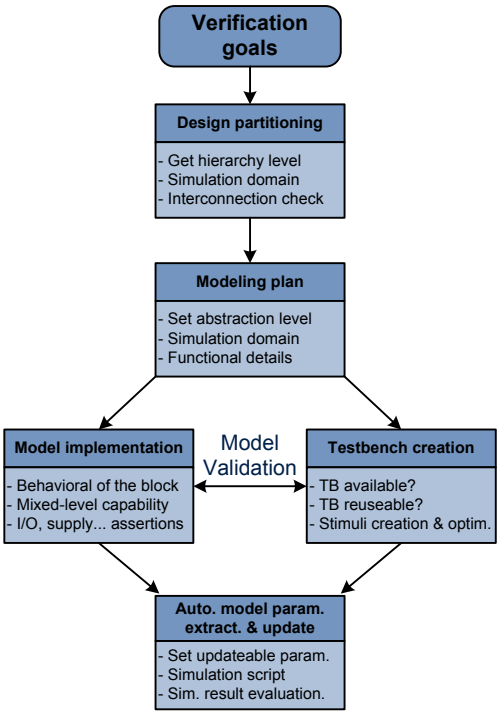



Figure 4.2: Key concepts for proposed hierarchical verification methodology.

architectural features and ensure the test cases can cover its full functionality. Starting from the system specification and design documentation, the analog and digital top-level interface, configuration buses and chip I/O are defined. During the planing phase, critical signal paths in the design can be identified by both teams, one possible candidate is e. g. the signal path of the RF front-end. In this case, all connections and the signal flow from the top-level down to each single block has to be the main concern or ***goal*** of the functional verification. In conjunction with that, the simulation and modeling strategies can be derived. Furthermore, the verification plan is important to track the actual status of the verification process and in charge to schedule the available man power for the verification goal. For the proposed flow, the main work for the verification team at the start consists not only the model implementation, but also the translation of the testbenches. Here, testbenches inside the analog design environment are commonly GUI-based[1], so they have to be translated to their equivalent script based version, in order to execute them unattended in the later design stage.

### 4.2.2 Hierarchical design partitioning

For hierarchical verification, the main target is to facilitate the full-chip simulation in a reasonable time frame. Thus, only using one testbench or one model for the whole system can neither justify the mentioned proposition, nor it is feasible enough to bring us close to the verification goals in time. The only possibility at the moment is to apply hierarchical decomposition to the whole system recursively, so it can be divided into small and hence manageable parts, as states in section 2.4. Once the system is partitioned into reasonable size, the simulation performance will increase and this results in the increasing verification coverage. However, since the verification coverage is also depending on the chosen level of abstraction, there will always be a trade-off between the levels of abstraction through the hierarchy and the coverage. On the one hand, the hierarchy level of a certain design part, to which models are applied, has to be determined. On the other hand, the functional and behavioral details of the applied model has to be chosen. Again, due to the hierarchical partitioning, there won't be one model for all test cases, but rather a set of models with different grade of details focusing on different verification goals.

For a single receiver front-end shown in Fig. 4.3, starting from the top-level, the whole design can be firstly divided into analog, digital, mixed-signal, power and equipment parts according to the system specification. Although the boundary between the analog and mixed-signal part is slowly disappearing (digital part will take more and more control and configuration of analog parts), it is still valid to define the partition as analog, as long as the main signal processing

[1] In contrary to digital design flow, which is mostly command line based. The analog design flow uses a lot of GUI based tools.

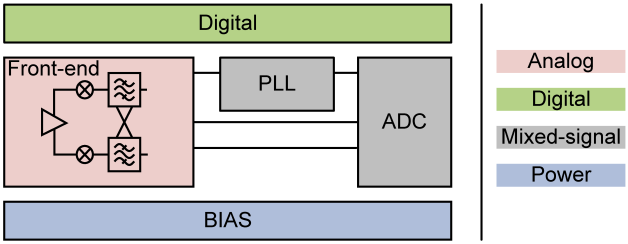



Figure 4.3: Design partitioning of a generic RF receiver front-end. The mentioned equipment part is regarded as component of the testbench, hence it is not shown here.

path is done in analog domain. In this case, the RF front-end is regarded as the analog part of the receiver, whereas the PLL and ADC are considered as the mixed-signal parts. For further hierarchical partitioning, following factors have to be taken into consideration:

- *Complexity of the block* - The complexity of each single block inside one partition have to be kept in a reasonable range. Otherwise, proper behavior implementation of the model will consider too many design variables. On one hand, model validation might take too much simulation time for a complex block; On the other hand, complex models tends to slow down the simulation. However, if the complexity and hierarchy of the partition is kept too low, lots of simple models are required, which in turn requires higher amount of testbenches and hence increasing the model maintenance overhead.
- *Model abstraction applied to the hierarchy level* - Higher abstraction level for a partition at certain hierarchy level has significant influence on the simulation performance and accuracy. Besides running fast, critical details of the blocks' behavior has to be considered in the model. Hence, simulation speed has to be traded against the grade of details in the models and the resulting accuracy decrease.
- *Signal integrity and block integration* - As mentioned in section 3.3.3, too many cross domain synchronization will also slow down the simulation, even at higher levels of abstraction. Still, using switchable analog and event-driven models based on common partitioning of the design is inevitable. This is the only way to keep the consistency between models at different levels of abstraction in a hierarchy. Consistent models lead to better signal integrity check and faster block integration.

As one can see, there is no straight forward guideline for determining the design partitioning and its corresponding levels of abstraction. For complex systems, the verification plan have to define the general partitioning directives with the consideration of simulation speed, end-to-end signal integrity and orders of non ideal effects described in the model.

### 4.2.3 Hierarchical model implementation

Fig. 4.4 shows the proposed model generation flow for the analog/RF part. To allow proper verification of the SoC, at least all functional units need to be modeled for full-chip simulation. Behavioral models can be considered as a higher level of abstraction for the transistor-level AMS design. The main reason of using models at different abstraction levels within the whole design hierarchy, either during top-down design or by bottom-up verification, is to allow a full-chip/overall-system simulation in a reasonable time frame. While significant work on RF modeling exists, most of them focus either on fast behavioral models supporting system design [SFB05a]; [MSPM03]; [HY+06], or models supporting additional noise and spectral analysis to accelerate the circuit design [RWT03]; [ZWB03]; [SN08]. Model implementation for functional verification however, does not necessarily require circuits' performance metrics, while simulation performance is the main concern. Thus the only non ideal effects included in models target functions verification are those for which a functional compensation algorithm is designed for. Such compensation algorithm can be implemented in the digital signal processing unit, or a mixed-signal calibration loop. Still, the detailed targets of applying models in certain design phase can still vary: In [MMS07], models has been used to reduce the complexity of analog blocks and to accurately map their critical performance characteristics in order to evaluate the overall performance of the SoC from a system-centric perspective. In [Che09], pin-compatible models at high abstraction level are deployed, mainly targeting to verify the functionality of the whole SoC.

Models at different abstraction levels serve different purpose. Pure analog models written in Verilog-A allow the side-by-side simulation of behavioral models with the transistor-level circuit blocks. Hence, they are the first choice for mixed-level simulation, helping the designer to implement sub system and estimate its performance at lower hierarchy level. Additionally, analog models can help the modeling/verification engineers to communicate with circuit designer, in order to align the model accuracy and to find out bugs in the model. However, study [WJ+09] shows, that the simulation performance of analog models is not sufficient as the design hierarchy and thus the system complexity moves up. Still, the analog models can serve as reference for further abstraction levels. The wreal/real number models for the analog blocks push the whole simulation into the digital domain. Since the higher abstraction of the wreal models significantly

reduces the computational effort, they are implemented targeting the high simulation efficiency for the final top-level verification of the whole chip. Digital subsystems require less modeling work, since the modeling task for all hierarchy and abstraction levels is done during the design flow (functional, RTL, gate level), as shown in Fig. 2.14 and Fig. 3.4.

The usage of wreal signal in Verilog-AMS can significantly increase simulation efficiency. But the wreal data type allows only to transfer one real number, in one direction all the time [Cad11c], which limits the usage of equivalent baseband model to get higher abstraction. Hence, for the proposed modeling and verification methodology, passband real number models are regarded as the highest level of abstraction in the design environment and will be used for the top-level sign off due to this tool constrain.

Nevertheless, as Fig. 4.4 depicts, the implementation of sign off quality models is not straightforward, but demands comprehensive iteration and validation process. In addition to that, critical model parameters have to be determined for the model refinement.

Basically, the top-down models can be used to describe the system specification at the top-level, so it can be executed in the simulation environment. However, starting from the block-level of the design, the top-down model require consistently refinement, sometimes, even a complete rework of the model is inevitable due to radical design changes. Hence, at the start of a design project, the top-down model serves only as alternative specification document, whereas lots of additional models at lower block-level have to be implemented according to the circuit design status in a bottom-up manner.

### Coding style

Starting from the first model of the project, it has to be kept in mind, that sustainable source code is the main factor for the model refinement and reliable verification result. Debugging and validating foreign models can be a very time-consuming task, if the source code of the model is too obscure. Therefore, a strict and unified coding style is preferred, the format of the code have to be distributed in the verification team. Since a large part of the analog/RF models are handcrafted, proper coding style and guideline is also for the model reuse and validation though other team members. In addition to the mentioned benefits, model refinement process can also profit from a formalized coding style.

**General model structure** Based on the mentioned requirement, the general code structure has been determined and shown in listing 4.1.

```
1  // General VAMS model code structure
2
3  `include "constants.vams"
```

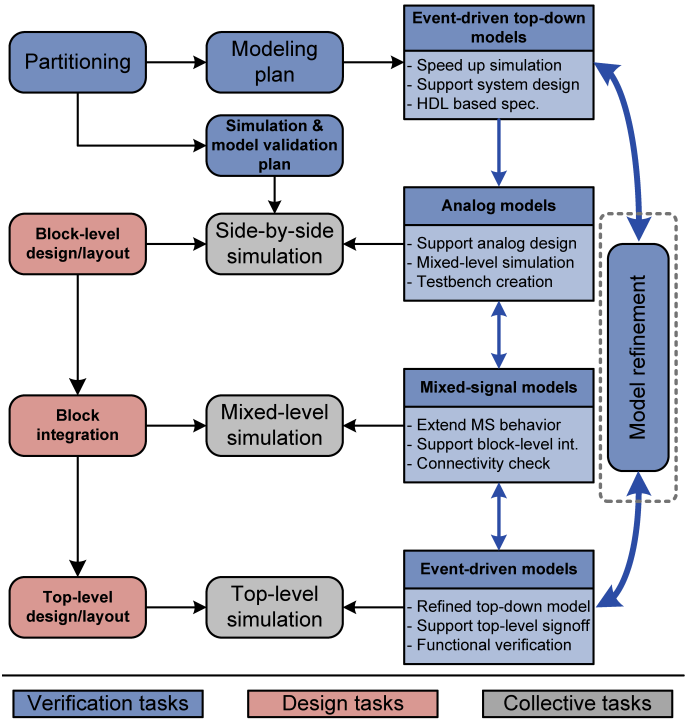



Figure 4.4: Detailed illustration of proposed model implementation flow. As one can observe, the model generation is not a straightforward process, but more a consistently iterating task. The three highlighted collective tasks require tight cooperation between the design and verification team.

```
4   `include "disciplines.vams"
5   `define SUPPLYCHECK 1
6
7   module model_name( module I/O )
8   ////////////////////////////////////////////////////////////////////////
9   // SIGNAL ports
10  ...
11  ////////////////////////////////////////////////////////////////////////
12  // BIAS ports
13  ...
14  ////////////////////////////////////////////////////////////////////////
15  // PARAMETER definition
16  ...
17  ////////////////////////////////////////////////////////////////////////
18  // VARIABLE definition
19  ...
20  ////////////////////////////////////////////////////////////////////////
21  // ANALOG BEHAVIOR
22  analog begin ... end
23  ////////////////////////////////////////////////////////////////////////
24  // MS BEHAVIOR
25  always @(...) begin ... end
26  ...
```

Listing 4.1: General model structure.

**Naming conventions** Besides a consistent model structure, strict naming convention is inevitable, for debugging and automated model parameter update. Table 4.1 shows the naming conventions proposed by [Sch11].

### Requirements for the modeling task

The main target of analog/RF model implementation is to assist the functional verification at top-level and gain simulation performance by increasing the level of abstraction and exploiting the simulation speed of a pure digital simulator. Following criteria must be met, in order to rise the abstraction level without loosing the functional accuracy of a model:

**Portability** Besides exhibiting high simulation performance, the handcrafted models must be capable of working both analog and digital design environments, so that both analog and digital design teams are able to run the models for their own simulation. Additionally, models capable in both simulation domain can also serve as a kind of communication medium between both design teams.

| Pre-/postfix | Type | Nature | Example | Description |
|---|---|---|---|---|
| p_ | parameter | - | p_ppf_gain | Gain in dB |
| r_ | real | - | r_gain_dec | Gain in decimal factor |
| v_ | real | voltage | v_out_i | Output voltage of I path |
| i_ | real | current | i_in_q | Input current of Q path |
| pwr_ | real | power | pwr_noise | Noise power at output port |
| int_ | integer | - | int_counter | Integer counter |
| f_ | integer | file identifier | f_inputfile | File Identifier for input file |
| s_ | integer/reg | supply check | s_supply_inst | Instance's supply check - Set to "1" when proper supply is detected |
| ..._apx | - | - | p_inst_gain_apx | gain parameter for automated refinement |
| d_ | reg | - | d_out | Digital output signals |

Table 4.1: Naming conventions for parameters and variables in Verilog-AMS.

**Compatibility** Models at lower abstraction level must be fully compatible with the analog simulator due to the four following reasons:

- Handcrafted models of analog/RF blocks must fully support the analog simulator. Only in this way, model validation can be done by executing side-by-side simulation of the model and schematic. In the analog design environment, lots of parameters cannot be determined by simple transient simulation. Therefore, misalignment between the model and schematic can only be exposed by applying advanced simulation techniques, as mentioned in section 2.3.1. Although most analog simulation technique are targeting the performance evaluation, the chance of fully debugging the analog model aside from the schematic is only given at this point. Here, Verilog-A models offer the full analog compatibility, which will also serve as the reference model for further abstraction.
- At lower design hierarchy, analog designer proses the best know-how and experience. Hence, analog models can be validated even faster.
- Mixed-level simulation with analog models can speed up the block integration, since analog models exhibit higher simulation performance compare to transistor-level. Also, models can be validated again at higher design hierarchy by using them for the block integration.
- As stated in [Che09], model based testbench creation can also speed up the performance evaluation of the analog circuits.

**Rules for verification engineers** Since handcrafting the analog/RF model is a daunting and error-prone task, the verification team has to obey the following rules, in order to keep the project in track:

- **Schematic cannot be touched!** This is one of the most important rules for a successful design flow. The task of verification team is to support the design and facilitate comprehensive verification. As soon as the verification team starts to change the designed schematic(due to model complexity or unfitting signal routing) the project will fail. Any change in design has to be discussed in detail, since this change will also leads to the reiteration of the model implementation process and eventually delay the schedule.
- **Keep the change in track** As already mentioned, any changes in the design or model will result in new iteration of model generation and validation process. If design change cannot be caught, verification will fail.
- **Focus on the signal path** One of the most important part of the design is the signal path and the signal processing blocks. For the full chip verification, the complete end-to-end signal path has to be verified in detail.

### Hierarchical modeling

So far, only the analog model implementation part of Fig. 4.4 has been discussed. As design hierarchy moves up, complexity of the system also increases. Here, only applying analog model is not feasible, since more mixed-signal functionalities and digital controls comes into the design. Therefore, the existing analog models has to be refined to map the mixed-signal behavioral. The only mixed-signal simulation technique applicable here is transient. Block-level integration will lead to another simulation slow down, so the full chip verification cannot deliver the required coverage. Hence, the analog behavior in mixed-signal models has to be abstracted further for macro-level (s. section 2.4.1) integration. Mixed-signal simulation at transistor-level for model validation at complex block-level, not to mention the macro level, will consume too much time and computational power. Therefore, analog and the top-down models will serve as reference for further abstraction and refinement. At this point both event-driven model set from the top-down and the bottom-up modeling will be aligned. One of the benefits for using consistent and strict coding style will emerge: since the most analog behavior has to be adapted to event-driven real number models (s. section 3.2.2), the sustainable code will result in a short adaption time.

One might start questioning the sign off quality of the highly abstract models, since the event-driven ones are derived from the analog models and lots of information regarding the performance are neglected at top-level for the sake of simulation speed. Firstly, since the whole model refinement process is in

an iterative manner, especially the analog model has been through a very comprehensive validation process, i. e. they are commonly aligned with the design on the basis of side-by-side simulation with the actual designs. Therefore, analog models are capable of serving as reference for further model abstraction. Secondly, since the functional verification mainly targets to find functional errors, which is more crucial for the design at advanced technology nodes. The verification team has to trust analog designers to deliver the right performance, especially at higher level of the design hierarchy.

### 4.2.4 Assertion and debug concepts

Debug concepts and assertions(s. section 2.2.5) have to be considered during the model implementation flow, since they are inevitable components for the functional verification.

**Debug macro** During the model validation phase, embedded debug macro in the model will significantly rise the clarity of the error source. Depending on the actual model implementation, different internal details of the model will be printed to the simulation logs. E. g. if the debug mode is enabled, a filter model, which gets its coefficients via an external database should strobe out the file path, file status and the coefficients from the database. In this way, some of the frequent error source can be detected untimely. Since debug macros will cause additional simulation overhead, it call be gradually grouped into different levels depending on the model's complexity.

```
1 `define display_coefficients_dig(az_R, bz_R, az_Q, bz_Q, j, order) \
2     $display("          %12s   %12s   %12s   %12s", "az_R", "bz_R", "az_Q", "bz_Q"); \
3     j = order;                                    \
4     for ( j = order; j >= 0; j = j - 1  )                    \
5     begin                                         \
6         $display("     [%d]  %12.64e   %12.64e   %12.64e   %12.64e", j, az_R[j], ↩
          ↪ bz_R[j], az_Q[j], bz_Q[j]); \
7     end
```

Listing 4.2: Debug macro for filter coefficients.

**Assertion implementation** VERILOG-A doesn't support the assertion language such as PSL(Property Specification Language) or SVA(SystemVerilog Assertions)[2]. Therefore, the assertions in the analog models are realized as external modules, which will be instantiated in the model if required. The main aspect of the assertions in both analog and event-driven models is to ensure the signals

[2]Both are standardized and well established digital assertion-based verification languages.

under surveillance are in the specified range and the correct interconnection between the blocks are given.

```
module voltage_check (vdd,gnd,assert_result); // module used to check supplies

input vdd, gnd;
output assert_result;

electrical vdd, gnd;
reg assert_result ;

parameter real p_vcheck_range = 0.1; //define the voltage uncertainty range in percent of Vmin or Vmax
(* desc="min. voltage to check", units="V" *)
parameter real p_vcheck_min_voltage = 0.2 from (-inf : inf);
(* desc="max. voltage to check", units="V" *)
parameter real p_vcheck_max_voltage = 1.2 from (p_vcheck_min_voltage : inf);

always @(above(V(vdd, gnd) - (p_vcheck_max_voltage*(1.0+p_vcheck_range))) begin // V > max?
    assert_result = 1'b0;
    `ifdef __DEBUG__
        $strobe("%M: @ %3.1f: maximum voltage exceeded, vdd: %3.1f, assert_result : %1b", $abstime, V(vdd, gnd), assert_result);
    `endif
end

always @(above(V(vdd,gnd)-0.5*(p_vcheck_max_voltage-p_vcheck_min_voltage)) begin //min < V < max?
    if (p_vcheck_max_voltage*(1+p_vcheck_range)>V(vdd,gnd) && V(vdd,gnd)>p_vcheck_max_voltage*(1-p_vcheck_range)) begin
      assert_result = 1'b1;
      `ifdef __DEBUG__
          $strobe("%M: @ %3.1f: vdd: %3.1f, assert_result : %1b", $abstime, V(vdd,gnd), assert_result);
      `endif
    end else begin
      assert_result = 1'b0;
      `ifdef __DEBUG__
          $strobe("%M: @ %3.1f: vdd: %3.1f, assert_result : %1b , please check the voltage", $abstime, V(vdd,gnd), assert_result);
      `endif
    end
end

always @(above(p_vcheck_min_voltage*(1.0-p_vcheck_range)-V(vdd,gnd))) begin // V < min
        assert_result = 1'b0;
    `ifdef __DEBUG__
        $strobe("%M: @ %3.1f: voltage too low, check connection first, vdd: %3.1f, assert_result : %1b", $abstime,V(vdd,gnd) , assert_result);
```

```
40          $strobe("%M:␣max:␣%3.1f,␣min␣%3.1f", p_vcheck_max_voltage, p_vcheck_min_voltage);
41      `endif
42  end
43
44  endmodule //voltage_check
```

Listing 4.3: Assertion model for voltage check.

```
44  `include "voltage_check.vams"
45  `define __DEBUG__
46  `define VOLTAGE_CHECK
47  module testmodel(v1, v2...);
48  //IO, port definitions
49  ...
50  electrical v1, v2;
51  logic v_check;
52
53  parameter real p_testmodel_max_voltage=1.0;
54  parameter real p_testmodel_min_voltage=0.1;
55
56  'ifdef VOLTAGE_CHECK
57    voltage_check #(.p_vcheck_min_voltage(p_testmodel_min_voltage),
58                    .p_vcheck_max_voltage(p_testmodel_max_voltage))
59                    voltage_check(.vdd(v1),.gnd(v2),.assert_result(v_check));
60  'endif
61   analog begin
62     ...
63   end
64  endmodule
```

Listing 4.4: Instanciating assertion module in a mixed-signal model.

In addition to that, since the assertions will continuously monitor the signal, it will cause a significant simulation overhead. Using conditional compiler directives to control the assertions is therefore necessary. Listing 4.3 shows an assertion module for monitoring the voltage between two ports, a deviation of $\pm 10\%$ is given for uncertainties. The debug output is activated at line 45 and the assertion is set to active at line 46, via compile directives. The instantiation of assertion is done in listing 4.4 at line 57. Alternatively, analog assertions can also be realized via macros in Verilog-AMS as proposed in [Sch11].

For mixed-signal or event-driven models in Verilog-AMS, PSL assertions can be applied to them according to [JPB10]. However, analog value, real number or wreal expressions have to be converted into Boolean or clock expressions, e. g. instead using external assertion module in listing 4.3 at line 57, following PSL assertion code can be used:

```
//psl assert (V(v1, v2) < 1.1*p_testmodel_max_voltage) && (V(v1, v2) > 0.9*p_testmodel_max_voltage) @(above(V(v1,v2)-0.5*(p_testmodel_max_voltage-p_testmodel_min_voltage)))

//psl assert (V(v1, v2) < 1.1*p_testmodel_min_voltage) && (V(v1, v2) > 0.9*p_testmodel_min_voltage) @(above(0.5*(p_testmodel_max_voltage-p_testmodel_min_voltage)-V(v1,v2)))
```

Listing 4.5: PSL assertion for voltage check in mixed-signal model.

The main advantage of using a well established assertion-based verification language such as PSL or SVA consists of the fact, that the tool support in this case is much advanced compare to the analog assertion macro, which generally relies on `if(...)then $strobe 'message'` approach. For PSL assertions, advanced digital verification tools, such as assertion browser or coverage estimation are already available, which will increase the reliability of functional verification at top-level.

Additionally, the assertion unit can be built in the testbench outside the module and hierarchically access different analog nodes by using the system task `$cds_get_analog_value(...)`, whereas the analog assertion macro is still a part of the model, which might interfere with the simulation, e. g. causing hidden state.

```
module hidden(out, in);
//IO, port definitions
electrical out, in;

parameter real p_hidden_thresh=0.5;
parameter real p_hidden_ttime=1p;
real r_tmpout;

if (analysis("pss")) begin
    @(cross(V(in)-p_hidden_thresh) ; //place a time step by creating a empty statement
    V(out) <+ transition((V(in)-p_hidden_thresh)), 0 , p_hidden_ttime);
end
else begin //if only non steady state analysis
    @(cross(V(in)-p_hidden_thresh) r_tmpout = V(in)-p_hidden_thresh;
    V(out) <+ transition(r_tmpout, 0, p_hidden_ttime);
end
```

Listing 4.6: Code example to avoid hidden state.

### 4.2.5 Automated parameter extraction and update

Hierarchical modeling leads to models at various levels of abstraction through the whole hierarchy. Model maintenance and refinement at later project phase will consume remarkable time and resource. Additionally, the lower levels of abstraction is chosen, the more parameters are required to accurately map the

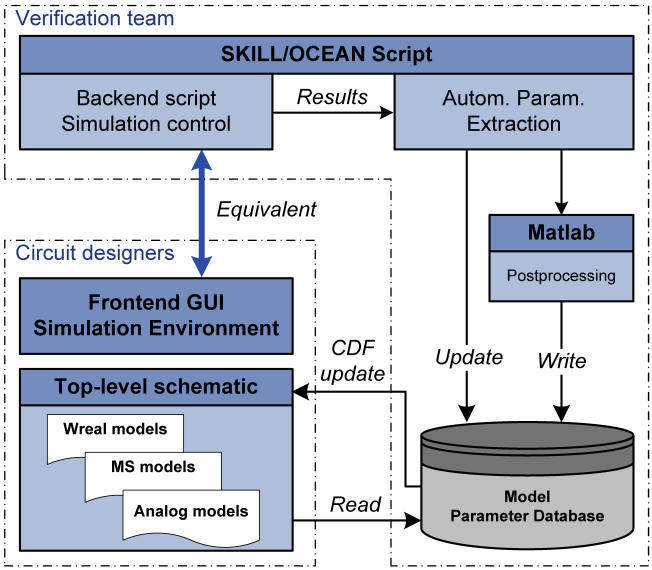



Figure 4.5: Overview of the proposed automated parameter extraction and update flow.

behavior of the circuit. In some cases, a model with high detail grade require hundreds of parameters. Since the models up-to-dateness will also significantly affect the verification result, its parameters has also to be synchronized with the current design version. Any changes in design will lead to a rework of the model or rearrangement of the parameter values. Manually updating the models of tracking the design changes is clearly unfeasible. Hence, for the sake of the project's success, an automated model parameter extraction and updating process is proposed.

Fig. 4.5 shows the basic architecture and tasks for the model parameter extraction and update. First it executes the testbenches, provided by the circuit designer as schematic. All simulation runs unattended, so that required parameter or results can be extracted on a daily schedule. Later on, if further processing steps of the simulation results are required in order to get the model parameter, an additional MATLAB® routine for post processing will be executed. The obtained results and parameters will be stored in an model parameter database. Part of the model parameters can be updated and stored through the CDF(component description format) parameter in the corresponding schematic. Other parameter

in the database can be accessed by the models dynamically during the simulation. The automation is based on the heavy usage of SKILL/OCEAN(Open Command Environment for Analysis) language, which will be briefly introduced in the following.

### Brief introduction to SKILL

SKILL is a high-level, interactive programming language based on the artificial intelligence language, LISP [Bar90]. It is designed to support command entry, access the design database and user interface and procedural customization in Cadence Design Framework [Cad11b]. So most of the functions of cadence tools can be accessed via SKILL. Hence it is the first choice to extend the functionality of the current design environment and get access to its underneath design database and simulation controls. Additionally, the OCEAN language, which is basically an extension to the SKILL command sets, can be used to access the simulation settings and data [Cad08].

### Model parameter extraction flow

Fig. 4.6 depicts the detailed parameter extraction flow. Two key elements of the flow are testbench execution and result evaluation routine for parameter extraction .

**Testbench** The simulation environment, as shown in Fig. 4.5 consists two equivalent parts: the front-end GUI and the back-end SKILL/OCEAN scripts. Both parts are equivalent. This allows both circuit design team and verification team to access and maintain the testbenches. In this way, they can be executed by both team members, so that simulation runs for model refinement are tightly synced with design progress. However, since analog design tools are mainly GUI based, the whole back-end script has to be done by the verification team. This will increase the work load of the corresponding team at the early phase of the project: they are in charge both in model implementation and coding for the back-end script generation for the testbench.

The back-end script consists of a set of SKILL scripts with additional OCEAN functions for post processing of the simulation data. A top-level script is in charge of initializing the Cadence environment followed by sequential execution of the equivalent testbench scripts. The testbench scripts contain both the setting for different analysis and the post processing routine for the collected date from simulation. For further evaluation tasks, the simulation data will be passed to MATLAB®.

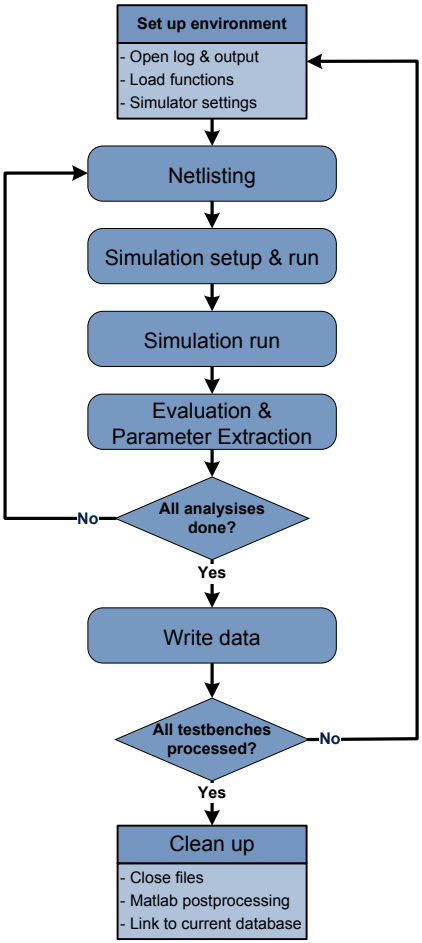



Figure 4.6: Proposed simulation flow for model parameter extraction.

**Extraction** Since simulating the design alone can't lead to the model parameter, further processing steps are required for their acquisition. After the simulation is done, the SKILL back-end starts the post processing routine for the simulation results.

For straightforward parameters, functions provided by SKILL respective OCEAN are sufficient. For sophisticated tasks, e. g. filter property and transition delay, additional routines have been developed. If all analyses have been processed, the extracted parameters of the corresponding simulation are stored in the respective files. In the following, two typical examples for parameter extraction are given.

**LNA IIP3 extraction** Assuming two sinusoidal signals with equal power at the input of the LNA, the nonlinear effect can be described by the third-order Taylor series $V_{out} = \alpha_1 V_{in} + \alpha_3 V_{in}^3$ according to [Raz98]. The output signal of the LNA then can be approximated to

$$V_{out} = \left(\alpha_1 + \frac{9}{4}\alpha_3 A^2\right) A\cos\omega_1 t + \left(\alpha_1 + \frac{9}{4}\alpha_3 A^2\right) A\cos\omega_1 t + \frac{3}{4}\alpha_3 A^3\cos(2\omega_1 - \omega_2) + \frac{3}{4}\alpha_3 A^3\cos(2\omega_2 - \omega_1) + \dots \ . \tag{4.1}$$

The IIP3[3] amplitude $A_{IIP3}$ is considered as the input amplitude, by which both the first order($\omega_1$ or $\omega_2$) and third order output signal components($2\omega_1 - \omega_2$ or $2\omega_2 - \omega_1$) could get the same power. Wenn

$$|\alpha_1|\, A_{IIP3} = \frac{3}{4}\,|\alpha_3|\, A_{IIP3}^3. \tag{4.2}$$

Which leads to the input amplitude:

$$A_{IIP3} = \sqrt{\frac{4}{3}\left|\frac{\alpha_1}{\alpha_3}\right|}\ . \tag{4.3}$$

To get the IIP3 parameter for the model, the geometrical relation between the first order and third order input power are used, as shown in Fig. 4.7. This code performs exactly the same IIP3 extraction as the GUI-based version from the designer, but can be executed automatically. A simulation with two different input powers $P_{in1}$ and $P_{in2}$ can be executed by following OCEAN command, in order to extract the IIP3 parameter.

```
1 ;; Set up simulator
2 simulator( 'spectre )
3
```

[3] Input-referred third-order Intercept Point.

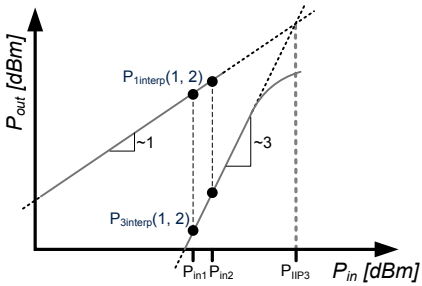



Figure 4.7: Geometrical extrapolation method to estimate IIP3.

```
;; Set up design
current_design = "TB_LNA"
design(  "design_lib" current_design "config" "r" )

;; Set up path to result dir
;; Create the netlist
scs_input = createNetlist(?recreateAll 't)

;; Set simulation options
saveOption( 'save "selected" )
save( 'v "/IN" "/INX" "/OUT" "/OUTX" ) ;voltage of the chosen nets
temp( 27 )  ;temperature

;; Set up design variables
;; Set control bit, input signal power etc.
;; Set up swept multi tone HB and HBAC analysis for IIP3 and run
analysis('hb ?oversample list("3")  ?fundfreqs list("2.4G")  ?maxharms list("3")
        ?errpreset "moderate"  ?tstab "10n"  ?param "prf"  ?start "-50"
        ?stop "0"  ?lin "10"  )
analysis('hbac ?start "2.402G"  ?stop ""  ?maxsideband "5"  )
run('hb 'hbac)
;extract iip3 by extrapolation
r_lna_iip3 = ipnVRI((v("/OUT" ?result "hbac_mt") - v("/OUTX" ?result "hbac_mt")) '(1 ↩
      ↪ -2) '(-1 0) ?rport 1 ?epoint -45)
;;write out parameter to database
iasLogParam("rffe_lna_ip3", r_lna_iip3, "%g")
```

Listing 4.7: Simplified back-end script for LNA IIP3 extraction.

**Filter coefficients extraction** For the extraction of a filter's coefficient, the filter type must be known before any further processing steps. Commonly. the filter

type is described in the design document. Fig. 4.8 shows a polyphase filter(PPF) using a Butterworth filter as prototype.

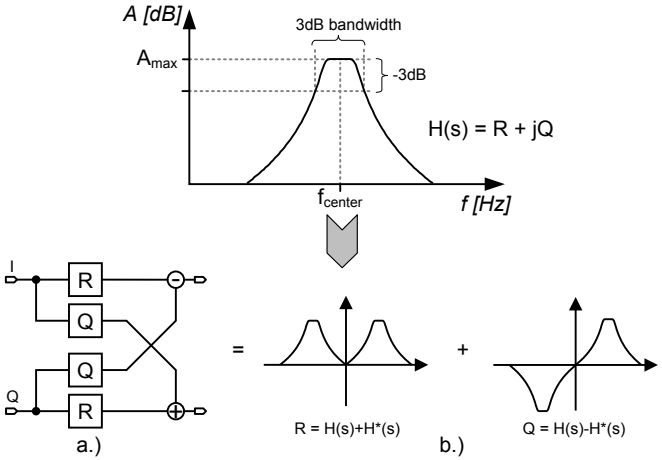



Figure 4.8: Polyphase filter using Butterworth filter prototype. a.) Signal flow of the complex transfer function. b.) Using two real transfer functions $R$ and $Q$ to realize the complex transfer function.

According to [ST+05], the complex transfer function $H_{PPF}(s)$ of a polyphase filter can be realized by two real transfer functions $R$ and $Q$.

$$H_{PPF}(s) = H_{prototype}(s - j\omega_{center}) = R + jQ \Leftrightarrow \tag{4.4}$$

$$R = Re\{H_{PPF}(s)\} = \frac{H(s) + H^*(s)}{2} \tag{4.5}$$

$$Q = Im\{H_{PPF}(s)\} = \frac{H(s) - H^*(s)}{2}. \tag{4.6}$$

SKILL back-end routine can only extract relevant filter characteristic, but not the actual filter coefficients. Therefore, MATLAB® functions are triggered after the simulation in order to get the filter coefficients based on the extracted filter characteristic. For a first order polyphase filter using Butterworth filter prototype, three parameters: gain, $-3\ dB$ bandwidth and center frequency are required.

```
1 ;; save the IO signals
2     save( 'v node_in_i node_in_ix node_out_i node_out_ix node_in_q node_in_qx ↩
          ↪ node_out_q node_out_qx)
```

```
;; Set simulation options and run AC simulation
    analysis('ac ?start "-10M"  ?stop "10M"  ?lin "100"  )
    run('ac)

;; determine the differential output and input amplitudes in dB
    voltage_i_in  = db20(v(node_in_i ?result "ac") - v(node_in_ix ?result "ac"))
    voltage_i_out = db20(v(node_out_i ?result "ac") - v(node_out_ix ?result "ac"))
    ;same for Q path
    gain_ppf_i = voltage_i_out - voltage_i_in
    gain_ppf_q = voltage_q_out - voltage_q_in

;; find out center frequency and the -3db bandwidth
    range_ppf_center = ymax(voltage_i_out - voltage_i_in)
    range_ppf_left   = cross((voltage_i_out - voltage_i_in) (range_ppf_center - 5) 1 '↩
        ↪ rising)
    range_ppf_right  = cross((voltage_i_out - voltage_i_in) (range_ppf_center - 5) 1 '↩
        ↪ falling)
    ;same for Q path

;; determine the bandwidth
    bandwidth_ppf = cross(gain_ppf_i (ymax(gain_ppf_i) - 3) 1 'falling ) - ↩
        ↪ range_ppf_center

;; send following parameters to matlab for coefficient evaluation
    ;gain_ppf_i, gain_ppf_q, bandwidth_ppf, range_ppf_center
```

Listing 4.8: Skill back-end script for PPF parameter extraction.

Listing 4.8 shows part of the back-end script code for filter parameter extraction. The obtained values are exported to MATLAB® to get the filter coefficients, which will be stored in the parameter database afterwards.

### Model parameter update flow

As already mentioned in section 4.2.5, there are two ways to refine the models' parameter: via CDF update in the top-level schematic, e. g. LNA IIP3, or let the model itself access the up-to-date parameter during the simulation, e. g. the filter coefficients. Since the same model in different schematics will also have different CDF parameter database, it is important to automatically update the CDF parameter in every schematic in the design environment. Clearly, this method has the drawback of consuming too much time. Furthermore, not all the schematic are accessible for update, since they can be checked out by other designer for circuit modifications. In order to avoid the error introduced by missing CDF update, the parameter update has to be initiated every time when schematic consisting models is accessed for simulation, in order to avoid inconsistencies in the CDF parameter and hence increase the reliability of the verification. For this purpose, all parameters shown in Fig. 4.5 are stored in an

external database and the model parameter update procedure is integrated into the design framework ad initio.

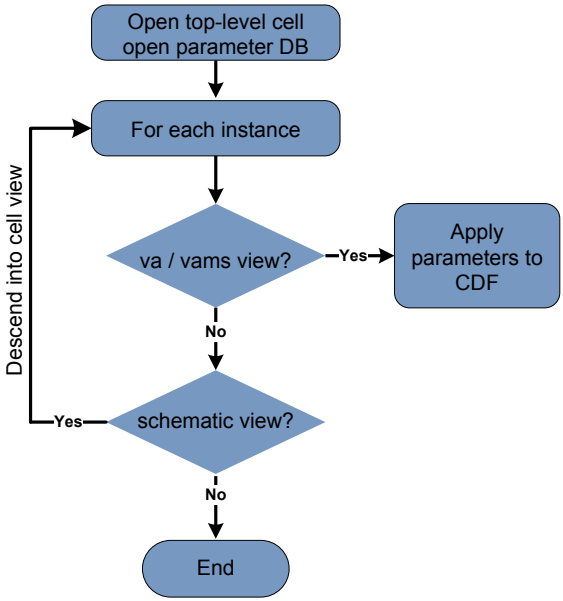



Figure 4.9: Automated hierarchical model update flow.

The main CDF update procedure is implemented in SKILL, it allows a recursive update of the CDF parameter through the whole schematic hierarchy. Fig. 4.9 depicts the model update flow. The function is embedded into the schematic editing GUI, so it can be accessed easily. In addition to the SKILL procedure, the naming convention of testbench schematics has been established from the beginning, every testbench schematic will have the prefix TB_ and the parameters with update feature get the postfix _apx. An alternative procedure will go through the whole testbench schematics, if accessible, and update the corresponding CDF parameters on a daily basis.

### 4.2.6 Top-level simulation and sign off

Once the design moves to top-level assembly, besides refining models at higher level of abstraction through the hierarchy, a comprehensive top-level verification

strategy is required to achieve a reasonable level of confidence. Following aspect have to be considered:

**Full chip integration** For the full chip integration, large amount connections for analog as well as the interface to digital part have to be done. In contrary to block-level, where every model can be validated with their transistor-level representation side-by-side, models at top-level require an accurate refinement based on the lower level models and exact description of their interface. Therefore, when refining the event-driven models from the top-down process, the primary concern should be preserving the correct functionality and accurate interface description of the block. This is also the reason, why the verification team is not allowed to change the schematic from the beginning as mentioned in section 4.2.3. Second or third order effects in the top-level model should be avoided as they will generally slow down the model development and the simulation speed. Additionally, each subsystem has to be simulated and verified standalone, before gradually moving them into the macro-level, and later on top-level schematic. Furthermore, assuming the digital part has been already verified standalone, it has to be ensured that the current working copy of the digital part, regardless RTL code or synthesized VERILOG netlist is checked out from the repository.

**End-to-end signal flow simulation** Once the subsystems have been merged into top-level, simulation with simple stimuli won't be enough to verify the complete signal path in detail. Typically, for the top-level simulation, modulated signals for the testbench are required. In this way, signals from the input to the output of the chip, crossing all blocks and paths can be checked. In addition to that, the digital configuration and algorithm can also be verified in the presence of their analog counter part. This is also the best method in practice to disclose further implementation errors in the digital part and to ensure the signal integrity of the whole chip.

## 4.3 Summary of the proposed methodology

This chapter presented a modeling methodology for the verification of a complex RF SoC. Several techniques, such as design partitioning, hierarchical model implementation, assertion coding, as well as automated model parameter extraction and update are discussed. The proposed modeling flow shifts part of the model refinement work to the early project phase, as the benefit of automated model refinement later on prevails the higher initial workload. The overall modeling flow can be summarized in the following 4 steps.

1. Starting with block-level design, build up the analog models and validate them side-by-side in the same simulator with the circuit. At the same

time, the key model parameters are identified, testbenches used by the designer will be translated into equivalent SKILL/OCEAN script for their automated execution. Needless to mention, most details of the block-level circuits, especially the start-up time, condition and sequence have to be documented in the design document for better debugging process later on.

2. Implement the event-driven models for each block, use the analog models as reference. Additional event-driven testbenches have to be built up by the verification team according to the analog ones in the first step. At this step, key parameters identified before can be updated automatically with the update approach proposed in section 4.2.5 and 4.2.5. For analog/RF blocks, the wreal data type is used to move the analog potential/flow into digital domain with higher abstraction. Frequency behavior in RF band should be mapped into equivalent delays. Otherwise, it will consume too much simulation time. Assertions have to be implemented into the models. At this step, the automated parameter extraction and update approach starts to be beneficial, since lots of model parameter refinement can be done without manual intervention.

3. Build up top-level testbench, implement input stimuli in addition to the ones available in th top-down model sets. Especially for the signal path, besides some signal tone sources for some initial check, real modulated signals are inevitable. The single tone signals are way better traceable through the hierarchy compare to the real modulated ones, but the real modulated signal is the only source to determine the functional correctness of the system. Refine the top-down model with detailed pin/port description. Validate the top-level testbench, define the simulation set for the top-level regression run and determine the required levels of abstraction for each test case. Furthermore, an early simulation performance estimation can be done by enabling the profiling options in the mixed-signal simulator, in order to find out if there is any unnecessary cross domain connections, which slows down the simulation.

4. Verify the system according to the specification. Apply the real modulated signal to the input, if any modem[4] activities is required for the functional verification. Run regression tests, find out available functional errors, communicate with design team for possible bugs.

The event-driven model for the top-level functional verification in the mentioned flow still uses passband modeling technique instead of the baseband one, due to the mentioned language limitation of Verilog-AMS. Although the simulation performance thereby is significantly increased compare to the analog simulation

[4] modulation and demodulation

at model or transistor-level. The reasonable system time for the simulation is therefore limited in the millisecond range. For simulating systems with increasing complexity or digital algorithm requiring even longer system time to converge, higher simulation performance is desirable. Hence, implementation of pin-accurate event-driven baseband models is an inevitable further step to facilitate functional verification at higher system design hierarchy.

Although part of the models can be updated automatically, radical design changes, such as add/remove pins, change digital control polarity from active low to active high still require manual intervention. That's also the reason for using a strict coding style, so that human errors can be reduced as far as possible. Furthermore, model parameters stored in an external data base can also be accessed by external tools and programs, this is especially beneficial for the SystemC virtual prototype, as will be discussed in the next chapter.

# 5

# A SystemC-Based RF Virtual Prototyping Methodology

This chapter presents a method to generate pin-accurate SystemC models based on the RF circuit schematics, using a SKILL based routine. This method addresses the verification and modeling of the analog/RF circuits in the event-driven digital domain using SystemC, based on the same database created by the circuit designer.

Recalling Fig. 3.1, the reachable design hierarchy using Verilog-AMS is confined from the transistor implementation to functional behavior level. This is also the mainly targeted design hierarchies in the current mixed-signal design environment e. g. Cadence® DFII. However, in addition to prove the functionality, since the hardware itself only serves as a part of a system, the Virtual Prototype (VP) of the whole chip, which captures the latest snapshot of the hardware part (think of IP reuse), is also necessary to ensure the correct hardware/firmware interaction. Hence, two factors have to be considered if moving to higher design hierarchy. First, the consistency of the design. As already mentioned, each tool has only limited coverage of its targeted design hierarchy. In our case, if the whole design hierarchies from HW/SW to circuit level have to be covered, different modeling languages is here inevitable[1]. Considering the system level design, most models have been created by using Matlab® or C-based HDLs(s. section 3.1.2)

[1] Although Fig. 3.1 indicates, that SystemVerilog as an extension to Verilog-AMS can cover higher design hierarchy, it is still a maturing language, which doesn't offer the high degree of openness as those from SystemC.

like SystemC, whereas Verilog-AMS[2] is mainly used in the circuit design hierarchy. Therefore, the consistency between the database from system level model to the circuit level is mandatory. Second, the simulation performance has to be increased, especially for the RF part of the SoC. Passband modeling methods used in Verilog-AMS cannot provide high enough simulation speed to come up with the increasing complexity in the whole system. Design and verification at higher hierarchy demand the introduction of models with higher levels of abstraction. However, most of the models of the analog/RF subsystems are handcrafted, therefore, adding a further level of abstraction will also increase the work of the modeling and verification team for updating and validating the models to ensure the design changes have also been covered by the models through the whole hierarchy.

The proposed method intends to refine system level models based on the schematic information, this leads to the possibility to keep the consistency between the system level models and those from schematic level. On the one hand, the connectivity information of the analog/RF subsystem can be extended into the system level and verified, therefore the consistent virtual RF prototype can be provided. On the other hand, the behavioral models in SystemC can be executed more efficiently, thus helps the development of further system components as well as SoC verification. The remainder of this chapter will describe the requirements and ideas of generating pin-accurate SystemC models based on the schematic database. Furthermore, pin-accurate baseband model implementation technique in SystemC to increase the simulation performance will be presented.

## 5.1 Requirements for RF virtual prototype implementation

For a consistent system level RF virtual prototype based on circuit design database, the model generation and refinement process has to be done automatically, capturing every possible design changes and reducing the human errors introduced by manual model implementation to minimum.

One might argue with extending the classical HDLs via their available C-interface, such as VPI. While the authors of [Che09]; [Che10] combines the real number and baseband modeling methods via VPI in order to get the most simulation performance, were the main target of the authors in [MZ+08] to add vector and matrix operation semantics in Verilog-AMS. Both of them share the same idea: bring the different levels of design abstraction into an unified simulation environment. However, such extensions of classical HDLs cannot

[2] Generally, classical HDLs such as VHDL or Verilog family can be used here, but in this work, as mentioned in early chapter, Verilog-AMS is the main HDL used in circuit design level.

offer the same degree of openness and flexibility of a C-based HDL such as SystemC. If more HW/SW interaction is required, the VPI approach will reach its limit with respect to flexibility. Additionally, extending a HDL will cause additional debug overhead. Hence, SystemC has been chosen for the proposed RF prototype.

Generally speaking, the proposed RF prototyping approach is a bottom-up model refinement process by creating pin-accurate system models in order to keep the consistency of models at every levels of abstraction from circuit to system design. In this case, the proposed approach has to fulfill following requirements:

- Circuit and system level models should share the same database.
- The schematic database serves as the reference for the model refinement process, it cannot be modified.
- The virtual prototype must preserve the pin-accuracy and reflect the hierarchical structure of the circuit level design.
- The virtual prototype has to be portable and can be simulated in an unified simulation environment with other components of system design.

## 5.2 Hierarchical SystemC model frame generation flow

As already mentioned in section 5.1, the virtual prototyping and functional verification of the RF systems have to be done in a pure digital simulator and require HDL which can handle higher levels of abstraction, in order to achieve the most simulation efficiency in the whole SoC. SystemC fulfill this requirement, hence the automatic generation of pin-accurate SystemC RF models is required. By generate the mostly part of the VP automatically, the error source introduced by handcrafting an additional model set can be excluded. Fig. 5.1 illustrates the proposed system level model refinement and virtual prototyping approach, which comprises of 3 components:

(1) Automatic generation of pin-accurate SystemC model frame based on the schematic information.

(2) Automatic extraction of model parameters based on transistor-level simulations.

(3) A comprehensive set of standard SystemC core behavioral models for analog/RF building blocks.

The groundwork of the proposed methodology is a formalized AMS design and verification flow, consisting of both top down model generation, as well as bottom up model refinement, which has been discussed in chapter 4.

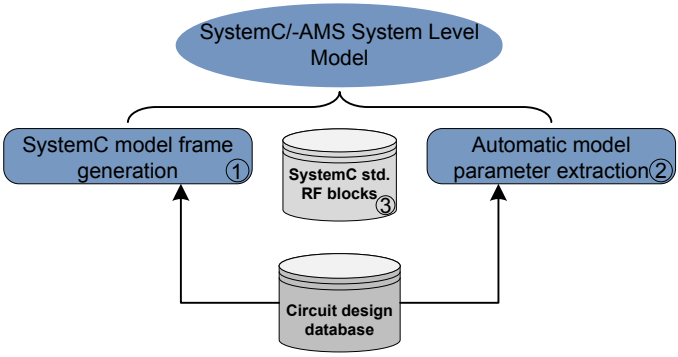



Figure 5.1: Proposed SystemC model refinement and RF virtual prototyping approach.

### 5.2.1 Comparison of different approaches

Two possibilities have been considered in order to get the hierarchy and structure information of the existing circuits/subsystems and export them accordingly:

1.) A pure SPICE netlist translator, which converts the whole netlist into SystemC model frames. This approach needs two steps to generate pin-accurate models: first, it has to build up the hierarchy from the already flattened netlist, which is based on the schematic information. Second, it generates the SystemC model frame and compares the SystemC instance with existing instances in the schematic database. While this approach is generic and almost universal for every SPICE-type netlist, it needs intensive keyword training and extension to understand the mixed-signal netlist[3], which contains analog/digital circuit information, simulation controls and HDL based behavioral models. Additionally, the analog part of a mixed-signal netlist typically doesn't contain any I/O port direction information, which is essential for data flow simulation of the SystemC models. This will cause a subsequent manual modification of the SystemC models, which is error prone since the models in SystemC don't have any graphical representations.

2.) A direct translator of the given schematic using the SKILL language. Since the schematic already comprises the hierarchy structure of the given

[3]Although the mixed-signal netlist is SPICE-like, some of the netlists uses Verilog-AMS to declare the instances(s. section 2.2.3), which generally will increase the keyword training for hierarchically parsing the netlist instances into equivalent SystemC models.

System, the SKILL language features the possibility to access and translate this information directly into SystemC. Additionally, the property of the ports can be invoked from SKILL to determine the I/O direction in the equivalent, pin-accurate SystemC models. Fig. 5.2 shows the major accessible schematic information via SKILL.

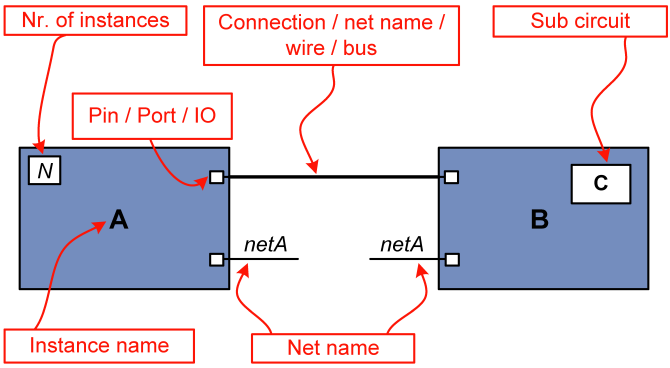



Figure 5.2: Accessible schematic information via SKILL

The SKILL based approach for the automatic SystemC model frame generation has been chosen, since the required information regarding schematic hierarchy, instance I/O and connectivities are already available in the design database, which is directly accessible using the SKILL based routine. Additionally, the SKILL code offers better integration possibilities into the available design flow. Therefore, possible errors through netlist re-mapping, keyword training and manual interaction can be reduced to the minimum.

### 5.2.2 Hierarchical generation of pin-accurate SystemC models

The overall model frame generation procedure is illustrated in figure 5.3. First, the hierarchical structure of the whole circuit inclusive the I/O direction of each instances and their connectivities will be analyzed. Second, connectivity validation and resolution e.g. naming conventions of schematic data will be considered to ensure correctness for the model translation. The corresponding names for instances, pins, and nets will be converted or expanded to fit the naming rules in SystemC syntax.

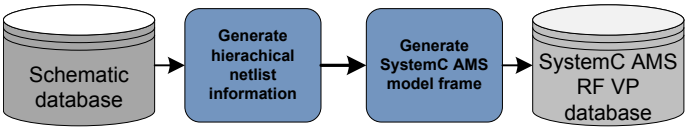



Figure 5.3: General flow of the automatic SystemC model frame generation

### Map the hierarchical structure

The entry point of the proposed flow is the top-level schematic, all instances in the schematic will be listed and sorted to prepare for further iteration. For each instance[4], two processes will be executed. First, the structure of the instance, including pins, IO directions and the subsequent interconnection to other available instances will be stored in a temporary database for further processing. Second, the available views[5] of each instance will be analyzed. If a "schematic"-view is available for the instance, the procedure will be executed recursively, until either the defined hierarchy level for the VP is reached or all instances don't have the "schematic"-view anymore. At the end of this process, the hierarchical structure of the targeted circuit will be mapped to an intermediate file, which will be passed on for the SystemC model frame generation. This intemediate file will be referred as hierarchy database in the following The flow chart of the mentioned process is shown in Fig. 5.4. The output hierarchy database consists of information about the schematic name, subsequent instance names, the corresponding library names, view names and the number of instances. Listing 5.1 shows part of its SKILL implementation.

```
1  defun( HierSchTree (@key    (cv (geGetEditCellView))
2                              (exList nil)
3                              (port poport)
4                              (layer 0))
5
6      let((sortedHeaders count s1 sv)
7      ;sortedHeaders:         Sorted header of the list file
8      ;count:                 Number of instances
9      ;s1:                    Tmp obj for cache instances database
10     ;sv:                    New cellview object
11
12     sortedHeaders = sort(cv~>instHeaders 'abCompareCellName);
13     foreach(header sortedHeaders
14         when(!member(header~>libName exList)
```

[4] Basically, the instance in the schematic can either be the circuit element, stimuli or sub-circuits.

[5] Views refer to the actual used abstraction of the instance for the top-level, e. g. extracted, layout, schematic, Verilog-A, Verilog-AMS, etc.

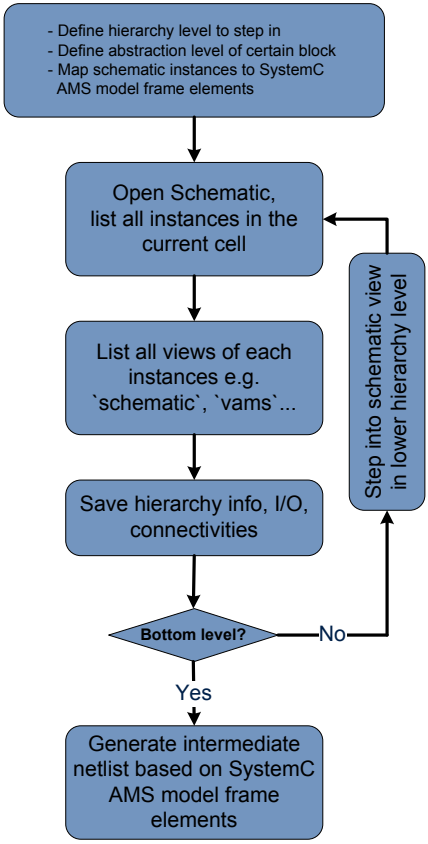



Figure 5.4: Flow chart for generating the intermediate file

```
15                  s1 = header~>instances;
16                  count = length(setof(inst s1 nequal(inst~>purpose "pin")));
17                  when(greaterp(count 0)
18                      sv = dbGetAnyInstSwitchMaster(car(s1) "schematic");
19                      TreeOutputElement(port layer header count)
20                      when(sv
21                          HierSchTree(?cv sv ?exList exList ?port port ?layer layer+1);
22                          );when(sv
23                      );when(greaterp(count 0)
24              );when(!member(header~>libName exList)
25          );foreach(header sortedHeaders
26      t);let..
27  )
```

Listing 5.1: Source code for generating the hierarchy database.

### Generate SystemC model frames

The model frame for the VP will be generated based on the information provided by hierarchy database. Since the schematic and the VP will share the same hierarchy structure, some similarities between the schematic and its corresponding SystemC VP is also given, as shown in Fig. 5.5. The detailed SystemC model

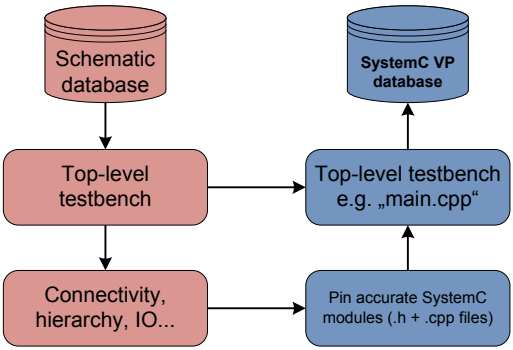



Figure 5.5: Overview of the schematic hierarchy and its corresponding SystemC model frames

frame including the model construction part, the hierarchical connection to its sub modules, its IO ports and signals are now generated. Since not all instances in the schematic contain sub-circuits, the model frame generator has to treat the cases differently. Three functions: `genSC_MAIN()`, `genSC_Module()` and `genSC_Cell()` are available to handle different mentioned cases. For instances without any

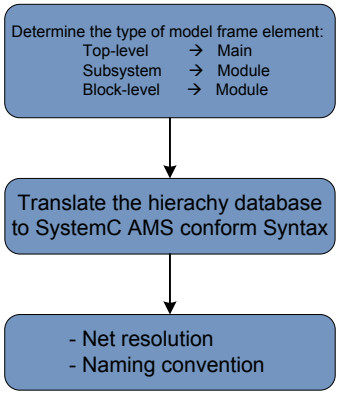



Figure 5.6: General flow description of the SystemC model frame generation

sub-circuit, the `genSC_Cell()` is called to generate the corresponding SystemC model frame, the corresponding SKILL code is shown in listing 5.2.

```
1  procedure(genSC_Cell(cv_, default_path_)
2      let((s1 s2 port pin_sc)
3          ;s1:                    tmp string to save cellName of cellview
4          ;s2:                    tmp string to save SystemC module and pin name
5          ;port:                  port to generate SC_Cell
6          ;default_path_:         the default path to access VP database
7
8                  s1 = cv_~>cellName;
9                  port = outfile(sprintf(nil "%s/%s.h" default_path_ s1));
10
11                 /*header part of SC_Cell*/
12                 ...
13                 fprintf(port "SC_MODULE(%s){\n\n" cv_~>cellName);
14
15                 /*generate pins declaration and port direction*/
16                 fprintf(port "//\tport_declaration\n");
17                 foreach(pin cv_~>terminals
18                         pin_sc = pinSch2SC(pin~>name "" "string");
19                         case(pin~>direction
20                             ;generate the port + direction
21                         );case
22                 );foreach
23
24                 /*inner signals defination*/
25                 fprintf(port "//␣signal␣declaration\n");
26                 foreach(pin cv_~>terminals
27                         s2 = pinSch2SC(pin~>name "_sig" "string");
```

```
28                          fprintf(port "\tsc_signal<T>␣%s;\n" s2);
29                  )
30                  ...
31      );let..
32  );procedure
```

Listing 5.2: Source code for generating a single SystemC cell.

Otherwise, the `genSC_Module()` will be executed and go through the specific path of the instance's own hierarchy. During traversing of the instance's hierarchy, `genSC_Module()` will also generate the hierarchical interconnections based on the information from the hierarchy database, including the basic information in the model frame mentioned in the former case. At the end, depending on the user's choice, the `genSC_MAIN()` will be executed to finish the top-level of the SystemC VP. Fig. 5.6 shows the general steps of the mentioned process. The last step will be described in the next section.

#### Naming conventions and resolution functions

Besides generating pin-accurate SystemC model frames, the naming conventions for the schematic design has also to be considered to ensure correctness for the model translation. The corresponding names for instances, pins, and nets will be converted or expanded to fit the naming rules in SystemC syntax. Some of the naming conventions used for net and bus in the design environment(s. Fig. 5.2) [Cad11a] e.g. $<*2>net02$(two times $net02$) or $net<0{:}12>$(bus with the width of 13) are not directly supported in SystemC. Hence, a proper net name remapping functional has to accompany the model frame generation process.

Fig. 5.7 shows part of the special cases of naming conventions in schematic, which should be resolved and remapped during the model frame generation process. Understanding the meaning of the naming conventions in schematic, the remapping keyword can be trained. The resulting names in SystemC corresponding to cases in table 5.1 are listed in table 5.2.

Three functions are implemented `PortSch2SC()`, `NetSch2SC()`, and `InstSch2SC()` in order to cover different naming convention cases in the nets, ports and instances.

While different naming conventions can be covered using different mapping functions, further processing steps are required to resolve some special net connection cases. In the analog design, it is common to connect some nets to a single node. The potential(voltage) keeps the same at one node, whereas the current sums up(s. section2.2.2). Models at higher level of abstraction in SystemC cannot resolve such relationship as mentioned in section 3.3.3. Therefore, a proper resolution function has to be implemented for such cases to avoid the multi driver on single node issue in digital simulation. e. g. case 1 shown in Fig. 5.7 the three parallel nets with the name of $net1 < 2 : 0 >$ are

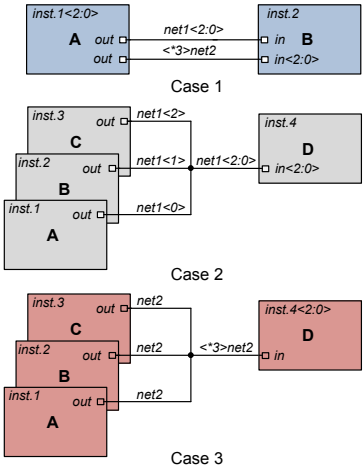



Figure 5.7: Special cases of schematic naming conventions in Cadence® DFII, which is not natively supported in SystemC.

| Instance name | Terminal name | Net name |
|---|---|---|
| Ix | out | net01 |
| Ix | out | net02<0> |
| Ix | out<14:0> | net03<14:0> |
| Ix<1:0> | out | <∗2>net01 |
| Ix<1:0> | out | net02<0:1> |
| Ix<1:0> | out | <∗2>net03<0> |
| Ix<1:0> | out<14:0> | net01<29:0> |
| Ix<1:0> | out<14:0> | <∗2>net02<14:0> |

Table 5.1: Schematic naming conventions in Cadence® DFII.

| Instance name | Terminal name | Net name |
|---|---|---|
| Ix | out | net01 |
| Ix | out | net02[0] |
| Ix | out[14]∼[0] | net03[14]∼[0] |
| SYSC_par_0_Ix | out | SYSC_par_0_net01 |
| SYSC_par_1_Ix | out | SYSC_par_1_net01 |
| SYSC_par_0_Ix | out | net02[0] |
| SYSC_par_1_Ix | out | net02[1] |
| SYSC_par_0_Ix | out | SYSC_par_0_net03[0] |
| SYSC_par_1_Ix | out | SYSC_par_1_net03[0] |
| SYSC_par_0_Ix | out[14]∼[0] | net01[29]∼[15] |
| SYSC_par_1_Ix | out[14]∼[0] | net01[14]∼[0] |
| SYSC_par_0_Ix | out[14]∼[0] | SYSC_par_0_net01[14]∼[0] |
| SYSC_par_1_Ix | out[14]∼[0] | SYSC_par_1_net01[14]∼[0] |

Table 5.2: Remapped naming conventions in SystemC.

connected to a single node at the input of the instance B. For SystemC VP, the name of the net has to be firstly remapped. Afterwards, the resolution function will generate a piece of code, which generally let the user to choose the behavior at the input of instance B. E. g. if multiple current sources are driving a single node:

$$\begin{aligned} \text{net1} = &\text{SYSC\_par\_0\_net1} \\ &+ \text{SYSC\_par\_1\_net1} \\ &+ \text{SYSC\_par\_2\_net1} \end{aligned}$$

Which generally creates a virtual summation node for the currents at the input of instance B, as shown in Fig. 5.8. After the net resolution process is done, the overall model frames for the SystemC VP are complete.

## 5.3 SystemC RF building blocks library

A library that contains a model set of RF building blocks and AMS components has also been implemented. The library is mainly targeting to describe the core functionalities and behavior of the RF blocks and subsystem for system level. All models are implemented in a way, so that they are highly flexible and customizable. Furthermore, the models have to be optimized in terms of simulation performance compare to the event-driven passband models in Verilog-AMS. Fig. 5.9 depicts the structure of the generated SystemC model

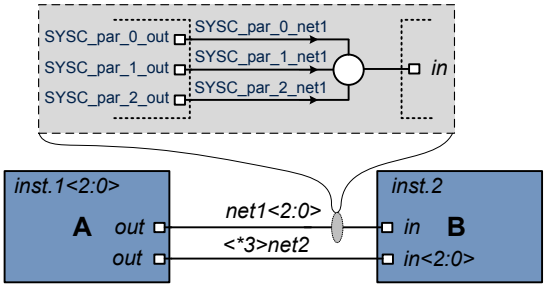



Figure 5.8: Net resolution function for SystemC VP. Here, both the output and input ports of instance 1 and instance 2 are considered as current ports. Hence, the resolution function will sum up the input signals.

based on the proposed methodology. It consists of the automatically generated model frame and the hierarchical connection information, as well as the basic characteristics taken from the SystemC RF building block library, such as e. g. gain, noise and linearity. Parameters for each of the mentioned behavior will be updated from the model parameter database mentioned in section 4.2.5, which has been extracted from the design environment. The only manual intervention for completing the model is to map the functional response of the block if digital controls apply, e. g. control sequence for gain and bandwidth switching, power on/off states.

### 5.3.1 Simulation performance aspects

The usage of wreal signal in Verilog-AMS can significantly increase simulation efficiency. However, the wreal data type allows only to transfer one real number, in one direction all the time [Cad11c], which limits the possibility to use equivalent baseband model to get higher abstraction: Either the wires and pins in signal path of the models has to be modified or the signal has to be changed externally via VPI to fulfill the requirement. Both will increase the functional verification effort in different ways: Changing pins means to change the design, which is not allowed for verification engineers. Using VPI means embedding additional C code into Verilog-AMS models, which will reduce the transparency of the models, hence reducing the reliability of the models for verification.

Since the SystemC models are not limited regarding their signal types, not only real numbers can be passed through between the models like the wreal approach mentioned in [WVM+09], but also user-defined signal types, such as those mentioned in section 3.2.3. The simulation of the SystemC models can

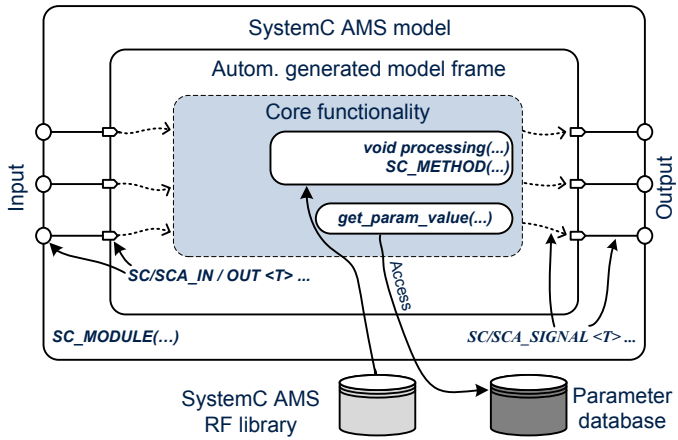



Figure 5.9: Structure representation of the automatically generated SystemC model.

only be done in the time domain, thus some of the analog specifications in the frequency domain like phase noise or filter characteristics have to be mapped into the event-driven time domain, which has been discussed in [WVM+09]; [WJ+09]. The SystemC model library consisting of the RF building blocks, must be able to pass complex numbers or even arrays through the ports for realizing the pin-accurate multi tone baseband modeling approach. In this way, the aforementioned limitation of Verilog-AMS can be vanquished while preserving the pin-accuracy, so that the simulation efficiency can be increased further. The whole models and signal types are implemented in a pure C++ environment with high code transparency, which makes the verification more reliable compare to embedding C code via VPI interface.

### 5.3.2 Switchable baseband models of RF blocks

In addition to the mentioned performance aspect, the RF core models exhibits the property to automatically switch between the passband and baseband signals, depending on the type of the input signal. Baseband models commonly require more signal components compare to the passband ones [Che05]. One baseband signal should contain at least three components: in-phase ($I$), quadrature-phase ($Q$) and carrier frequency ($\omega_0$). If more concern about the nonlinear effects or higher order harmonics has to be made, additional components have to

be considered and append to the existing baseband signal. The key feature of the proposed methodology is the consistent pin-accuracy in all levels of abstraction, from the circuit design to the SystemC VP environment. As previously anticipated in earlier section, the advantages of the SystemC modeling approach lies in the fact that the SystemC modules are based on the C++ classes and offers its object oriented features, new signal type can be manually defined as a C++ class [DG03]. By overloading the operators in SystemC, the operators' functions can be defined according to the type of its operands. Hence, the representation of a signal is not limited in real number, moreover it can also be represented in vectors and matrices. A class called *BB_double* containing 8 parameters $DC$, $I_1$, $Q_1$, $I_2$, $Q_2$, $I_3$, $Q_3$ and $w_0$ are defined based on Taylor series approximation:

$$\begin{aligned} s(t) = DC(t) &+ I_1(t)cos(w_0t) + Q_1(t)sin(w_0t) \\ &+ I_2(t)cos(2w_0t) + Q_2(t)sin(2w_0t) \\ &+ I_3(t)cos(3w_0t) + Q_3(t)sin(3w_0t) \end{aligned} \tag{5.1}$$

Basically, the `+`, `-`, `/`, `*`, `=` and `==` operators have been overloaded, both operand `BB_double` and **`double`** have been considered. Listing 5.3 shows the current implementation of the mentioned `BB_double` class, where only the overload of `*` operator is shown in detail.

```
1  class BB_double{                                    //baseband double;
2  public:
3      double f0;                                      //passband frequency;
4      double DC;                                      //s(t) = DC + I1*cos(w0t) + Q1*sin(w0t)
5      double I1, Q1, I2, Q2, I3, Q3;;                 //+ I2*cos(2w0t) + Q2*sin(2w0t)+ ↩
           ↪ I3*cos(3w0t) + Q3*sin(3w0t);
6
7  public:
8      ...
9      BB_double operator * (double A) const{          //overload * if operand is double
10         BB_double result;
11         result.f0 = f0;
12         result.DC = DC * A;
13         result.I1 = I1 * A;
14         result.Q1 = Q1 * A;
15         ...
16         return result;
17     }
18     ... //overload / if operand is double
19     ... //overload /int_number for quantization effeects
20     BB_double operator * (BB_double & rhs) const{   //operator * overloading;
21         BB_double result;                           //In case BB_double * BB_double;
22         if(rhs.f0 == f0){
23             result.f0 = f0;
```

```
24              result.DC = DC * rhs.DC + I1*rhs.I1/2 + I2*rhs.I2/2 + I3/2*rhs.I3/2
25                          + Q1*rhs.Q1/2 + Q2*rhs.Q2/2 + Q3*rhs.Q3/2;
26              result.I1 = DC*rhs.I1 + rhs.DC*I1
27                          + (Q1*rhs.Q2)/2 + (rhs.Q1*Q2)/2 + (Q2*rhs.Q3)/2 + (rhs.Q2*Q3)/2
28                          + (I1*rhs.I2)/2 + (rhs.I1*I2)/2 + (I2*rhs.I3)/2 + (rhs.I2*I3)/2;
29              result.Q1 = DC*rhs.Q1 + rhs.DC*Q1
30                          + (rhs.I1*Q2)/2 + (rhs.I1*Q2)/2 - (rhs.I2*Q1)/2 - (rhs.I2*Q1)/2
31                          + (I2*rhs.Q3)/2 + (rhs.I2*Q3)/2 - (I3*rhs.Q2)/2 - (rhs.I3*Q2)/2;
32              result.I2 = DC*rhs.I2 + rhs.DC*I2
33                          + (I1*rhs.I1)/2 + (I1*rhs.I3)/2 + (rhs.I1*I3)/2
34                          + (Q1*rhs.Q3)/2 - (Q1*rhs.Q1)/2 + (rhs.Q1*Q3)/2;
35              result.Q2 = DC*rhs.Q2 + rhs.DC*Q2
36                          + (I1*rhs.Q1)/2 + (rhs.I1*Q1)/2 + (I1*rhs.Q3)/2
37                          + (rhs.I1*Q3)/2 - (I3*rhs.Q1)/2 - (rhs.I3*Q1)/2;
38              result.I3 = DC*rhs.I3 - (rhs.Q1*Q2)/2 - (Q1*rhs.Q2)/2
39                          + rhs.DC*I3 + (I1*rhs.I2)/2 + (rhs.I1*I2)/2;
40              result.Q3 = DC*rhs.Q3  + (I1*rhs.Q2)/2 + (rhs.I1*Q2)/2
41                          + rhs.DC*Q3 + (I2*rhs.Q1)/2 + (rhs.I2*Q1)/2;
42          }
43          else {
44              if(rhs.f0 == 0){
45                  result.f0 = f0;
46                  result.DC = DC * (rhs.DC + rhs.I1 + rhs.I2 + rhs.I3);
47                  result.I1 = I1 * (rhs.DC + rhs.I1 + rhs.I2 + rhs.I3);
48                  result.Q1 = Q1 * (rhs.DC + rhs.I1 + rhs.I2 + rhs.I3);
49                  result.I2 = I2 * (rhs.DC + rhs.I1 + rhs.I2 + rhs.I3);
50                  result.Q2 = Q2 * (rhs.DC + rhs.I1 + rhs.I2 + rhs.I3);
51                  result.I3 = I3 * (rhs.DC + rhs.I1 + rhs.I2 + rhs.I3);
52                  result.Q3 = Q3 * (rhs.DC + rhs.I1 + rhs.I2 + rhs.I3);
53              }else if(f0 == 0){
54                  result.f0 = rhs.f0;
55                  result.DC = rhs.DC * (DC + I1 + I2 + I3);
56                  result.I1 = rhs.I1 * (DC + I1 + I2 + I3);
57                  result.Q1 = rhs.Q1 * (DC + I1 + I2 + I3);
58                  result.I2 = rhs.I2 * (DC + I1 + I2 + I3);
59                  result.Q2 = rhs.Q2 * (DC + I1 + I2 + I3);
60                  result.I3 = rhs.I3 * (DC + I1 + I2 + I3);
61                  result.Q3 = rhs.Q3 * (DC + I1 + I2 + I3);
62              }else{
63                  cout<<"Multiplication_with_unbalanced_frequency!!!!!"<<endl;
64              }
65          }
66          return result;
67      }
68      ...// overload + operator, considering different cases regarding f0s
69      ...// overload - operator
70  };
```

Listing 5.3: Source code of the mentioned `BB_double` class.

Additionally, the baseband model has to share the same code with its passband counterpart. Otherwise, seamless switch between passband and baseband signal type would take too many code and resource changes, which is not feasible. Hence, the only change, which has to be made for switching between those two signal abstractions has to be limited to the change of the signal type. Listing 5.4 shows such implementation.

```
template<class T>
SC_MODULE(SC_LNA){
public:
    sc_in<T> SC_LNA_INPUT;
    sc_out<T> SC_LNA_OUTPUT;

public:
    SC_HAS_PROCESS(SC_LNA);
    SC_LNA(sc_module_name, ...);

private:
    void proc(void);
    //parameters for LNA model such as ICP IIP3...
};

template<class T>
SC_LNA<T>::SC_LNA(sc_module_name n, double Gain_, double NF_, double BW_, double IIP3_,
        double RISO_, double IRL_, double ORL_, double Rin_, double Rout_, double Load_){

    c1 = sqrt(pow(10, Gain/10)) * Rout / Rin;
    c3 = 2 * c1 / pow(10, (IIP3-30)/10) / Rin / 3;
    rms = sqrt(kB * LNA_T * (pow(10, NF/10)-1) * BW);
    Amax = sqrt(4/3 * c1 / c3);

    SC_METHOD(proc);
    sensitive<<SC_LNA_INPUT;
};

template<class T>
void SC_LNA<T>::proc(){
    T noise_signal_in = AWGN(SC_LNA_INPUT.read(), 0, rms);
    noise_signal_in = limit(noise_signal_in*ref_Loss, Amax);
    T third_harm = noise_signal_in * noise_signal_in * noise_signal_in;
    T tmp_out_sig = (noise_signal_in*c1) + (third_harm * c3);
    SC_LNA_OUTPUT.write(tmp_out_sig);
};
```

Listing 5.4: Source code of a switchable SystemC LNA model.

## 5.4 Summary of the proposed RF VP methodology

Based on the methodology proposed in this chapter, pin-accurate SystemC RF VP according to the circuit-level hierarchy and interconnections can be generated automatically. The multi frequency simulation capability of SystemC can be demonstrated by realizing the pin-accurate RF baseband models without any further modifications of the connectivity information. The flexibility and portability of an open source C++ based HDL allows us to verify the complete signal path based on the same hierarchical information from schematic level. Thus increases the speed and reduces the error proneness of the VP generation process. In order to demonstrate the effectiveness of the methodology, some application examples and their results will be shown in the next chapter.

# 6

# Application Examples

This chapter consists of three application examples that are results of industry cooperated projects, in order to demonstrate the effectiveness of the proposed verification and virtual prototyping methodology.

Based on the rules of design partitioning and the criteria for model implementation given in the early chapters, the examples shown here will also point out some experience and notes gained through the projects in addition, especially in terms of determining modeling domains and approaches, defining the partition of the design and how to trade simulation performance against accuracy for a fast and reliable functional verification.

## 6.1 A fractional-N PLL based transmitter

Fig. 6.1 shows the structure of the PLL frequency synthesizer, which is the core part of the investigated transmitter. It consists of the VCO, multi-modulus divider with a digital $\Delta\Sigma$-modulator, phase frequency detector(PFD), charge pump(CP) and a loop filter(LPF).

### 6.1.1 Verilog-AMS block-level model implementation

The mentioned PLL can be considered as a typical mixed-signal system, where the analog VCO generates the high frequency oscillation, while the digital part is in charge of controlling the center frequency and modulation. Although the structure shown in Fig. 6.1 is clear, the long simulation time, especially when modulation comes into consideration, will greatly slow down the project's

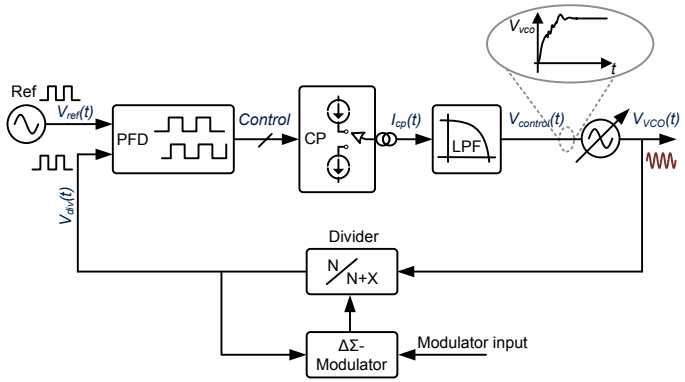



Figure 6.1: Block-level of the investigated fractional-N PLL based transmitter

progress. Therefore, implementing models of corresponding blocks for the functional verification is an inevitable task. The main target of the verification in this case, besides to check the functionality of the whole system, is to provide the overall phase noise parameter based on the circuit level simulation of each block.

Considering the different modeling contributions for PLL frequency synthesizers, different approaches has been presented in the past:

1.) The phase domain approach, which has been presented in [HMW07] and [YCK07], uses linear phase-domain models to predict the influence of phase noise. In this case, the VCO model is the most time consuming block in the PLL, since it was implemented in the analog domain. In [HMW07] and [Kun06], the model of divider has been merged into the VCO model in order to increase the simulation efficiency. While this modeling approach is well suited for the top-down design to estimate the overall noise of the PLL system, it has some drawbacks when it comes to the verification for given design. On the one hand the realization of merged VCO plus divider doesn't correspond to the block-level circuit structure, which makes the verification of connectivities and debugging of single blocks on transistor-level almost impossible. On the other hand, using analog operator like `@cross()` or analog system function like `$transition` in the VCO model is not efficient enough as we will see later in section 6.1.1.

2.) The event-driven approach [SFB05b] is originally developed for pure digital circuits. It formulates the phase noise into its equivalent jitter in the time domain. Compare to the phase-domain models, the digital models

have much higher simulation efficiency, but the target of the mentioned implementation was to estimate the overall phase noise performance of the PLL. Hence, it didn't consider aspects in the model for the functional verification.

The proposed approach combines the advantages of the aforementioned methods. On the one hand it is possible to model each of the blocks in the event-driven domain in order to increase the simulation efficiency. On the other hand the models can be embedded into testbenches on transistor-level for a mixed-level simulation, since the wreal data type enables analog precision in the event-driven domain. However, special calculations have to be done in order to accurately map the block-specification from frequency- into time-domain while keeping the simulation efficiency as high as possible, as we will see in the following sections.

Commonly, the PLL is partitioned into the high and low frequency part during the design period. The designer tries to use analog models to substitute the VCO and the divider in order to reduce the simulation time and hence to make a proper overall phase noise estimation by mixed-level transient simulation. Since the analog models of the PLL system can only provide limited simulation performance enhancement and no special RF simulation techniques can be applied to the whole PLL. It has been come to the conclusion that the analog model set of the given PLL will only serve as an auxiliary aid for the designer in order to estimate some performance parameters, e. g. phase noise. For the functional verification at the PLL top-level, mainly the event-driven model set will be used.

### VCO

The event-driven analog VCO model structure is shown in figure 6.2. The modeling target is to provide a VCO model with high simulation efficiency, which also maps the given specification from circuit level accurately enough in order to fulfill the aforementioned verification requirements. First the input voltage signal is converted to the output frequency or rather the desired oscillation period. Then the timing jitter, which is computed according to equations 3.10 and 3.15, is added to the period, which leads to the noisy VCO output signal.

Listings 6.2 and 6.1 show the analog and event-driven VCO model implementations in each case. The phase noise specification given from circuit level simulation result is $-91.5\ dBc/Hz$ at an offset frequency of $1\ MHz$ and $f_0 = 433\ MHz$. The VCO model uses *dist_normal*() system function to map the white noise equivalent timing jitter.

```
1 module wrealVCO(out, Vcontrol)
2 output out;
```

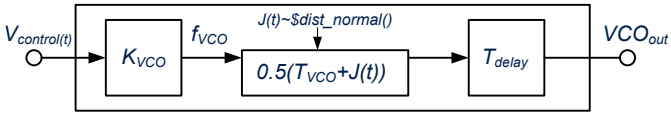



Figure 6.2: Event-driven VCO model structure.

```
input Vcontrol;
wreal out, in;
//...parameters
//...check configuration
always@(Vcontrol) begin
//calculate the freqeuncy
freq=Vcontrol*Kvco
end
always begin
#next
...
c=pow(10,phase_noise/10.0)*pow(Deltaf,2)/(pow(freq,2));
accJitter=sqrt(c/(freq*2.0));
dT0=accJitter*$dist_normal(dSeed,0,1);
dT1= dT*fc*next+dT1_last;
next= 0.5/freq+dT0+dT1;
dT1_last = dT1;
...
end
assign out= vco_out;
...
endmodule
```

Listing 6.1: Event-driven VCO model.

```
freq=Vcontrol*Kvco
accJitter = sqrt(pow(10,phase_noise/10)*pow(Deltaf,2)/pow(freq,3));
n = pow(accJitter/1.414,2)*pow(freq,3)*2;
freq_n= freq/(1+noise*freq);
phi = 2*`M_PI*idt(freq_n,0);
V(phase) <+ 2*`M_PI*(idtmod(freq_n,0.0,1.0,-0.5));
@(cross(V(phase)-0.5*`M_PI,1,ttol) or cross(V(phase)+0.5*`M_PI,1,ttol))
begin
    noise = accJitter*sqrt(2)*$rdist_normal(seed,0,1);
end
V(out) <+ cos(phi) ;
```

Listing 6.2: Simplified analog VCO model.

As one can observe, the output of the analog VCO model uses `cross()` to detect the change of phases, in order to apply the phase noise to the output. The corresponding output signal is generated using `cos(phi)`. Here, the analog simulation time step has to be reduced by applying the `$bound_step()` system function, in order to ensure the accurate `cross()` detection. Consequently, the simulation of $3 \cdot 10^5$ oscillation cycles with wreal VCO model needs about 1 seconds to complete, while the simulation with the phase model, which also has been used in [YCK07] with the same specification and condition needs about 6.1 minutes. Fig. 6.3 shows the comparison of the results of phase noise simulation between analog and wreal model implementations according to the aforementioned specifications.

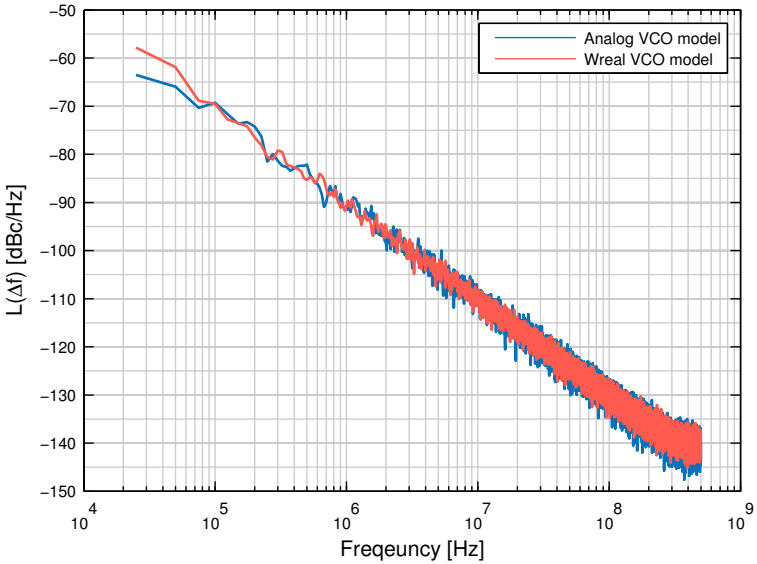



Figure 6.3: Phase noise comparison between analog and wreal VCO model.

Fig. 6.3 clearly demonstrates that the wreal model can achieve higher simulation efficiency while keeping sufficient accuracy for verification purpose. However, the random delay of the timing jitter is quantized to the timing accuracy settings of the digital simulator and the implementation, as mentioned in section 3.3.1. Hence, proper timescale setting has to be considered during the model implementation process.

### Divider

The divider shown in Fig. 6.1 reacts to falling and rising edges of the VCO output signal. Since the wreal signal doesn't offer edge detection, the input signal has to be converted into logic value. One possibility to do this is to use an internal wire to convert the wreal signal, as shown in listing 6.3. The counter implemented in the model increments by one when the VCO signal creates a rising or falling edge. The divider output will remain unchanged until the counter has reached the divider ratio. Additionally, the synchronous jitter is added to the output in order to map the circuit level specification.

```
1  module divider_wreal( out, in, dsm_in ) ;
2  input in ; wreal in ;
3  input dsm_in; wreal dsm_in;
4  output out;  wire out;
5  parameter real vlo, vhi;
6  ...parameters of jitter and divider ratio...
7  integer counter, fp, dSeed, dratio;
8  wire vin_intern;
9  reg vintern ;
10 real diff, prev, now, dT;
11 assign out = vintern;
12 assign vin_intern=(in==Vhi ? 1'b1 : 1'b0);
13 initial begin
14 ...
15 end
16 always@(posedge vintern or negedge vintern) begin
17 if (counter == dratio-1) begin
18 dT=syncjitter*$dist_normal(dSeed,0,1) ;
19 counter = 0 ;
20 #((td+dT)/1f) vintern=~vintern;
21 end
22 else
23 counter = counter + 1 ;
24 end
25
26 always@(dsm_in) begin
27 //change the divider ratios
28 end
29 endmodule
```

Listing 6.3: Part of the divider source code.

The output signal of the divider serves also as clock signal for the $\Delta\Sigma$ modulator, which changes the divider ratio pseudorandomly in order to produce its fractional part.

### ΔΣ modulator

**Modulator** The ΔΣ modulator used in the mentioned PLL minimizes the phase noise close-by the center frequency by pushing the phase error towards higher frequency bands. Due to its pure digital implementation, no additional modeling work for the ΔΣ modulator is required. Fig. 6.4 shows its transfer function. The output $Y$ of the modulator is used to set the current divider ratio.

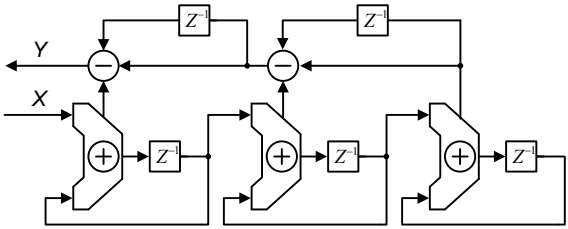



Figure 6.4: Third order MASH 1-1-1 ΔΣ modulator used in the PLL.

The noise contribution of the ΔΣ modulator to the overall PLL phase noise is shown in Fig. 6.5. The spurious are in multiples of the reference frequency, which is 13 $MHz$.

**Dithering** Dithering is a mechanism used to intentionally provide irregularity in the quantization errors of the ΔΣ modulator. If the quantization error is subject to a certain periodicity and is correlated with the input signal of the modulator, it is called a cyclically repetitive and deterministic errors, this error will result in unwanted large spurs in the output signal spectrum. Hence, dithering has to be applied to the modulator, in order to reduce the spurs in the output signal. Linear-Feedback-Shift-Register(LFSR) is commonly used to generate pseudo random numbers for the dithering mechanism. For the mentioned PLL, a 15-Bit LFSR is used for the random number generation. It will be added to the modulator input signal at the first accumulator.

### PFD and charge pump

Based on circuit level simulation and the theory from [LP+03], the PFD cannot detect small phase errors between reference and divider output signals. Thus it loses its linearity in small phase errors. The ΔΣ modulator is very sensitive to such nonlinearity and responds to increased phase noise and spurious emissions. Therefore, additional delay elements shown in Fig. 6.7(a) have been inserted

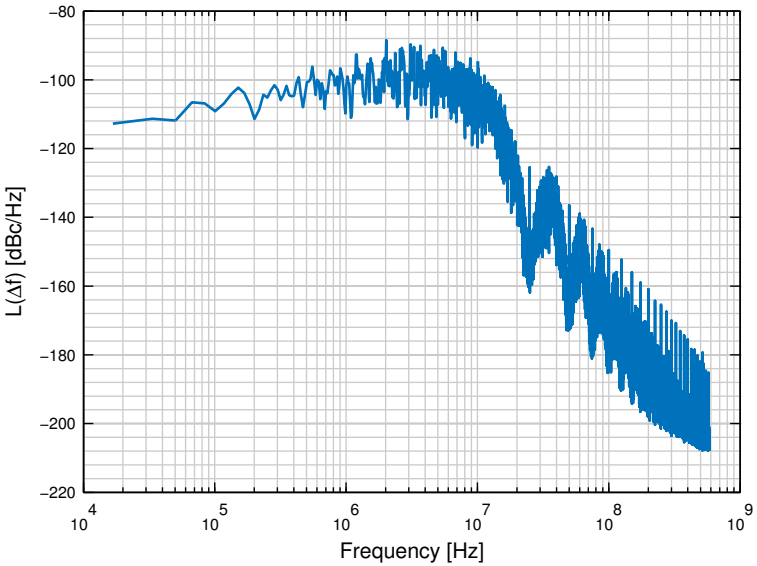



Figure 6.5: Phase noise contribution of the 3rd order ΔΣ modulator used in the PLL.

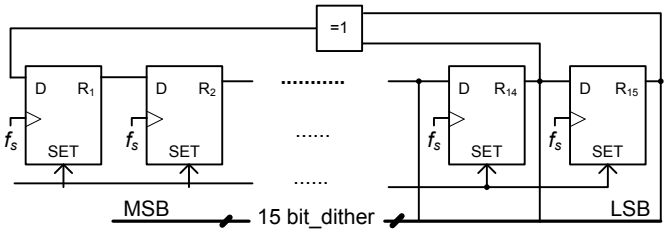
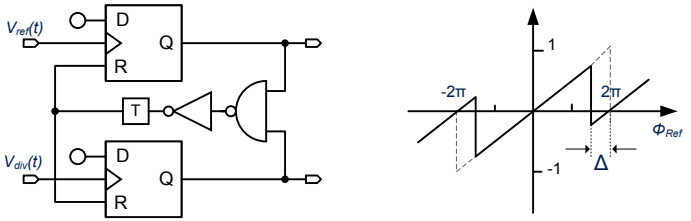



Figure 6.6: The 15-Bit LFSR used for dithering.

between the "AND" gate and the reset signal input of the flip flops in order to reduce the influence of small phase errors.




(a) The PFD structure with additional delay element.

(b) The PFD behavior in the phase domain, dahsed line shows the reduced linear range Δ.

Figure 6.7: Structure of the PFD and it's limited linear range.

This also leads to a reduced linear range of the PFD [LP+03]: only phase errors between 0 and $2\pi - \Delta$ , where $\Delta = \frac{2\pi \cdot \tau_{delay}}{T_{ref}}$, can be corrected. However, the delay time $\tau_{delay}$ has to be kept as small as possible, otherwise, it will significantly increase the PLL lock-in time and cause additional unwanted metastable oscillation on the output signal.

### Loop filter

For the implementation of the loop filter model in VERILOG-AMS, three different approaches are available: analog, equivalent digital IIR filter with constant sampling rate or non-uniform sampled. For analog implementation, the Laplace operators in VERILOG-A `laplace_nd()` or `laplace_zp()` can be used. For the event-driven implementations, the filter specification has to be firstly converted to the discrete time domain, e. g. using MATLAB®. listings 6.4 and 6.5 depicts the different event-driven implementations for a second order loop filter, shown in Fig. 6.8, in each case.

```
1 assign out=yn ;
2 always@(in) begin
3 xn=in; act_time=$realtime;
4 hn=1.0f*(act_time-prev_time);
5 dxn=(xn-xn1)/hn;
6 vn=vn1+hn*xn1; un=un1+hn*yn1;
7 yn=(( z0*vn+z1*xn+z2*dxn-p0*un)/p1+p2/p1*yn1/hn)/(1.0+p2 /( p1*hn ));
8 end
```

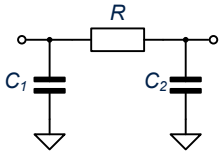



Figure 6.8: A second order loop filter.

Listing 6.4: Non-uniform sampled LPF model.

```
assign out=y0;
always #ts begin
x2 = x1; x1 = x0; x0 = in;
y2 = y1; y1 = y0;
y0 = ((n0*x0)+(n1*x1)+(n2*x2)-(d1*y1+d2*y2))/d0;
end
```

Listing 6.5: Equivalent IIR filter with equidistant sampling points.

Consider the analog implementation as reference. Fig. 6.9 shows the AC simulation results of the analog and non-uniform sampled LPF implementations. As one can observe, both implementations show good match in the lower frequency range between 10 $KHz$ and 10 $MHz$. However, in the range between 10 $MHz$ and 100 $MHz$, the non-uniform sampled version clearly deviates from the specification, due to the stability issues mentioned in section 3.3.1. This will considerably affect the final phase noise result. Therefore, an alternative event-driven loop filter implementation, which is triggered by both its input and an internal system clock has been done, in order to accurately map the phase noise specification. The timing error $\Delta T$, shown in Fig. 6.10, will be corrected by an additional calculation of the corresponding amplitude error $\Delta A$, in order to improve the accuracy of the non-uniform sampled model.

### 6.1.2 Top-level simulation of the PLL

Each block of the PLL has been modeled and refined according to the method presented in the chapter 4. Subsequently, the top-level has been built up by combining the blocks. In addition to model the required second order effects such as overall phase noise and limited linear region of PFD, the correct start-up time of each block is also considered in the model. Since the mentioned PLL is designed for low power applications. The timing of the chip's start-up also plays an essential role for the verification. For a low power operation, each block of the chip will only be activated if needed. E. g. the most part of the digital circuits

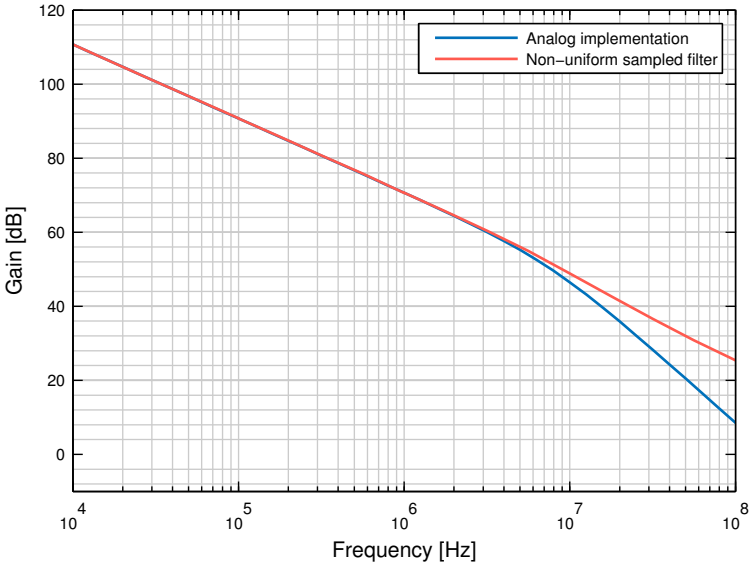



Figure 6.9: Comparison of different LPF implementations.

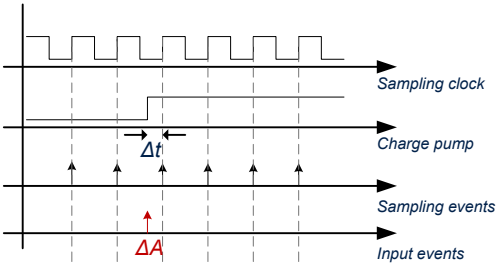



Figure 6.10: The timing diagram of the proposed LPF implementation.

will be turned off during the standby. The activation of the required blocks will also obey a fixed schedule. Further more, since powering off the digital part will also cause the loss of its current configuration, the correct sequence for the settings has to be in the focus of the assertions. E. g. the digital configuration can only be applied, if the main clock(which typically depends on the reference clock) is up and ready.

Starting from the supply and biasing network, the PLL will be activated after 200 $\mu s$, when the reference clock is settled. Wrong start-up sequence due to twisted pin or mis interpreted specification will result in the failure of the chip. Table 6.1 shows the basic PLL specification. While the top level simulation on

| **Reference** | **VCO tuning range** | **Overall phase noise** |
|---|---|---|
| 25 $MHz$ | 368 $MHz$ 450 $MHz$ | $-100\ dBc/Hz@\ \Delta f = 1\ MHz$ |

Table 6.1: Part of the PLL specification.

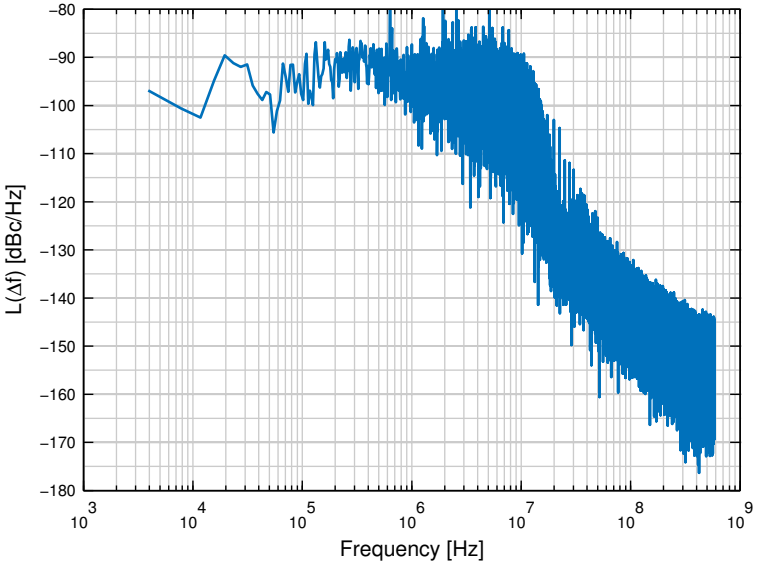



Figure 6.11: The overall phase noise of the PLL system based on model simulation.

circuit level could take several days to complete, apart from the consideration of convergence and memory problems, the verification based on the wreal models took 16 minute to complete the start-up check, the configuration check and the phase noise simulation in 10 $ms$ system time. So that every small design changes caught by the automatic parameter extraction routine can be verified at the top-level "over night". Fig. 6.11 shows the overall phase noise of the PLL based on the event-driven simulation. The phase noise at $\Delta f = 1\ MHz$ is 101.5 $dBc/Hz$, which is about 1.5 $dB$ off from the silicon measurement.

While the above example clearly shows the feasibility of the top-level functional verification based on the event-driven models. Same approach has been applied to a more complex Bluetooth PLL transmitter system shown in Fig. 6.12. The

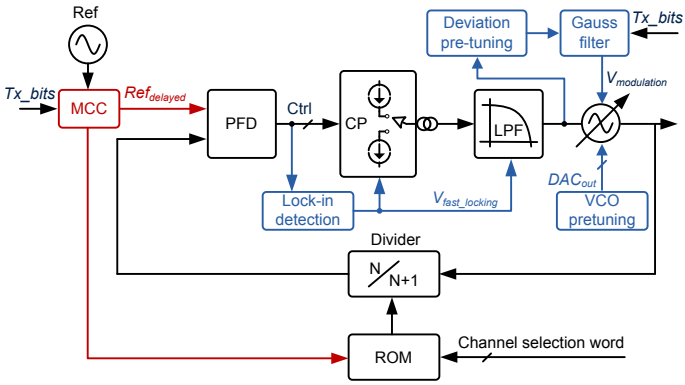



Figure 6.12: Block-level overview of the integer-N PLL based Bluetooth transmitter.

Bluetooth transmitter system has to fulfill the basic requirements shown in table 6.2.

Compare to the PLL architecture shown in Fig. 6.1, the Bluetooth transmitter design shown in Fig. 6.12 is based on an integer-N PLL. The reference clock is at 2 MHz, so each Bluetooth signal channel frequency can be covered by multiplication between the divider ratio and the reference. However, the slow reference clock will also increase the lock-in time of the PLL, which has to be compensated for the sake of Bluetooth standard. According to the standard, the Bluetooth transmitter operates with frequency hopping spread spectrum(FHSS), which generally spread the signal over the 79 channel at the rate of 1600 times per second in transmission mode and 3200 in paging mode respectively. For such kind of operation mode, a precise timing diagram is inevitable for the functional

| Specification | Parameter |
|---|---|
| Frequency range | $2.400 - 2.4835$ GHz |
| RF channels | $f = 2402 + k$ MHz, $k = 0...79$ |
| Lower & Upper guard | 2 MHz& 3.5 MHz |
| Modulation scheme | GFSK[a] |
| Basic data rate | 1 Mbit/s |
| BT | 0.5 |
| Modulation index | $[0.8 : 0.35]$ |
| Frequency deviation | $140 - 175$ KHz |
| -20dB bandwidth | $< 1$ MHz |
| Transmitter settling time | $< 220\mu s$ |
| Hopping interval paging/transmission | $312.5\mu s$ / $625\mu s$ |

[a]Gaussian Frequency Shift Keying

Table 6.2: Basic Bluetooth transmitter specification according to [The02]

verification. Therefore, the modeling task has to focus not only on the timing and functional behavior at RF side according to the specification, but also on the resulting non-ideal effects which causes additional delays, as well as the required baseband functions for the calibration and tuning process. Table 6.3 depicts the timing requirement for the transmission mode. In the following, only the background and the model implementations of the transmitter blocks highlighted in blue and red in Fig. 6.12 will be discussed in detail.

| Processes | Timing requirement |
|---|---|
| VCO pre-tuning | $0 - 11\mu s$ |
| Lock-in at channel N-1 | $12 - 62\mu s$ |
| Lock-in at channel N+1 | $62 - 112\mu s$ |
| Lock-in at channel N & modulation deviation calibration | $112 - 162\mu s$ |
| Signal transmission | $220 - 625\mu s$ |

Table 6.3: Timing diagram for tuning and transmission processes during one hopping interval.

**Lock-in detection** In addition to the limited linear region in the PFD(s. section 6.1.1), the charge pump possesses similar non-ideal behavior. Assume the PLL is nearly locked and the $\tau_{delay}$ is small compare to the period of the reference

clock, so that the phase error $\Phi_{err}$ between reference and feedback clock becomes nearly zero, In this case, the up and down control pulse at the output of the PFD(see Fig. 6.1) become so short, that neither the PMOS nor the NMOS current source in the charge pump can be properly turned on due to the finite rising time and the internal logic signal propagation delay. The time interval, in which the CP cannot generate current pulse for the phase correction, is defined as backlash [SYR02]. Common approach to combat the mentioned issue, besides to extend the delay time $\tau_{delay}$ in the PFD, is to lower the CP's current as soon as the PLL is considered as locked. The lock-in detector evaluates the delay time between the up and down pulses, if the delay is small, it indicates a lock-in condition of the PLL. In this case, the lock-in detector will send the control signal to lower the CP current and reduce the LPF bandwidth at the same time, in order to reduce the overall phase noise. The lock-in control signal has to be synchronized with the reference clock. Otherwise, switching the CP's current will cause additional unwanted phase distortion.

**Deviation and VCO pre-tuning** In order to keep modulation index of GFSK between 0.28 and 0.35, the peak modulation voltage has to be adjusted every time before the transmission for each Bluetooth channel, due to variations come from the chip's fabrication. Hence, the actual VCO gain $K_{VCO}$ has to be estimated, before the transmission starts at certain channel $N$. For this purpose, the PLL will be locked at the $N-1$ $N+1$ and $N$ three times before the transmission. Based on the output voltage of the LPF at the first two lock-in points, the deviation pre-tuning block will extrapolate the actual tuning characteristic of the VCO for the channel $N$ and set the proper modulation voltage $V_{modulation}$ accordingly. The VCO pre-tuning pursuits the similar goal. If the transmitter is going to hop into a new channel, the VCO will be disconnect from the loop and coarsely set to the wanted frequency with a tolerance of 5 $MHz$ within 11.5 $\mu s$, in order to speed up the PLL lock-in process.

**Modulation compensation circuit(MCC)** During the transmission of modulated GFSK signal, each transmitted bit will lead to the change of the VCO output frequency. Due to the feedback nature of the PLL, a long sequence of "1"s or "0"s will force the PLL to regulate VCO frequency, which will negatively affect the transmission. The MCC is in charge of temporarily adjusting the divider ratio and the reference clock delay, so that the PLL loop won't affect the transmission.

The blocks can be modeled and simulated stand alone for model validation. However, correctness of each single block is not enough. The tight interaction between the mentioned blocks, each required timing activities and the resulting

functional correctness can only be verified by executing a top-level simulation. Fig. 6.13 shows the top-level simulation result of the functional verification, As

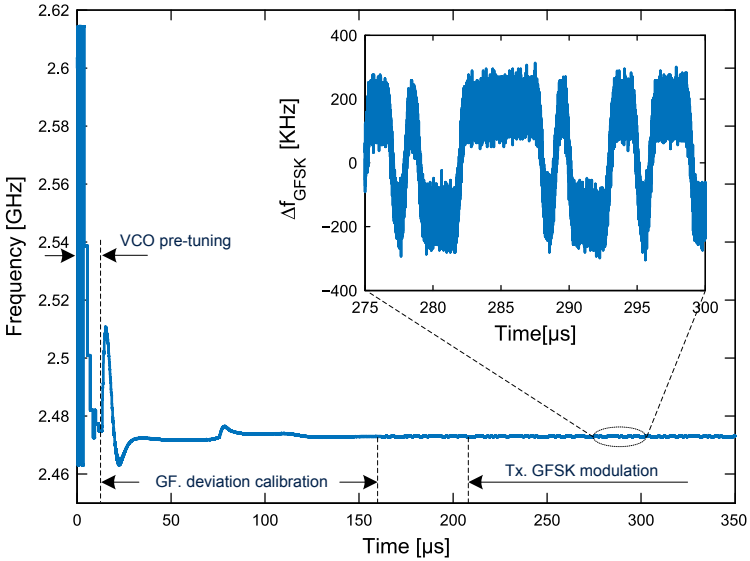



Figure 6.13: Transient simulation result of the Bluetooth transmitter.

one can observe, the VCO is firstly pre-tuned to about 2.47 $GHz$. Afterwards, the PLL loop is locked at $2.471GHz$ after $50\mu s$ and at $2.473GHz$ after $100\mu s$ respectively. Finally, PLL is locked at $2.472GHz$ after $150\mu s$ and the transmission starts after $200\mu s$. As shown in the zoomed-in plot in Fig. 6.13, the modulated signals fulfills the mentioned requirements for Bluetooth shown in table 6.2. Fig. 6.14 shows the spectrum of the transmitted signal at 2.45 $GHz$. Table 6.4 shows the simulation times at different levels of abstraction for the PLL based Bluetooth transmitter.

### 6.1.3 Pin-accurate SystemC VP of the PLL

**Top-level PLL VP simulation results** The SystemC virtual prototyping process has been initiated right after the top-level testbench has been built up. Since the SystemC PLL models have also been implemented based on the event-driven approach, same modeling fundamentals discussed in the early section apply here. Part of the SystemC VCO model has been shown in listing 6.6. Since the

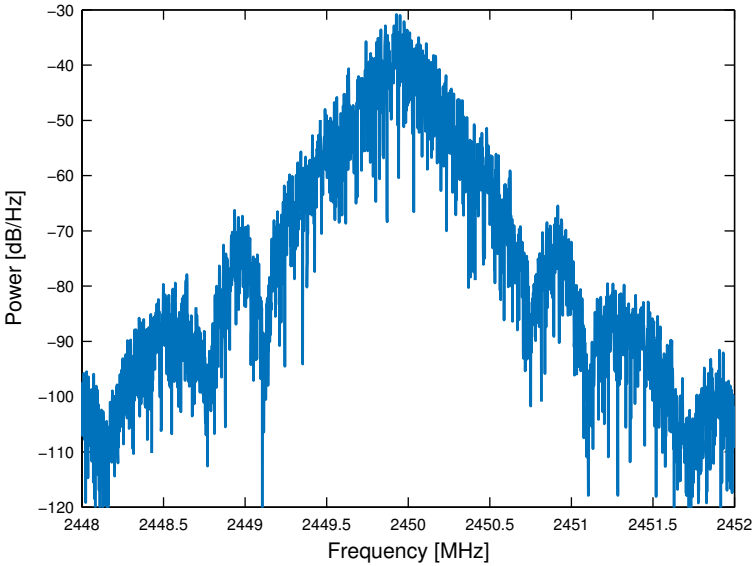



Figure 6.14: Spectrum plot of the Bluetooth PLL output signal at 2.45 $GHz$.

| **Level of abstraction for top-level** | **CPU time** |
|---|---|
| VERILOG-AMS event-driven model | 0.213 $H$ |
| Transistor-level | est. 100 $H^a$ |

[a] Transistor-level simulation for 1 $ms$ was impossible due to some convergence issues in the mixed-signal simulator. Thus, the simulation time shown here is only estimated based on a 20$\mu s$ simulation run.

Table 6.4: Comparison of the top-level simulation performance of the Bluetooth transmitter for 1 $ms$ system time.

equivalent jitter implementation (line 23-25) requires the generation of normal distributed numbers, the box-muller[1] [BM58] algorithm has been used to fulfill this task (line 15-21). The PLL VP therefore cannot only verify the functionality of the chip, but also deliver the overall phase noise performance estimation. Till now, the PLL model sets are available both in Verilog-AMS and SystemC, the transient simulation time for 500 $\mu s$ for different HDLs have been compared in the table 6.5.

```
1  //*** begin automatic generated part:
2  SC_MODULE(block_3stage_vco_calibration_dnw){
3    sc_in<double>VCON
4    sc_in<bool>HIGHER_FREQ
5    sc_in<double>VSS
6    sc_in<double>VDD
7    sc_out<PLL_BB>VCO_OUT
8  //other hierarchical connection information
9  ...
10 //*** begin hand crafted part
11 - Define model parameters
12 - Calculate the VCO frequency Freq=f(Vcontrol, Kvco)
13 - Generate normal distribution based on
14   uniform distributed random numbers:
15 double x1, x2, y1, y2, w;
16 do{ x1=2.0*(double)rand()/(double)RAND_MAX -1.0;
17      x2=2.0*(double)rand()/(double)RAND_MAX -1.0;
18      w=x1*x1+x2*x2;
19   } while(w>=1.0);
20   w=sqrt((-2.0*log(w))/w);
21   y1=x1*w; y2=x2*w;
22 //Calculate the phase noise equivalent timing jitter
23 accSD = sqrt(pow(10, phasenoise/10)*pow(deltaf, 2)/pow(freq_org, 3))/sqrt(2.0);
24 Jitter = accSD/std * (mean + std * y1);
25 - Update the VCO frequency
26 //write out the VCO signal
27 VCO_OUT.write(...); };
```

Listing 6.6: Part of the SystemC VCO model.

The lock-in processes during the transient simulation of both model sets are shown in Fig. 6.15.

As expected, no significant difference, either in the simulation time or in the lock-in process, can be observed, given the fact that both simulations are done in the event-driven simulator. Listing 6.7 shows the automatically generated PLL top-level testbench in SystemC.

[1] Basically, there are other implementation variants in C++ for generating normal distributed random numbers. The box-muller algorithm has been chosen, due to its implementation simplicity(it requires only two uniform distributed random numbers).

| HDLs | Simulation time |
|---|---|
| SystemC 2.2.0 | 31$s$ |
| Verilog-AMS(wreal) | 40$s$ |

Table 6.5: Simulation times of the PLL models

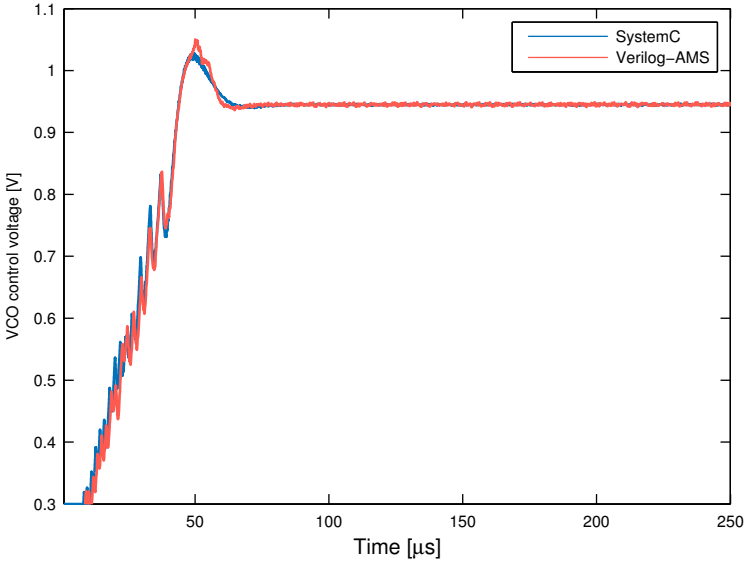



Figure 6.15: VCO control voltage during the PLL lock-in process.

```
1  int sc_main(int argc, char* argv[]){
2  sc_signal<bool> net028, net027;
3  sc_signal<bool> vco_tune[2];
4  sc_signal<double> vdd;
5  sc_signal<BB_SIG> VCO_outn;
6  sc_signal<bool> net48;
7  sc_signal<bool> net048
8  sc_signal<bool> freq_out
9  sc_signal<double> gnd
10 sc_signal<bool> net61, net043, net9;
11 sc_signal<bool> net023, net037, net050;
12 sc_signal<BB_SIG> VCO_outp;
13 sc_signal<double> net047;
```

```
14 sc_signal<bool> net026;
15
16 PLL_toplevel i0_PLL_toplevel("PLL_toplevel");
17 i0_PLL_toplevel.VCO_Out_N(VCOoutn);
18 i0_PLL_toplevel.VCO_Out_P(VCOoutp);
19 i0_PLL_toplevel.F_Reference(net61);
20 i0_PLL_toplevel.Fast_calibration(vco_tune[1]);
21 i0_PLL_toplevel.Ibias_100u_source(net047);
22 i0_PLL_toplevel.Slow_calibration(vco_tune[0]);
23 i0_PLL_toplevel.Switch_Reference(net028);
24 i0_PLL_toplevel.VDD(vdd);
25 i0_PLL_toplevel.VSS(gnd);
26 i0_PLL_toplevel.frac_mode(net027);
27 i0_PLL_toplevel.reset(net026);
28 i0_PLL_toplevel.sdin(net9);
29
30 digital_config_interface
31 i0_("digital_config_interface");
32 i0_.DIV_RATIO(net9);
33 i0_.DS_RST(net026);
34 i0_.FRAC_MODE(net027);
35 i0_.REF_SEL(net028);
36 i0_.VCO_TUNE(vco_tune);
37 i0_.CFG_DATA(net037);
38 i0_.CFG_PCLK(net48);
39 i0_.CFG_SCLK(net023);
40
41 //models for testbench configuration
42 //and monitoring
43 freq_meter i0_freq_meter("freq_meter");
44 i0_freq_meter.average_freq(net048);
45 i0_freq_meter.out(freq_out);
46 i0_freq_meter.in(VCO_outp);
47
48 cfg_data i0_cfg_data("cfg_data");
49 i0_cfg_data.data(net037);
50 i0_cfg_data.pclk(net023);
51 i0_cfg_data.sclk(net48);
52
53 clk_gen i0_clk_gen("clk_gen_verilog");
54 i0_clk_gen.clk_out(net61);
55 ...
56 sc_start(...);
57 \end{lstlisting}
```

Listing 6.7: Automatically generated top-level testbench in SystemC.

**Involving AMS extension into the VP concept** While SystemC was initially developed to model digital systems, the TDF MoC offered by its AMS extension

allows the direct implementation of continuous time Laplace transfer functions, which makes the model implementation of the LPF much easier. Hence, a the LPF in the mentioned PLL VP has been replaced with an TDF MoC. The TDF time step is set as same as the sampling freqeuncy of the event-driven LPF implementation. The synchronization therefore between TDF and SystemC kernel slows down the simulation around 100 seconds. As a result here, it can be stated that the TDF MoC cannot show its true potential in a pure event-driven implementation environment.

**Increase the VP simulation performance** In addition to the simulation results of the PLL lock-in process, the overall phase noise performance of the SystemC VP is shown in Fig. 6.16. Compare to the phase noise evaluation result shown in Fig. 6.11, it can be observed that the SystemC models exhibits the best simulation efficiency without losing the accuracy of the models by considering some major performance characteristics of the PLL.

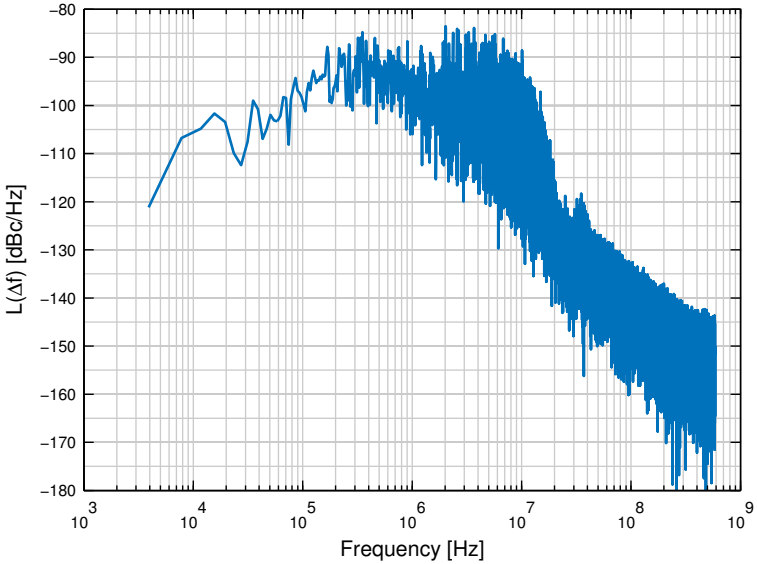



Figure 6.16: PLL phase noise evaluation based on SystemC VP.

However, the presented simulation performance of the PLL cannot satisfy the speed requirement for the VP. Especially for a multi-mode, multi-standard RF

SoC, in which the PLL is mainly used for generating the center frequency for the analog signal processing path. The events generated by the VCO output will trigger the RF front-end, causing the degradation of the overall simulation performance. As previously mentioned in earlier section, the advantages of the SystemC modeling approach lies in the fact that the SystemC modules are based on the C++ classes and offers its object oriented features. As shown in Fig. 6.17, the output signal structure of the PLL can be extended to a baseband signal version without any modification of the connectivity information, by creating the signal with user-defined data types as shown in listing 6.8.

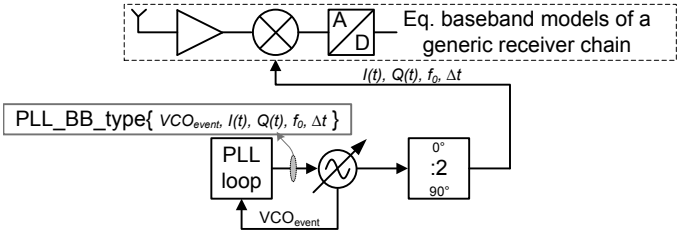



Figure 6.17: The signal structure of the VCO output.

In the new signal type `PLL_BB_type`, six parameters are assigned: `vco_event`, `f_noise`, `f_0`, `I_1`, `Q_1`, `t_step`. The signal `vco_event` is Boolean type used generate the VCO output events for the PLL feedback path. The signal `f_noise` represents the instantaneous noisy frequency. The `f_0`, `I_1`, `Q_1` are similar to the `BB_double` signal type shown in section 5.3.2. Equations 6.1 show their derivation.

$$\varphi(t) = 2\pi(f_0 - f_{noise})t \tag{6.1}$$

$$\begin{aligned} VCO_{event} &= cos(2\pi f_0 t + \varphi(t)) \qquad (6.2)\\ &= cos(2\pi f_0 t)cos(\varphi(t)) - sin(2\pi f_0 t)sin(\varphi(t)) \\ &= I(t)cos(2\pi f_0 t) - Q(t)sin(2\pi f_0 t) \qquad (6.3) \end{aligned}$$

```
1 class PLL_BB_type{
2 public:
3     bool vco_event;                          //square wave from VCO;
4     double fnoise;                           //Instanious noisy frequency ;
5     double f0;                               //Center frequency (f=ref*N) of the PLL;
6     double I1;                               //eq. BB phase noise
7     double Q1;
8     double t_step;                           //phase noise sampling time for ↩
          ↪ baseband signal
```

```
9  public:
10     PLL_BB_type(){
11         vco_event = 0;
12         fnoise = 0;
13         f0 = 0;
14         I1 = 0;
15         Q1 = 0;
16         t_step = 1e-9;                          // set a typical time step value
17     }
18     PLL_BB_type(bool vco_event_, double fnoise_, double f0_, double I1_, double Q1_, ↩
           ↪ double t_Step_){
19         vco_event = vco_event_;
20         fnoise = fnoise_;
21         f0 = f0_; I1 = I1_; Q1 = Q1_;
22         t_step=t_step_;
23     }
24     operator bool() const{                      //convert to bool type;
25         return vco_event;
26     }
27     bool operator == (const PLL_BB_type& rhs){
28         return (vco_event == rhs.vco_event);
29     }
30     PLL_BB_type & operator = (const PLL_out_type& rhs){
31         vco_event = rhs.vco_event;              //operator = overloading;
32         fnoise = rhs.fnoise;
33         f0 = rhs.f0;
34         I1 = rhs.I1;
35         Q1 = rhs.Q1;
36         t_step=rhs.t_Step;
37         return *this;
38     }
39 };
```

Listing 6.8: Signal Type of the VCO output.

For the RF receiver chain shown in Fig. 6.17, only the slow varying part of the VCO output is used for baseband models. In this case the equivalent baseband amplitudes $I(t)$, $Q(t)$, the center frequency $f_0$ and the instantaneous phase $phi(t)$ describing a potential phase modulation as well as the phase noise present in the VCO signal, respectively. For the detailed PLL model, the divider is triggered by the passband VCO oscillation event `vco_event`, in order to get the accurate response of the equivalent SystemC RF blocks in the PLL. This can be realized by overloading the `==` operator in the `PLL_BB_type` class, which permits only the changes in the `vco_event` are considered in the sensitive list. Further considerations like the interaction between the PLL output signal and RF baseband signal can be resolved by overloading the operators.

## 6.2 A RF-DAC based multi standard transmitter

### 6.2.1 Block-level of the RF-DAC based transmitter

Fig. 6.18 shows the block diagram of the RF-DAC[2] based transmitter targeting the WiFi and 3GPP(3rd generation partnership project) LTE(Long Term Evolution) band. Most part of the chip are high performance digital circuits. Starting from the baseband data(I, Q and CLK) generated from an external FPGA[3], the deserializer converts the sequential input Bits to a 12-Bit parallel signal feeding into the dynamic clock domain conversion block(CDC). The CDC resamples the input signal with the LO(local oscillator) frequency and decimates the resampled signal to an integer fraction of the RF carrier, in order to convert the signals' clock domain from off-chip to on-chip. A detailed description of this block can be found in [TZN10]. The upsampling block implemented here is intended to realize a fractional upsampling factor of $M/N$, in order to increase the possible frequency schemes.

The digital lowpass $\Delta\Sigma$ modulator(DSM) has a resolution of 9-Bits, its transfer function has been optimized both for achieving a good SNR in the wanted band and reducing the unwanted emission far from the carrier frequency. The output signals of DSM feed into the element selection logic cell(ESL), which is implemented to digitally compensate the static and dynamic mismatch error in the DAC.

The analog part of the transmitter comprises different dividers providing required frequency component in the digital part, biasing block and the RF-DAC itself. The RF-DAC consists of 16 unary unit cells for the MSBs(most significant Bit) and 15 binary unit cells for the LSBs(least significant Bit) for each I- and Q- path. Due to the ESL algorithm, the number of unary unit cells is increased to 18 and 30 for the binary unit cell respectively.

### 6.2.2 Model implementation and virtual prototyping

**Block-level model implementation** The maximum output frequency of the high performance digital part is around 870 $MHz$, while the LO operates at around 2.6 $GHz$. Due to these high frequencies, implementing analog models is in this case not feasible, especially when the high speed outputs of the DSM are going to connect the analog part directly. Hence, the proposed modeling flow shown in section 4.2.3 has to be slightly modified to adapt to the given circuit architecture:

1. Due to the large high speed digital part, the analog model in this case cannot deliver reasonable speed up for block-level integration, since the most blocks are embedded into the digital netlist. Hence, the only analog

[2]Radio frequency digital to analog converter.
[3]Field Programmable Gate Array

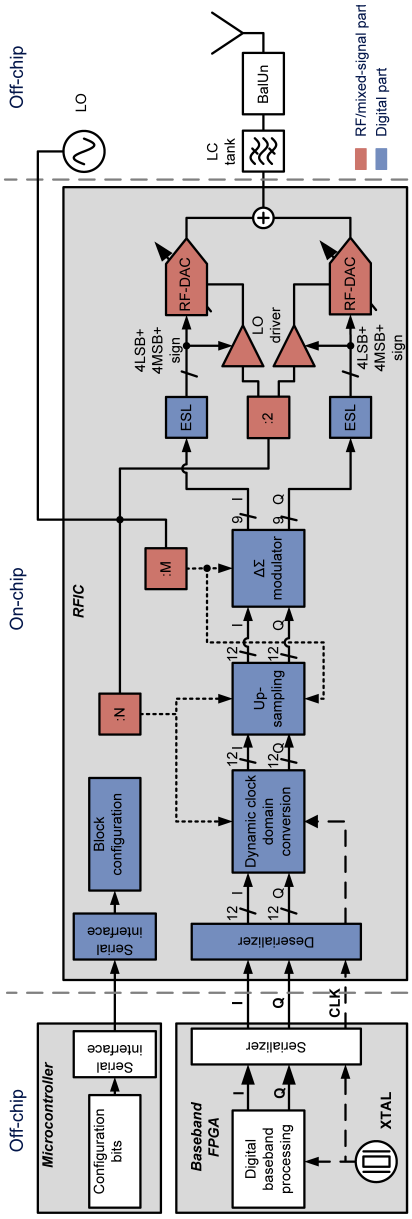



Figure 6.18: Detailed block diagram of the RF-DAC based transmitter according to [Zim11]. The digital section dominates the whole RFIC[a]. However, the RF/mixed-signal part at transistor-level consume the most simulation resource.

[a] Radio frequency integrated circuit.

models implemented here are the ideal ADCs and DACs, in oder to generate proper input stimuli for the analog front-end.

2. Fig. 6.18 also shows a large part of off-chip equipment, which is also required for the top-level simulation. Therefore, event-driven models for signal and configuration generation have to be implemented and validated upfront. In this case, MATLAB® is used to generate and to store baseband signals and configuration commands in binary form, the event-driven models then read the binary data and feed them into the DUV. A co-simulation with MATLAB® simulink is also possible, however, it will slow down the simulation due to additional synchronization between simulators.

3. The large bandwidth(around 20 $MHz$) and the high sampling frequency of the digital output make the baseband modeling approach unattractive for the RF-DAC. Additionally, the actual digital to analog conversion is taking place inside each of the DAC unit cell, making the baseband models even less feasible. Therefore, only event-driven models are implemented for the RF-DAC cells.

4. The output of the RF-DAC unit cells are in current domain, the overall amplitude is achieved by connecting all the outputs of the unit cells together, leading to the multiple drivers issue for the event-driven models mentioned in section 3.3.3. In order to overcome this issue while preserving pin-accuracy, two possible solutions are given: first, increase the level of abstraction of the RF-DAC, model only the analog sub-block and make the necessary calculation internally. Second use the global wreal resolution function for the sub-blocks, so that the outputs can be connected together and behave like a current summation node. The first solution is obviously the fastest one, but the risk of missing hierarchal connection errors is also given, which will miss the target of the functional verification. Therefore, the second solution is chosen. Since the wreal resolution function applies globally, the uniqueness of the current summation point at the RF-DAC's output has to be determined up-front, otherwise it will aggravate the functional verification for the analog front-end.

The key modeling task for the RF front-end is to implement event-driven model of the RF-DAC. Fig. 6.19 shows the internal structure of the corresponding event-driven model. Basically, the digital input signal is converted into corresponding current and multiplied with its sign. The LO signal works like a switch, which samples the signed amplitude of the DAC to the output. Besides the mentioned functional behavior of the block, assertions of the bias current, sign Bit and other digital configurations are implemented in the model. Performance parameter, such as nonlinearity and mismatch of the DAC, has been neglected during the modeling process, based on two facts: 1.) The performance of the analog front-end

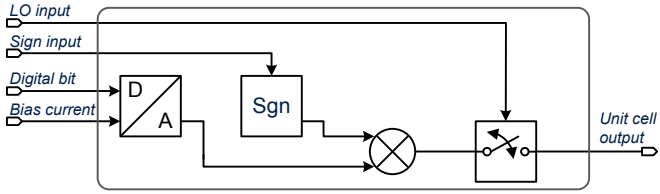



Figure 6.19: Block diagram of the event-driven RF-DAC unit cell model.

can be simulated at transistor-level with existing LTE signal. 2.)Modeling non-ideal effects will significantly slow down the top-level simulation and hence reduce the coverage of functional verification. Since both model set are implemented in pure event-driven manner and digital part dominates the design, the SystemC VP exhibits almost the same simulation performance. Hence, only the top-level simulation results of Verilog-AMS based models will be presented here.

**Top-level verification of the RF-DAC** Before moving to the top-level, two major blocks are assembled and simulated stand alone:

1. **RF-DAC analog front-end** : simulation has been done at transistor-level with Verilog-A coded OFDM[4] and single tone input stimuli for determining the analog performance.
2. **Digital part**: The overall signal block has been synthesized and partially simulated with baseband signal.

Fig. 6.20 shows the top-level schematic of the transmitter. For the top-level verification of the RF-DAC, the CML to CMOS converter and PLL models are switched to the highest level of abstraction. The PLL is only modeled as a state machine, which only generates the output signal based on the product of divider ratio and reference clock.

Besides the event-driven models of the building blocks, some peripheries for the top-level testbench are also required:

1. Frequency detector: for checking the PLL output frequency.
2. Baseband signals and configuration command: both bit sequences have been generated using Matlab®, a Verilog model is used to read the bit stream and send them to the chip.

The baseband signals contain not only typical LTE signal, but also simple single tone signals for a better traceability through the design hierarchy. Fig. 6.21 shows the top-level testbench schematic. For the top-level verification, different

[4]Orthogonal Frequency Division Multiplex.

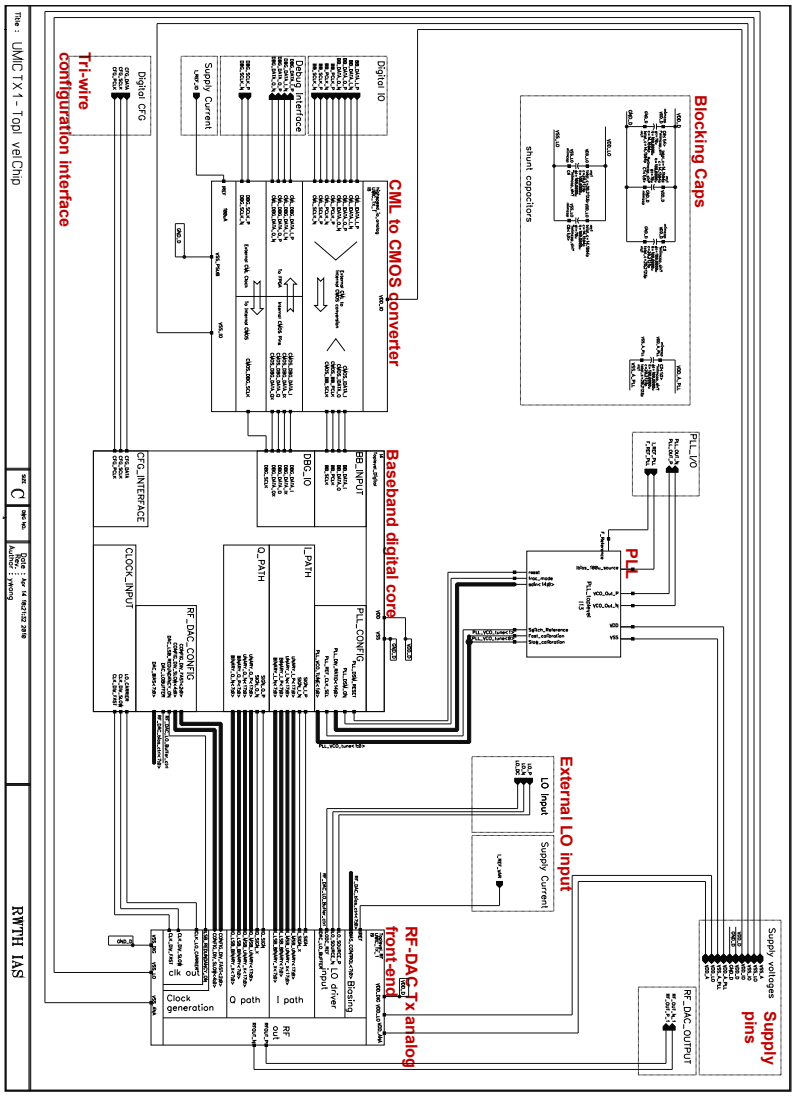



Figure 6.20: Top-level schematic of the RF-DAC based transmitter.

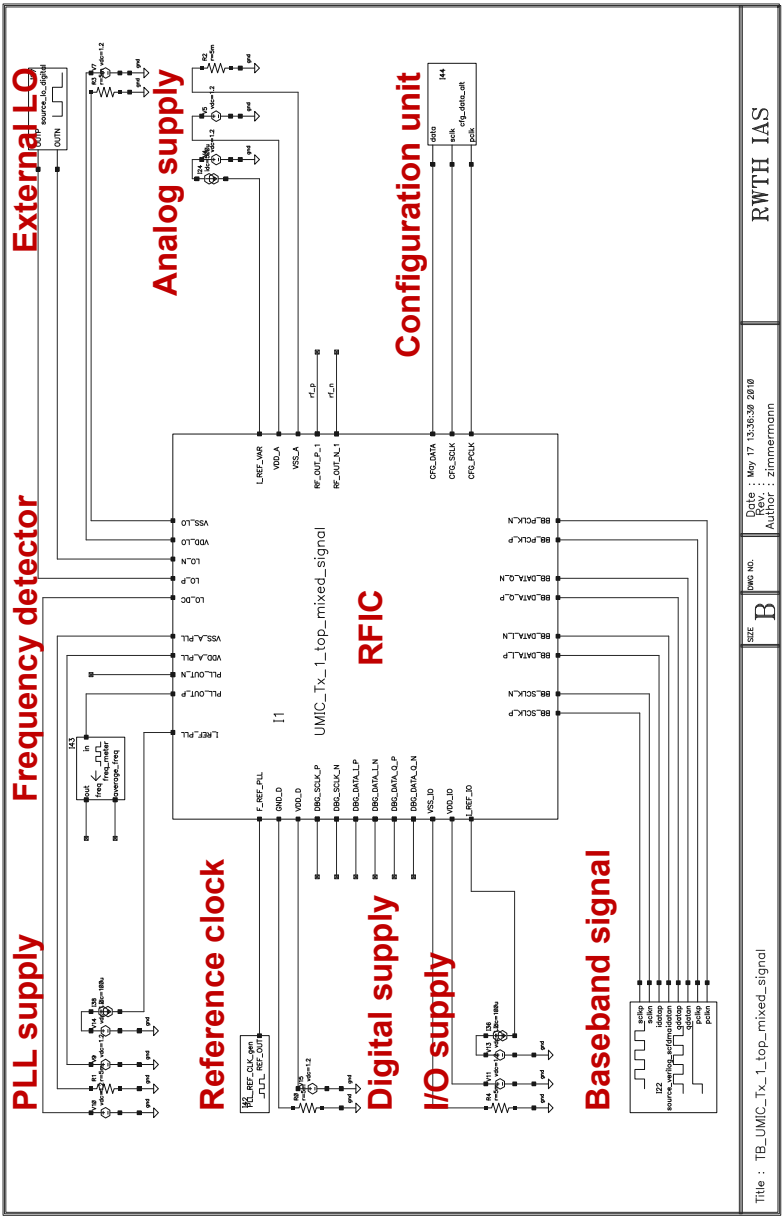



Figure 6.21: Top-level testbench of the RF-DAC based transmitter.

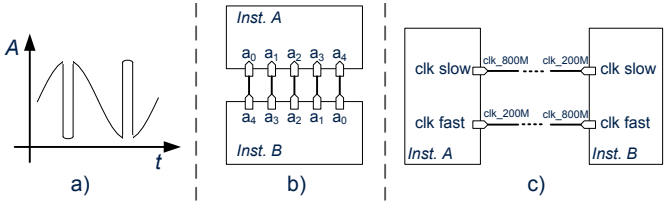



Figure 6.22: Functional errors of the RF-DAC found by the top-level verification.

configuration commands setting up the chip within several $ms$ need to be simulated. For such extensive simulations a MATLAB® pre-processor breaks the commands into 50 $\mu s$ chunks and the simulation is distributed on a cluster of workstations. In this way, results from each workstation can be analyzed and evaluated more efficiently. Fig. 6.22 shows some functional errors found out through the top-level functional verification.

By applying a pure sinusoidal baseband signal with the maximum allowed OFDM amplitude, an overflow error has been found in the digital processing part shown in Fig. 6.22 a). This error could be hardly found with typical OFDM baseband signals, since the likelihood of reaching maximum amplitude by OFDM signal is small. The assertions inside the PLL have detected the twisted connections shown in Fig. 6.22 b), since the wrong divider ratio is out of the defined range. The connect-by-name error shown in Fig. 6.22 c) has been found by an other assertion, which checks the input clock frequency.

Table 6.6 shows the transistor count for analog and digital part of the RF-DAC with their corresponding simulation time for 50 $\mu s$. The simulation has been

| **Circuit partition** | **Transistor count** | **simulation time** |
|---|---|---|
| Analog | ca. 3341 | Transistor-level: 50 $\mu s$ for 2 days |
| Digital | $4.12 \cdot 10^5$ gates | Gate-level: 50 $\mu s$ for 1.22 $H$ |
| Mixed-signal | $4.12 \cdot 10^5$ | Event-driven model+ Gate-level: 50 $\mu s$ in 1.36 $H$ |

Table 6.6: Transistor count and top-level simulation time of the RF-DAC based transmitter.

executed on a workstation with a Q9550 CPU clocked at 2.83 $GHz$ and 8 $Gb$ RAM. Based on the shown simulation time for both analog and digital part,

the overall simulation time of the transmitter for 50 $\mu s$ at transistor-level with gate-level netlist would take approximately 57 days to complete, without regard to possible mixed-signal simulator synchronization overhead and the resulting analog convergence issues. These facts again emphasize the importance and usefulness of model based top-level functional verification. The project time frame starting from top-level to tapeout is around three months, simulating the whole design at the transistor-level might indeed detect the mentioned functional errors shown in Fig. 6.22. However, the bugs cannot be corrected in time, which will result in functional failure of the chip.

Fig. 6.23 shows the simulated output spectrum of the transmitter. For this simulation, event-driven models have been applied to the RF-DAC through the circuit hierarchy. An OFDM signal with 20 $MHz$ is used at the baseband, the carrier frequency is set at 2.4 $GHz$. The transmitter is configured with its default setting, which implies that the CDC is on bypass mode, upsampling filter is operating with its default filter coefficients and the DSM uses only the first order feedback.

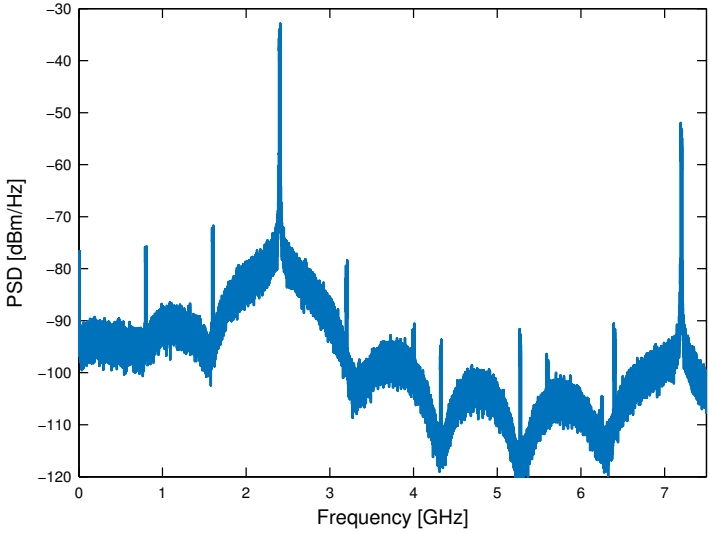
**Preliminary conclusion** Both VPs of the presented PLL and RFDAC based transmitters show no significant speed improvement compare to the event-driven passband Verilog-AMS model, due to two reasons:

1. The baseband modeling technique talked in section 3.2.3 cannot be applied to the presented RF systems, which generally forces the sampling frequency of the RF signal at fifteen times of gigahertz and hence slows down the overall simulation.
2. The development trend of the nano-scale RF transmitter architecture is clearly converging to the digital domain. The number of analog signal processing blocks in the transmitter are consistently decreasing, the remaining ones are continuously pushed towards the antenna. Simulating large digital part with high RF frequency will also rise the demand for higher simulation performance. This is also the reason. why the VP of RF-DAC doesn't show noticeable simulation speed up compare to the event-driven Verilog-AMS models.

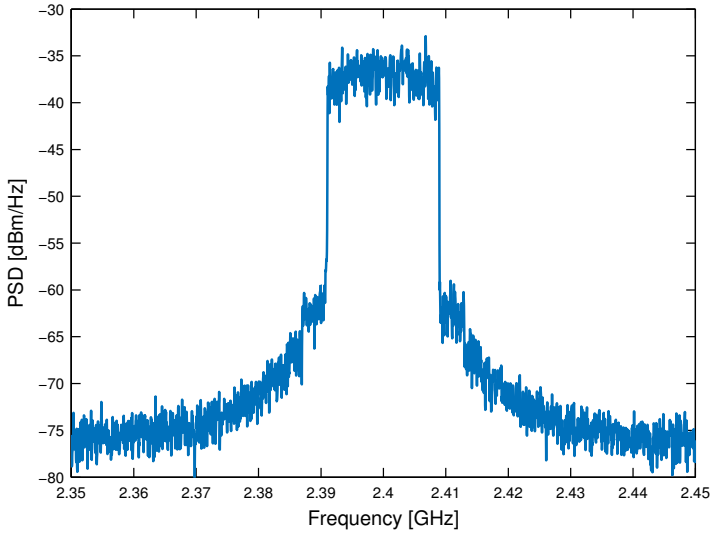
However, system with large part of analog signal processing block will benefit from the SystemC VP, as will be shown in the next section.

## 6.3 A low power Bluetooth transceiver

The experience gained in the virtual prototyping processes for the PLL and RF-DAC serves as the starting point for the low power Bluetooth (LPBT) [Blu10] transceiver system project.




(a) Wideband plot.




(b) Narrowband plot

Figure 6.23: Event-driven model based top-level simulation output spectrum of the RF-DAC transmitter.

### 6.3.1 System overview

Figure 6.24 shows the simplified block diagram of the mentioned transceiver.

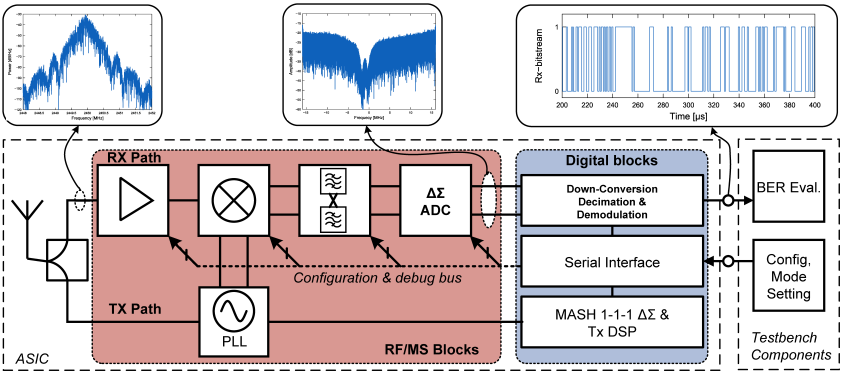



Figure 6.24: Block diagram of the low power Bluetooth transceiver chip.

For the receiver path, the analog front-end uses a low-IF[5] architecture. Following the LNA, the signal is down converted to the $-1\ MHz$ IF frequency by using the passive mixer and the PLL-driven LO, which separates the signal into its I and Q components. A $3rd$ order complex bandpass filter with the bandwidth of 1.3 MHz centered at -1 MHz rejects the image signal. The IF signal is converted into digital domain by using a $3rd$ order, single bit quadrature bandpass $\Delta\Sigma$-ADC. The output bit stream of the ADC is connected to the digital down conversion mixer in the baseband and decimated with lower sampling rate. The digital signal then will be demodulated by a quadricorrelator demodulator. For the transmitter path, a PLL based two-point modulation transmitter architecture is used. The core building block is a fractional-N PLL which will also be configured as the frequency synthesizer for the receiver path. Due to the architectural similarity to the first example, only the results of the receiver path will be described in detail.

### 6.3.2 System partitioning and model implementation

The verification environments consolidates different testbenches and evaluation tasks in different levels of abstraction. In the presented example, two verification environments has been chosen to demonstrate the proposed verification and virtual prototyping methodology: top-level and VP of the chip.

[5] low intermediate frequency

### System partitioning of the low power Bluetooth transceiver

Fig. 6.25 shows the top-level schematic of the low power Bluetooth transceiver consisting 4 large blocks: RF macro, ΔΣ-ADC, Bias block and the PLL with digital baseband. Basically, the system is also partitioned into these 4 parts du to following reasons:

1. For the RF macro partition, the most important consideration to make is to enable the steady state simulation methods for fast performance verification and model parameter extraction. One might argue for moving the ΔΣ-ADC(DSM-ADC) also into the partition. However, DSM-ADC doesn't provide the required condition for steady state analysis, which makes the analysis of the circuit and post evaluation of the simulation results difficult. Therefore, the RF macro consists of only the LNA, mixer and PPF. For the LO signal, the output signals of the I-Q divider are stored in a data file, which will be read from a PWL source. The RF macro itself has been further divided into the RF front-end part, which consists of LNA and mixer, and the IF part consisting of the PPF.

2. As previously mentioned, the DSM-ADC block has been separated from the RF macro due to the steady state simulation problem. In addition to the simulation limitation, the DSM-ADC requires very long transient system time. Only in this way, long enough bit streams can be provided for a reasonable FFT[6] result, in order to analyze the output spectra and the resulting SNR of the ADC.

3. The PLL and digital baseband are partitioned together, since the experience of PLL model implementation and reusable models are given, so the event-driven model set can be realized quickly. Additionally, the PLL in transmission mode requires tight interaction with the digital part.

4. The bias block has been put into the top-level for a better wiring possibility. Although it will not cause degradation of simulation performance, models with assertions for the BIAS are still required to check the input bias current and supplies.

### Model implementation of the low power Bluetooth transceiver

Both analog and event-driven model sets are implemented for the RF macro and the DSM-ADC according the criteria mentioned in section 4.2.3. The analog model set are used for block integration and top-level assembly, whereas the event-driven set is mainly targeting the top-level verification.

[6] Fast Fourier Transformation.

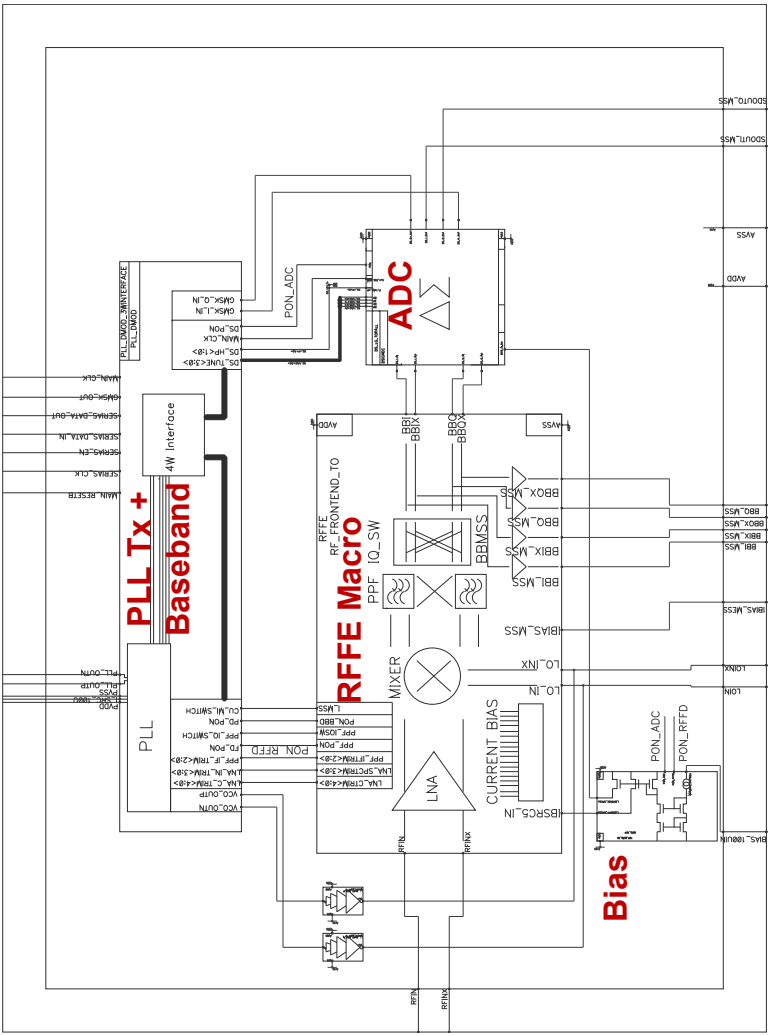



Figure 6.25: Top-level partitioning of the low power Bluetooth transceiver chip.

One note regarding the event-driven filter implementation has to be mentioned here. As equations 4.4, 4.5 and 4.6 show, a complex transfer function $H$ of a filter can be realized by two filters $R$ and $Q$ with real coefficients. While VERILOG-A provides build-in Laplace operator for continuous time filter, the equivalent filter coefficients of the event-driven model have to be converted firstly based on the analog ones [Hor63]. Basically, a discrete time IIR(Infinite Impulse Response) filter can be converted based on its continuous time counter part and implemented according to Fig. 6.26.

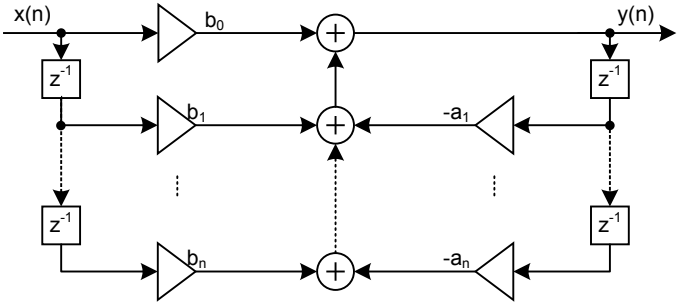



Figure 6.26: Direct-form I IIR filter.

Modeling such discrete time filter has to face two challenges. First, the feed back structure is prone to numerical errors of its coefficients. Therefore, proper transformation method has to be chosen. For the loop filter inside DSM-ADC, the *matched* transformation method is preferred [Sch03]; [Kim09]. Second, the direct-form I implementation is prone to value overflow in the summation path. Especially for complex valued transfer function, both the equivalent $R$ and $Q$ filter have the twice the order compare to the complex one, resulting in higher overflow probability. While these issues can be avoided in the PPF by dividing the filter into smaller stages according the circuit structure, it causes simulation errors in the $3rd$ order DSM-ADC model. Therefore, the PPF model implementation inside DSM-ADC uses the state-space approach shown in Fig. 6.27. This approach offers two advantages over the IIR implementation. First, state-space implementation requires half of the registers compare to the IIR one, consequently it also requires less multiplications and additions, so that the potential risk of overflow errors can be reduced. Second, the DSM-ADC has been designed using the delta-sigma toolbox [ST+05]; [Sch03], which also uses the state-space approach to describe the DSM-ADC. In this way, the coefficients of the ADC can be directly exported from MATLAB® to the models.

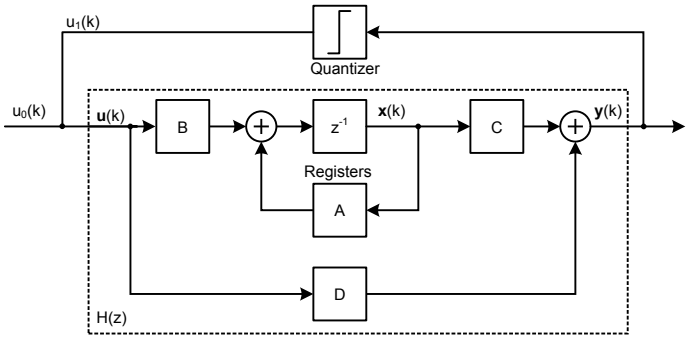



Figure 6.27: General state space description of the quadrature bandpass ΔΣ-ADC.

### 6.3.3 Top-level functional verification of the low power Bluetooth transceiver

Fig. 6.28 shows the top-level testbench of the low power Bluetooth transceiver chip. For the top-level functional verification in the circuit design environment, the examination of the end-to-end signal flow is required, in order to trace the signals crossing the whole design hierarchy. In most projects, this is the only method to prove the functional correctness of the chip prior to tapeout. Due to the complexity of system, it is not feasible to run this kind of simulation on transistor-level. The pin-accurate wreal models are mainly used for the end-to-end signal simulation of the whole chip. In this case, an ideal GFSK baseband signal is up-converted to RF and oversampled with a ratio of 20, in order to accurately resolve small amplitude changes at the receiver input. The digital part of the receiver front-end (gatelevel netlist + timing back annotation) sets all the analog blocks in the appropriate mode and demodulates the signal at the ADC output.

While sufficient for tracing the signals in all the blocks, simulations on higher abstraction levels are required in order to roughly evaluate the receiver performance in terms of the BER. Thus the pin-accurate SystemC VP of the transceiver has been implemented. The testbench of the receiver front-end VP consists of following elements: An input signal source, which is capable of generating GFSK signal in the wanted and various blocker signals at interference frequency band (not shown in figure 6.24) depending on the test case. An evaluation unit, which computes the bit error rate. A configuration unit, which contains all the configurations and mode settings of the chip for different test scenarios according to [Blu10]. Additionally, there are two simulation versions depending

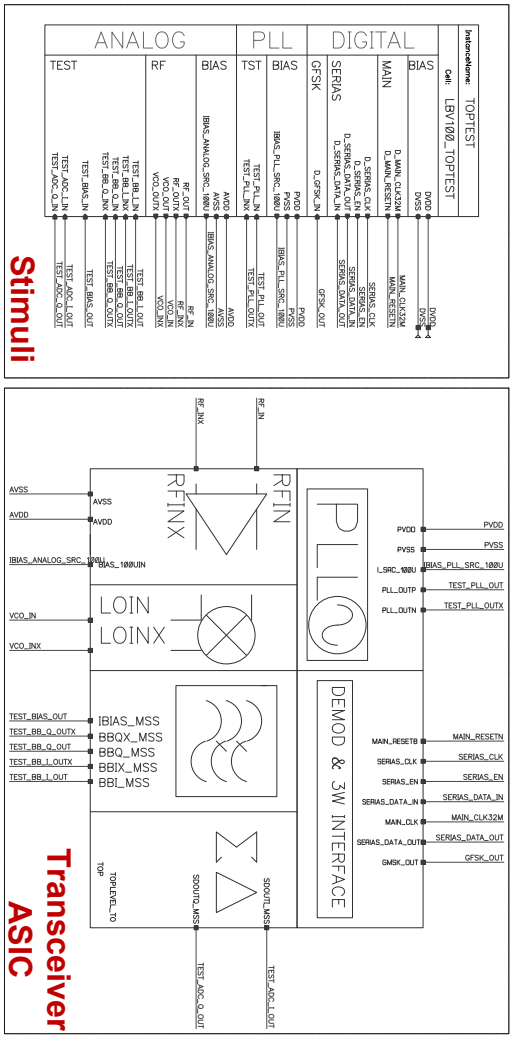
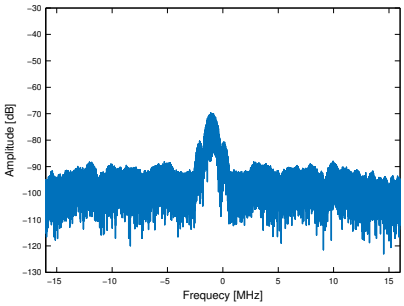
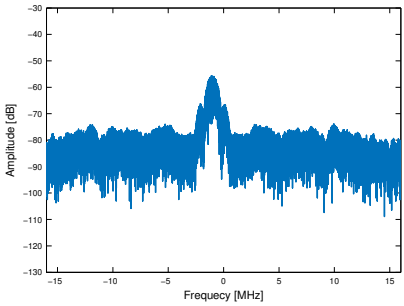



Figure 6.28: Top-level testbench of the low power Bluetooth transceiver chip.

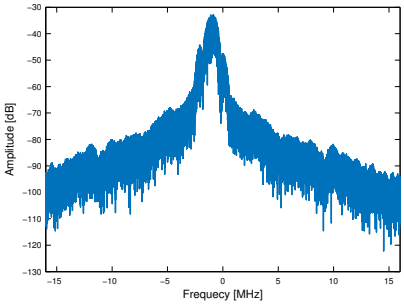
on the structure of the input signal. One is the passband version, similar to the simulation using wreal model. The other one is the equivalent baseband version, which uses the manually defined signal class `BB_double` mentioned in the section 5.3.2. Since the pin-accuracy is preserved for all simulation versions of the VP, tracing connectivity issues and signal flow between the subsystems at higher design hierarchy is also possible.



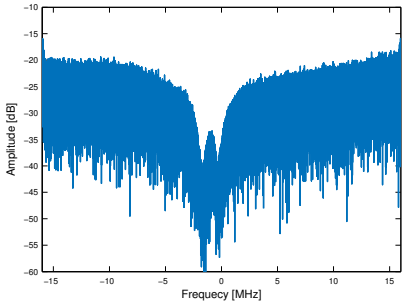

(a) Spectrum of the transmitted GFSK signal.




(b) Mixer output.




(c) Second Filter Stage.




(d) $\Delta\Sigma$ ADC output spectrum.

Figure 6.29: Simulation Results of the Bluetooth front-end Virtual Prototype. Output Spectra in the Receiver Path.

Fig. 6.29 shows the equivalent baseband simulation results of the SystemC VP. For this simulation, the wanted signal is placed at -1 MHz with a symbol rate of 1 $MSamples/s$. In figure 6.29(b), it is clearly visible that the wanted signal is properly amplified. Fig. 6.29(c) shows the wanted signal in the passband of the PPF. At the $\Delta\Sigma$ ADC output(Fig. 6.29(d)), the typical quantization noise

shaping curve is visible. Since the PPF and the ADC are designed from different team members, it is very important to ensure that there is no twisted I/Q signal paths due to wiring errors or mis-interpreted specs. The signal is down-converted to the baseband in the digital block and passes through a decimation filter and processed by an FIR filter before the actual demodulation is performed.

**Top-level start-up and Bluetooth LE standard verification** The top-level start-up verification has been done based on the VERILOG-AMS sign-off models. The simulation environment consists of not only the wreal models of the low power Bluetooth front-end, but also the digital gate-level netlist with timing back-annotation. Table 6.7 shows the specified start-up timing diagram of the transceiver. Due to the language limitation of VERILOG-AMS, the wanted passband signal is sampled with around 80 $GHz$, resulting in the main bottleneck of the simulation performance. Hence, the PLL lock-in time has been neglected at the top-level, in order to speed up the simulation a little.

| Processes | Timing requirement |
|---|---|
| Digital supply on | $0 - 10\mu s$ |
| Chip global reset | $10 - 30\mu s$ |
| Analog supply on | $30 - 40\mu s$ |
| Power up RF front-end | $40 - 76\mu s$ |
| Power up DSM-ADC | $76 - 102\mu s$ |
| Set PLL divider ratio | $102 - 138\mu s$ |
| Set up IQ switch | $138 - 164\mu s$ |
| Release PLL reset | $164 - 190\mu s$ |
| PLL supply on | $190 - 200\mu s$ |
| PLL lock-in | $200 - 270\mu s$ |

Table 6.7: Timing diagram for the full chip power up.

Fig. 6.30 shows the transmitted and demodulated Bit-stream. In this case, the power of the transmitted wanted signal has been set to $-66\ dBm$. One can observe the path delay between the transmission and demodulation.

The Bluetooth LE[7] verification has been done in the SYSTEMC VP environment, in order to exploit the simulation performance of the pin-accurate baseband event-driven models. For the Bluetooth LE standard test, a PRBS(Pseudo Random Bit Stream) source generates the random bit stream[8] according to the regulations mentioned in [Blu10].

[7] Bluetooth Low Energy standard according to [Blu10].
[8] Basically, the PRBS is generated using the LFSR shown in section 6.1.1.

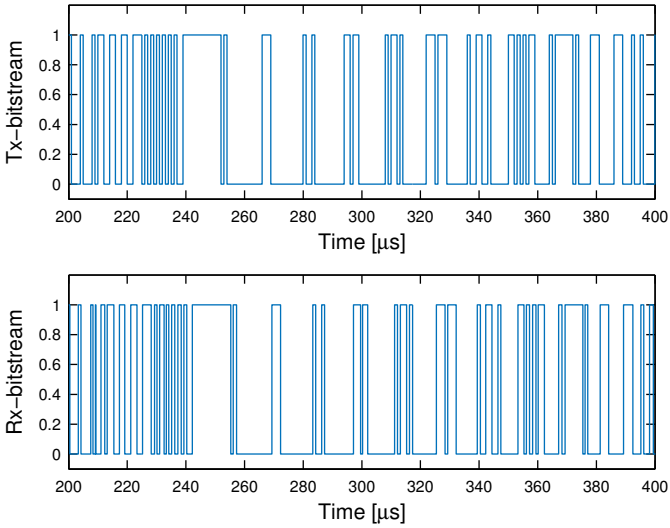



Figure 6.30: Bit-stream of the transmitted and demodulated GFSK signal.

Table 6.8 shows the simulation time of 1 ms at different levels of abstraction, it can be observed that the pin-accurate SystemC VP exhibits the best simulation performance. Since the model generation is automatized, the planned verification cases for the whole chip in different conditions can be set up and distributed to the workstation cluster.

| Receiver front-end model | Simulation time |
|---|---|
| Transistor-level | 723800$s$ |
| Analog model(Verilog-A) | 26400$s$ |
| Wreal model (Verilog-AMS) | 376$s$ |
| SystemC AMS VP | 9.2$s$ |

Table 6.8: Simulation times of the low power Bluetooth front-end for 1 $ms$ at different levels of abstraction

Table 6.9 shows part of the BER simulation results according to the Bluetooth LE standard [Blu10], which specified, that the maximum BER in all test cases cannot exceed 0.1%. For each of the simulation, run time between 10 $ms$ to 100 $ms$ are required in order to evaluate the BER. As the result overview in table

6.9 shows, most of the required verification cases in terms of the BER based on the standard have been passed.

**Achieved level of confidence** With the help of the SystemC VP, all control connections have been traced and verified to the block/subsystem level hierarchy during the simulation. End-to-end signal flow simulation in various operation modes, including most test scenarios according to the Bluetooth LE standard, such as interference and receiver sensitivity tests, can be done with very high simulation efficiency.

The digital subsystem has been simulated with the front-end VP, so that every control signal for the front-end and the demodulator can be verified from the RTL to final Verilog netlist with timing back annotation. Some timing issues regarding the digital clock can be detected.

The twisted I/Q path between DSM-ADC and the RF front-end has been found out and during the top-level simulation. Furthermore, during the block-level model validation, the mismatch in the dynamic range between ADC and front-end has been detected.

All the mentioned issues has been corrected before tapeout. The final top-level has been verified through numbers of regression runs. It can be expected, that the chip is functional.

| Test | Variant | BER Result | *Wanted* Power | *Interferer* Freq. offset | *Interferer* Power | *Single Tone* Freq. | *Single Tone* Power |
|---|---|---|---|---|---|---|---|
| Minimal Sensitivity | Normal | 0.0572% | -70 dBm | – | – | – | – |
| Maximum Input Level | | 0.0% | -10 dBm | – | – | – | – |
| Co-Channel Interference | | 0.0% | -67 dBm | 2.41 GHz | -88 dBm | – | – |
| Adjacent Interference | 1 MHz | 0.0% | -67 dBm | +1 MHz | -82 dBm | – | – |
| | 2 MHz | 0.0096% | -67 dBm | +2 MHz | -50 dBm | – | – |
| Image Interference | | 0.0% | -67 dBm | -2 MHz | -58 dBm | – | – |
| | Adj. -1MHz | 0.0% | -67 dBm | -3 MHz | -52 dBm | – | – |
| Intermodulation | n=3 | 0.0% | -64 dBm | +6 MHz | -50 dBm | 2.413 GHz | -50 dBm |
| | n=4 | 0.0% | -64 dBm | +8 MHz | -50 dBm | 2.414 GHz | -50 dBm |
| | n=5 | 0.0% | -64 dBm | +10 MHz | -50 dBm | 2.415 GHz | -50 dBm |

Table 6.9: Part of the Bluetooth LE compliance simulation results, test cases are selected according to [Blu10].

# 7

# Conclusions

In this thesis the author's contribution to the hierarchical modeling and virtual prototyping methodologies for the functional verification of RF-SoC has been presented.

First, the basic verification techniques and simulation methods are classified and the simulation-based verification described in detail. The main challenges of the simulation-based functional verification for RF-SoCs is identified; namely, the rapidly rising functional complexity of the RF front-end that comes from today's multi-mode, multi-standard communication and connectivity requirements. As a consequence, there is currently no simulator capable of simulating such a complex RF SoC design at transistor-level within an acceptable simulation time. The analog simulator with its iterative way of solving non-linear DAEs according to the circuits' behavior has been identified as the major bottleneck for the functional verification.

From the above-mentioned challenges it is found that the only way to combat the high complexity and tremendous simulation time of the RF-SoCs is to improve and formalize the existing design and verification flow. This can be done by hierarchically decomposing the large design into small but more manageable partitions and implementing models at different levels of abstraction through the whole design hierarchy. Especially for the RF subsystem, the only way to get reasonable simulation performance for the functional verification is to verify the whole system in a pure digital simulator using event-driven RF models.

For model implementation targeting high simulation performance while keeping the consistency of the design database, it is desirable to find a suitable HDL

that is capable of modeling the analog/RF part of the design in the event-driven domain and keeping the pin-accuracy through all the levels of abstraction. Therefore, different HDLs have been evaluated and compared regarding the mentioned requirements. Based on the results of this comparison it is apparent that the only way to handle a broad range of abstraction levels in the design hierarchy, while keeping the consistency of the design database, is to combine the advantages of different HDLs while consistently improving the verification and model refinement flow. Hence, two HDLs, namely Verilog-AMS and SystemC, have been selected to demonstrate the proposed hierarchical verification and virtual prototyping methodologies.

In a following step the key concepts in the hierarchical model implementation and verification in the design environment are presented. First, the critical path for model generation in the verification flow proposed by [CK07] has been identified. Afterwards, a novel methodology including design partitioning, hierarchical model implementation, assertion coding, as well as the automated model parameter extraction and update is introduced. The proposed methodology mainly aims to improve the model generation and validation process. It is noticed that the model refinement process has to be automated as much as possible in order to avoid the human error factor that is introduced during the model refinement process. Hence, the verification team has to accompany the design team from the early project phase based on the verification flow proposed in this work. Starting with block-level design, besides implementing and validating the analog models side-by-side in the same simulator with the circuit, the verification team has to identify the key parameters of the model and to translate the testbenches into equivalent SKILL/OCEAN script for automated execution. Most details of the block-level circuits, especially the start-up time, condition and sequence have to be documented in the model for a better debugging process later on. With the help of automatically executed testbenches, key model parameters can be extracted regularly; most of these parameters can be automatically applied to the models via a SKILL based CDF parameter update routine. Although the testbench script translation and the additional SKILL based parameter update routine indeed increase the initial workload of the verification team, it pays off for the model refinement in the later project phase. Due to the language constraints in the current Verilog-AMS version, a pin-accurate baseband model cannot be realized. Hence, the usage of wreal data type and the resulting event-driven models are considered as the highest level of abstraction in the circuit design environment. The event-driven wreal models indeed increase the simulation performance for the RF front-end by shifting the whole design into the digital simulator. However, it also shows the limitation of the classical HDL in terms of simulation performance, modeling capabilities, and reachable levels of abstraction. For example, the RF signals in the wreal models are sampled at the passband, resulting in sampling frequencies in the range of 80

~100 GHz depending on the actual signal bandwidth and its modulation, which greatly degrades the digital simulation performance. The only way to reach a higher level of abstraction, and therefore increase the simulation performance further, is to introduce new modeling language and formalisms while preserving the consistency of the design database at a higher level of design hierarchy.

Finally, a SystemC-based RF virtual prototyping methodology, targeted at fulfilling the abovementioned modeling requirements, is introduced. The whole flow consists of two parts:

1.) Automated generation of pin-accurate SystemC model frame based on the schematic information. The whole process is implemented in SKILL language, which is compatible with the current circuit design environment. Starting from the top-level schematic, the hierarchical structure of the whole design will be analyzed and stored in an intermediate file. Each element of the design will be translated into its corresponding SystemC model frame based on the schematic information; e. g. pin, port I/O direction, and instance name. Subsequently, naming conventions and resolution functions are implemented while connecting the SystemC model frames according to the schematic. The corresponding names for instances, pins, and nets will be converted or expanded to fit the naming rules in SystemC syntax. In order to resolve some analog wiring issues, e.g. multiple ports driving a single node, which are common in the analog but not allowed for the digital simulation, the resolution function will remap the connectivity of the SystemC models depending on the signal taken into consideration for the model(current or voltage). At the end of this step a pin-accurate SystemC netlist based on the RF schematic is created.

2.) A SystemC RF library that contains a model set of basic RF components has also been implemented. Each model in the library mainly targets to describe the core functionalities and behavior, e. g. parameterizable gain, noise and linearity, in order to maintain its flexibility. The model parameters can then be updated from the parameter database shown in section 4.2.5. They can be directly instantiated in the model frames in step 1. The only manual modification for completing the modes is to map the digital control states and settings for power on/off, bandwidth or gain switching. With the advantage of the C-based HDL, a user defined complex data type "BB_double" is implemented. This not only enables the seamless switching between the baseband and passband simulation with the same model through operator overloading, but also enables equivalent baseband simulation using multiple sets of baseband components (s. Fig. 3.8), with the highest simulation performance.

The proposed modeling and virtual prototyping methodology has been demonstrated through three different application examples from different projects. For

the PLL system the SystemC VP shows, on the one hand, matched phase noise and lock-in simulation results compared to the Verilog-AMS models. On the other hand, it exhibits the best simulation performance, especially in conjunction with the usage of user defined PLL signal: *PLL_BB_type* shown in Fig. 6.17 for analyzing the modulated GFSK signal of a PLL-based polar transmitter. In the RF-DAC based transmitter example, the model based simulation performance is around 1000 times higher than the simulation at the transistor-level. In this way different functional bugs(s. Fig. 6.22), which could be a show-stopper for the chip, can be detected and corrected before the tapeout. However, the difference in simulation time between the VPs of both mentioned systems and their Verilog-AMS models are not significant due to the large portion of high frequency digital part in the system. The receiver part of the Bluetooth transceiver system consists of a large portion of analog and RF subsystem with the digital part operating at comparatively low frequency. The simulation time of the SystemC VP with baseband data type only takes 9.2 seconds, which is 7.8e4 times faster than the transistor-level simulation. The VP simulation enables the end-to-end signal flow simulation in various operation modes, while verifying all the control connections down to the block level hierarchy so that the twisted I/Q path issue and some digital timing issue in conjunction with the RF front-end mentioned in section 6.3.3 can be fixed before tapeout. In addition to that, pre-silicon simulation based BER tests according to [Blu10] can be executed in time, which significantly increases the level of confidence for the tapeout.

All three mentioned examples are taped out and the measurements show no functional errors in the chip, which confirms that the proposed modeling and VP methodology can speed up and enhance the functional verification of the advanced nano-scale CMOS RF subsystems.

## 7.1 Outlook

Although already available, the graphical user interface of the proposed VP can be improved further in order to provide better overview of the circuit hierarchy and thus reduce possible errors introduced during the VP process. Furthermore, it is desirable not only to verify the chip's functionality, but also the correctness of the measurement setup before it goes to the hardware. In this way a verification platform with virtual instruments and the SoC's VP can be built up on a unified verification environment. On top of that the vanishing boundary between the hardware and software, due to the complexity of today's SoCs, requires an overall system simulation. This includes not only the hardware, but also the interaction between the firmware and the pin-accurate hardware VP. This is required in order to increase the confidence level of the design and shorten the development of the overall system. This can be done by unifying both the RF-SoC and its firmware verification environments.

# Bibliography


[3gp] *3GPP TS45.005 V7.9.00: Radio transmission and reception*. Technical Specification Group GSM/EDGE, Feb. 2007.

[Abi06] A. Abidi. „Phase noise and jitter in CMOS ring oscillators“. In: *Solid-State Circuits, IEEE Journal of* 41.8 (2006), pp. 1803–1816.

[AC06] D. Automation and S. Committee. „IEEE Standard SystemC Language Reference Manual“. In: *IEEE Computer Society* 2002.March (2006), 1666–2005. URL: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1617814.

[AD+90] E. Acuna, J. Dervenis, et al. „Simulation techniques for mixed analog/digital circuits“. In: *Solid-State Circuits, IEEE Journal of* 25.2 (1990), pp. 353–363.

[AJK05] H. Al-Junaid and T. Kazmierski. „Analogue and mixed-signal extension to SystemC“. In: *Circuits, Devices and Systems, IEE Proceedings* - (Dec. 2005), pp. 682 –690. ISSN: 1350-2409. DOI: doi:10.1049/ip-cds:20045204.

[Ake78] S. Akers. „Binary decision diagrams“. In: *Computers, IEEE Transactions on* 100.6 (1978), pp. 509–516.

[Ams] *Virtuoso Multi-Mode Simulation*. Datasheet. Cadence Design Systems, 2011. URL: http://www.cadence.com/rl/resources/datasheets/virtuoso_mmsim.pdf.

[Bar90] T. Barnes. „SKILL: a CAD system extension language“. In: *Design Automation Conference, 1990. Proceedings., 27th ACM/IEEE*. June 1990, pp. 266 –271. DOI: 10.1109/DAC.1990.114865.

[BD+09] D. C. Black, J. Donovan, et al. *SystemC: From the Ground Up, Second Edition*. Springer, 2009. ISBN: 0387699570.

[Bda] *Analog FastSPICE Platform*. Berkeley Design Automation. URL: http://www.berkeley-da.com/prod/datasheets/Berkeley_DA_Platform_DS.pdf.

[Ber03] J. Bergeron. *Writing Testbenches: Functional Verification of HDL Models*. 2nd ed. Springer US, Feb. 2003. ISBN: 1402074018.

[Ber06] J. Bergeron. *Writing testbenches using system Verilog*. Springer-Verlag New York Inc, 2006.

[BH+94] J. Buck, S. Ha, et al. „Ptolemy: A framework for simulating and prototyping heterogeneous systems“. In: (1994).

[BKe08] C. Baier, J. Katoen, and I. ebrary. *Principles of model checking*. Vol. 950. MIT press, 2008.

[BM58] G. E. P. Box and M. E. Muller. „A Note on the Generation of Random Normal Deviates“. In: *Annals Of Mathematical Statistics* 29.2 (1958), pp. 610–611.

[BN+05] F. Bouchhima, G. Nicolescu, et al. „Discrete-continuous simulation model for accurate validation in component-based heterogeneous SoC design“. In: *Rapid System Prototyping, 2005.(RSP 2005). The 16th IEEE International Workshop on*. IEEE. 2005, pp. 181–187.

[Boo54] G. Boole. *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*. Vol. 2. Walton and Maberly, 1854.

[Bry86] R. Bryant. „Graph-based algorithms for boolean function manipulation“. In: *Computers, IEEE Transactions on* 100.8 (1986), pp. 677–691.

[BS+11] M. Barnasconi, N. Semiconductors, et al. „Advancing the SystemC Analog/Mixed-Signal (AMS) Extensions“. In: (2011).

[BW03] J. Butcher and J. Wiley. *Numerical methods for ordinary differential equations*. Vol. 2. Wiley Online Library, 2003.

[CA03] L. Charest and E. Aboulhamid. „A VHDL/SystemC comparison in handling design reuse“. In: *System-on-chip for real-time applications* (2003), p. 41.

[CD+10] E. Cerny, S. Dudani, et al. *The power of assertions in systemverilog*. Springer Verlag, 2010.

[CE+97] H. Chang, C. E., et al. *A top-down constraint-driven design methodology for analog integrated circuits*. Springer, 1997.

[CGP00] E. Clarke, O. Grumberg, and D. Peled. *Model checking*. The MIT Press, 2000. ISBN: 0262032708.

[Che05] J. Chen. „Modeling RF systems“. In: *The Designer's Guide Community, Tech. Rep* 2005 (2005), pp. 1–41.

[Che09] J. E. Chen. „A Modeling Methodology for Verifying Functionality of a Wireless Chip“. In: *Behavioral Modeling and Simulation Workshop, 2009. BMAS 2009. IEEE*. 2009, pp. 96–101. DOI: 10.1109/BMAS.2009.5338892.

[Che10] J. E. Chen. Pat. 20100286807 (Sunnyvale, CA, US). Nov. 2010.

[CK07] H. Chang and K. Kundert. „Verification of Complex Analog and RF IC Designs“. In: *Proceedings of the IEEE* 95 (2007), pp. 622–639. ISSN: 0018-9219.

[CM+92] E. Charbon, E. Malavasi, et al. „A constraint-driven placement methodology for analog integrated circuits“. In: *Proc. IEEE CICC*. Vol. 28. Citeseer. 1992, pp. 1–4.

[Cot90] R. Cottrell. „Event-driven behavioural simulation of analogue transfer functions“. In: *Design Automation Conference, 1990. EDAC. Proceedings of the European*. IEEE. 1990, pp. 240–243.

[CR88] L. R. Carley and R. A. Rutenbar. „How to automate analog IC designs“. In: *IEEE Spectrum* 25.8 (1988), pp. 26–30. DOI: 10.1109/6.7160.

[CS+06] E. Cheung, P. Satapathy, et al. „Runtime deadlock analysis of SystemC designs“. In: *High-Level Design Validation and Test Workshop, 2006. Eleventh Annual IEEE International*. IEEE. 2006, pp. 187–194.

[DC05] T. Dastidar and P. Chakrabarti. „A verification system for transient response of analog circuits using model checking“. In: *VLSI Design, 2005. 18th International Conference on*. IEEE. 2005, pp. 195–200.

[DG03] R. D. D. Grosse G. Fey. „Modeling multi-valued circuits in SystemC“. In: *Multiple-Valued Logic, 2003. Proceedings. 33rd International Symposium on*. 2003.

[DH+08] M. Damm, J. Haase, et al. „Bridging MoCs in SystemC specifications of heterogeneous systems“. In: *EURASIP J. Embedded Syst.* 2008 (2008), 7:1–7:16. ISSN: 1687-3955. DOI: http://dx.doi.org/10.1155/2008/738136. URL: http://dx.doi.org/10.1155/2008/738136.

[DPR96] C. Dawson, S. Pattanam, and D. Roberts. „The Verilog Procedural Interface for the Verilog Hardware Description Language“. In: *Verilog HDL Conference, 1996. Proceedings., 1996 IEEE International*. Feb. 1996, pp. 17 –23. DOI: 10.1109/IVC.1996.496013.

[Dre04] R. Drechsler. *Advanced Formal Verification*. Norwell, MA, USA: Kluwer Academic Publishers, 2004. ISBN: 1402077211.

[DS+94] S. Donnay, K. Swings, et al. „A methodology for analog high-level synthesis“. In: *Custom Integrated Circuits Conference, 1994., Proceedings of the IEEE 1994*. IEEE. 1994, pp. 373–376.

[FH+05] R. Frevert, J. Haase, et al. *Modeling and Simulation for RF System Design*. 1st ed. Springer, 2005, p. 291. ISBN: 0387275843.

[FM98] D. FitzPatrick and I. Miller. *Analog Behavioral Modeling with the Verilog-A Language*. 1.0. Springer, Sept. 1998. ISBN: 9780792380443.

[FO00] P. Frey and D. O'Riordan. „Verilog-AMS: Mixed-signal simulation and cross domain connect modules“. In: *Behavioral Modeling and Simulation, 2000. Proceedings. 2000 IEEE/ACM International Workshop on*. IEEE. 2000, pp. 103–108.

[Fre80] H. Freeman. *Discrete-time systems: an introduction to the theory*. Robert E. Krieger, 1980.

[GL+02] T. Grötker, S. Liao, et al. *System design with SystemC*. Springer, 2002.

[Gro02] T. Grotker. *System Design with SystemC*. Norwell, MA, USA: Kluwer Academic Publishers, 2002. ISBN: 1402070721.

[GST01] S. Gottlieb, C. Shu, and E. Tadmor. „Strong stability-preserving high-order time discretization methods“. In: *SIAM review* (2001), pp. 89–112.

[Har03] J. Harrison. „Formal Verification at Intel“. In: *Proc. 18th Annual IEEE Symp. Logic in Computer Science*. 2003, pp. 45–54. DOI: 10.1109/LICS.2003.1210044.

[HC+09] A. Hadjichristos, M. Cassia, et al. „Single-chip RF CMOS UMTS/EGSM transceiver with integrated receive diversity and GPS“. In: *Solid-State Circuits Conference-Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*. IEEE. 2009, pp. 118–119.

[HC09] W. Hartong and S. Cranston. *Real Valued Modeling for Mixed Signal Simulation*. Application Note. Cadence Design Systems, Inc., Jan. 2009.

[Hei92] S. J. Heinen. *CAD-Verfahren für die Signal-und Rauschanalyse analoger linearer und nichtlinearer Schaltungen im eingeschwungenen Zustand*. 1992.

[HLL99] A. Hajimiri, S. Limotyrakis, and T. Lee. „Jitter and phase noise in ring oscillators". In: *Solid-State Circuits, IEEE Journal of* 34.6 (1999), pp. 790–804.

[HMW07] S. Huang, H. Ma, and Z. Wang. „Modeling and Simulation to the Design of Fractional-N Frequency Synthesizer". In: *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*. 2007, pp. 1–6. DOI: {10.1109/DATE.2007.364606}.

[Hor63] I. Horowitz. *Synthesis of feedback systems*. Vol. 1663. Academic Press New York, 1963.

[HR04] M. Huth and M. Ryan. *Logic in Computer Science: Modelling and reasoning about systems*. Cambridge University Press, 2004.

[HRB75] C. Ho, A. Ruehli, and P. Brennan. „The modified nodal approach to network analysis". In: *Circuits and Systems, IEEE Transactions on* 22.6 (1975), pp. 504–509.

[HW10] S. Heinen and R. Wunderlich. *Grundlagen der Schaltungstechnik I*. Institut of Integrated Analog Circuits and RF systems, 2010.

[HY+06] J. He, J. Yang, et al. „System-Level Time-Domain Behavioral Modeling for A Mobile WiMax Transceiver". In: *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International*. IEEE. 2006, pp. 138–143.

[ICT11] ICT. „ICT FACTS AND REPORTS". In: *ITU Telecom* (2011). URL: http://www.itu.int/ITU-D/ict/facts/2011/material/ICTFactsFigures2011.pdf.

[IEE90] IEEE. „IEEE Standard Glossary of Software Engineering Terminology". In: *IEEE Std 610.12-1990* (1990), p. 1. DOI: 10.1109/IEEESTD.1990.101064.

[JBS00] M. C. Jeruchim, P. Balaban, and K. S. Shanmugan. *Simulation of Communication Systems: Modeling, Methodology and Techniques*. 2nd ed. Springer, Oct. 2000, p. 924. ISBN: 0306462672.

[JGH07] S. Joeres, H.-W. Groh, and S. Heinen. „Event driven analog modeling of RF frontends". In: *Behavioral Modeling and Simulation Workshop, 2007. BMAS 2007. IEEE International*. 2007. DOI: 10.1109/BMAS.2007.4437523.

[JH06] S. Joeres and S. Heinen. „Functional Verification of Radio Frequency SoCs using Mixed-Mode and Mixed-Domain Simulations". In: *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International*. 2006. DOI: 10.1109/BMAS.2006.283485.

[JNT90] C. Johnson, Y. Nie, and V. Thomée. „An a posteriori error estimate and adaptive timestep control for a backward Euler discretization of a parabolic problem". In: *SIAM journal on numerical analysis* (1990), pp. 277–291.

[Joe08] S. Joeres. „Systemsimulationen zur funktionalen Verifikation von HF- und Mixed-Signal-Schaltungen“. PhD in Electrical Engineering. RWTH Aachen University, Germany, 2008, p. 176.

[JPB10] D. O. John Pierce and P. K. Bhattacharya. „Mixed-Signal Assertion Based Verification“. In: *ARM Technology Conference*. 2010.

[KC+00] K. Kundert, H. Chang, et al. „Design of mixed-signal systems-on-a-chip“. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 19.12 (2000), pp. 1561–1571.

[KC09a] K. Kundert and H. Chang. „Verifying all of an SOC-analog circuitry included“. In: *IEEE Solid-State Circuits Magazine* 1.4 (2009), pp. 26–32. ISSN: 1943-0582. DOI: 10.1109/MSSC.2009.933430.

[KC09b] K. Kundert and H. Chang. „Verifying all of an SOC-analog circuitry included“. In: *Solid-State Circuits Magazine, IEEE* 1.4 (2009), pp. 26–32. ISSN: 1943-0582. DOI: 10.1109/MSSC.2009.933430.

[KG95] K. Kundert and P. Gray. *The Designer's Guide to SPICE and SPECTRE*. Kluwer Academic Publishers, 1995.

[Kim09] S.-B. Kim. „A Contribution to Continuous-Time Quadrature Bandpass Sigma-Delta Modulators for Low-IF Receivers“. PhD in Electrical Engineering. RWTH Aachen University, Germany, 2009, p. 149.

[Kir47] G. Kirchhoff. „Ueber die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Vertheilung galvanischer Ströme geführt wird“. In: *Annalen der Physik* 148.12 (1847), pp. 497–508.

[Kun03] K. Kundert. „Predicting the phase noise and jitter of PLL-based frequency synthesizers“. In: *Phase-Locking in High-Performance Systems* (2003), pp. 46–69.

[Kun05] K. Kundert. „Challenges in RF simulation“. In: *Radio Frequency integrated Circuits (RFIC) Symposium, 2005. Digest of Papers. 2005 IEEE*. 2005, pp. 105–108. ISBN: 1529-2517.

[Kun06] K. Kundert. „Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers“. In: *Designers Guide Community* (2006).

[KZ04] K. Kundert and O. Zinke. *The Designer's Guide to Verilog-AMS*. first. Kluwer Academic Publishers, Boston, May 2004, p. 270. ISBN: 1402080441.

[Lam08] W. K. Lam. *Hardware Design Verification: Simulation and Formal Method-Based Approaches*. 1st. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2008. ISBN: 0137010923, 9780137010929.

[LP+03] K. Lee, B. Park, et al. „Phase frequency detectors for fast frequency acquisition in zero-dead-zone CPPLLs for mobile communication systems“. In: *Solid-State Circuits Conference, 2003. ESSCIRC'03. Proceedings of the 29th European*. IEEE. 2003, pp. 525–528.

[Lyo10] R. G. Lyons. *Understanding Digital Signal Processing (3rd Edition)*. 3rd ed. Prentice Hall, Nov. 2010. ISBN: 9780137027415.

[Meh10] S. Mehndiratta. „Solutions Mitigate Mixed-Signal SoC Implementation Headaches“. In: *Electronic Design* (2010). URL: http://electronicdesign.com/article/design-solutions/solutions_mitigate_mixed_signal_soc_implementation_headaches.aspx.

[Mey03] S. J. Meyer. „MIXED SIGNAL SIMULATION“. Pat. EP1305765. May 2003.

[Mey04] A. Meyer. *Principles of Functional Verification*. illustrated edition. Bertrams Print on Demand, Apr. 2004. ISBN: 0750676175.

[MG08] E. Martens and G. Gielen. *High-level modeling and synthesis of analog integrated systems*. Springer Verlag, 2008.

[MG96] T. Murayama and Y. Gendai. „A top-down mixed-signal design methodology using a mixed-signal simulator and analog HDL“. In: *Design Automation Conference, 1996, with EURO-VHDL'96 and Exhibition, Proceedings EURO-DAC'96, European*. IEEE. 1996, pp. 59–64.

[MM04] P. Molitor and J. Mohnke. *Equivalence checking of digital circuits: fundamentals, principles, methods*. Springer, 2004.

[MMS07] K. Muhammad, T. Murphy, and R. Staszewski. „Verification of Digital RF Processors: RF, Analog, Baseband, and Software“. In: *Solid-State Circuits, IEEE Journal of* 42.5 (May 2007), pp. 992 –1002. ISSN: 0018-9200. DOI: 10.1109/JSSC.2007.894327.

[MSPM03] P. Mak, U. Seng-Pan, and R. Martins. „A front-to-back-end modeling of I/Q mismatch effects in a complex-IF receiver for image-rejection enhancement“. In: *Electronics, Circuits and Systems, 2003. ICECS 2003. Proceedings of the 2003 10th IEEE International Conference on*. Vol. 2. IEEE. 2003, pp. 631–634.

[MZ+08] E. S. Morales, G. Zucchelli, et al. „Novel Methodology for Functional Modeling and Simulation of Wireless Embedded Systems“. In: *EURASIP Journal on Embedded Systems* 2008 (2008).

[Nag75] L. W. Nagel. „SPICE2: A Computer Program to Simulate Semiconductor Circuits“. PhD thesis. EECS Department, University of California, Berkeley, 1975. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/1975/9602.html.

[New78] A. R. Newton. „The Simulation of Large-Scale Integrated Circuits“. PhD thesis. EECS Department, University of California, Berkeley, 1978. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/1978/9605.html.

[OBC05] S. Orcioni, G. Biagetti, and M. Conti. „SystemC-WMS: a wave mixed signal simulator“. In: *Proceedings of the 8th International Forum on Specification & Design Languages (FDL)*. 2005, pp. 27–30.

[Ohm27] G. Ohm. *Die galvanische Kette*. 1827.

[Pan01] P. . Panda. „SystemC - a modeling platform supporting multiple design abstractions“. In: *System Synthesis, 2001. Proceedings. The 14th International Symposium on*. 2001.

[Pat05] S. . Patel H.D. ; Shukla. „Towards a heterogeneous simulation kernel for system-level models: a SystemC kernel for synchronous data flow models“. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 24.8 (Aug. 2005), pp. 1261–1271.

[Pel09] R. Pelánek. „Fighting state space explosion: Review and evaluation“. In: *Formal Methods for Industrial Critical Systems* (2009), pp. 37–52.

[Raz98] B. Razavi. *RF Microelectronics*. Ed. by T. S. Rappaport. Prentice Hall, Nov. 1998. ISBN: 9780138875718.

[RS09] J. Reichardt and B. Schwarz. *VHDL-Synthese: Entwurf digitaler Schaltungen und Systeme*. Oldenbourg Wissenschaftsverlag, 2009.

[RWT03] D. Root, J. Wood, and N. Tufillaro. „New techniques for non-linear behavioral modeling of microwave/RF ICs from simulation and nonlinear microwave measurements“. In: *Proceedings of the 40th annual Design Automation Conference*. ACM. 2003, pp. 85–90.

[Sch03] R. Schreier. „The delta-sigma toolbox“. In: (2003).

[Sch11] M. Schleyer. „Hierarchical Functional Verification Approaches for a Low Power Bluetooth RF-Frontend“. MA thesis. RWTH Aachen University, 2011.

[SFB05a] R. Staszewski, C. Fernando, and P. Balsara. „Event-driven simulation and modeling of phase noise of an RF oscillator“. In: *Circuits and Systems I: Regular Papers, IEEE Transactions on* 52.4 (2005), pp. 723–733.

[SFB05b] R. Staszewski, C. Fernando, and P. Balsara. „Event-driven Simulation and modeling of phase noise of an RF oscillator“. In: *Circuits and Systems I: Regular Papers, IEEE Transactions on* 52.4 (2005), pp. 723–733. ISSN: 1549-8328. DOI: {10.1109/TCSI.2005.844236}.

[Sma] *A Simulink/SMASH co-simulation interface*. Application Note. DOLPHIN INTEGRATION, Oct. 2003. URL: http://www.dolphin.fr/medal/smash/notes/simulinksmash.pdf.

[SN08] A. Soury and E. Ngoya. „Using sub-systems behavioral modeling to speed-up RFIC design optimizations and verifications“. In: *Integrated Nonlinear Microwave and Millimetre-Wave Circuits, 2008. INMMIC 2008. Workshop on*. IEEE. 2008, pp. 165–168.

[SN90] R. Saleh and A. Newton. *Mixed-mode simulation*. Kluwer Academic Publishers, 1990.

[ST+05] R. Schreier, G. Temes, et al. *Understanding delta-sigma data converters*. IEEE press NJ, 2005.

[Ste11] S. Steinhorst. „Formal Verification Methodologies for Nonlinear Analog Circuits“. PhD thesis. Goethe University Frankfurt am Main, 2011.

[SYR02] S. Soliman, F. Yuan, and K. Raahemifar. „An overview of design techniques for CMOS phase detectors“. In: *Circuits and Systems, ISCAS IEEE International Symposium on* 5 (May 2002), pp. 457–460.

[The02] D. Theil. „Entwicklung eines optimierten Transmitterkonzeptes für Bluetooth“. PhD thesis. Universität Dortmund, Sept. 2002.

[TS94] Y. Tsividis and K. Suyama. „MOSFET modeling for analog circuit CAD: Problems and prospects“. In: *Solid-State Circuits, IEEE Journal of* 29.3 (1994), pp. 210–216.

[TZN10] B. Thiel, N. Zimmermann, and R. Negra. „Digital asynchronous signal interpolation and clock domain crossing“. In: *Microwave Conference Proceedings (APMC), 2010 Asia-Pacific*. IEEE. 2010, pp. 1420–1423.

[Vla91] M. Vlach. „Mixed-Mode-Simulator Interface“. Pat. US4985860. Jan. 1991.

[Wan08] Y. Wang. „Frontendmodellierung von HF Funksystemen für die Full-Chip-Verifikation“. MA thesis. RWTH Aachen University, 2008.

[WCH10] Y. Wang, Z. Chen, and S. Heinen. „Hierarchical generation of pin accurate SystemC models based on RF circuit schematics“. In: *Behavioral Modeling and Simulation Conference (BMAS), 2010 IEEE International*. IEEE. 2010, pp. 25–30.

[Wer11] T. D. Werth. „A Feedback Interference Cancellation Technique for Mitigation of Blockers in Wireless Receivers“. PhD in Electrical Engineering. RWTH Aachen University, Germany, 2011, p. 166.

[WGR05] B. Wile, J. Goss, and W. Roesner. *Comprehensive Functional Verification: The Complete Industry Cycle (Systems on Silicon)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN: 0127518037.

[Whi95] J. Whitesitt. *Boolean Algebra and its applications*. Dover Pubns, 1995.

[WJ+09] Y. Wang, S. Joeres, et al. „Modeling Approaches for Functional Verification of RF-SoCs: Limits and Future Requirements“. In: *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on* 28.5 (May 2009), pp. 769 –773. ISSN: 0278-0070. DOI: 10.1109/TCAD.2009.2014533.

[WVM+09] Y. Wang, C. Van-Meersbergen, et al. „Event driven analog modeling for the verification of PLL frequency synthesizers“. In: *Behavioral Modeling and Simulation Workshop, 2009. BMAS 2009. IEEE*. Sept. 2009, pp. 25 –30.

[WWF87] D. Watt, B. Wichmann, and W. Findlay. *Ada language and methodology*. Prentice Hall International (UK) Ltd., 1987.

[YCK07] Z. Ye, W. Chen, and M. P. Kennedy. „Modeling and Simulation of DeltaSigma Fractional-N PLL Frequency Synthesizer in Verilog-AMS“. In: *IEICE Trans Fundamentals* E90-A.10 (Oct. 2007), pp. 2141–2147. DOI: 10.1093/ietfec/e90-a.10.2141. URL: http://ietfec.oxfordjournals.org/cgi/content/abstract/E90-A/10/2141.

[Ypm95] T. Ypma. „Historical development of the Newton-Raphson method“. In: *SIAM review* (1995), pp. 531–551.

[ZG+95] M. Zwolinski, C. Garagate, et al. „Anatomy of a simulation backplane“. In: *Computers and Digital Techniques, IEE Proceedings-*. Vol. 142. 6. IET. 1995, pp. 377–385.

[ZGH10] Y. Zaidi, C. Grimm, and J. Haase. „On Mixed Abstraction, Languages, and Simulation Approach to Refinement with SystemC AMS“. In: *EURASIP Journal on Embedded Systems* 2010 (2010).

[Zim11] N. Zimmermann. „Design and Implementation of a Broadband RF-DAC Transmitter for Wireless Communications“. PhD in Electrical Engineering. RWTH Aachen University, Germany, 2011, p. 151.

[ZWB03] A. Zhu, M. Wren, and T. Brazil. „An efficient Volterra-based behavioral model for wideband RF power amplifiers“. In: *Microwave Symposium Digest, 2003 IEEE MTT-S International*. Vol. 2. IEEE. 2003, pp. 787–790.

[Acc11] Accellera Organization. *Standard Universal Verification Methodology Class Reference Manual*. June 2011. URL: http://www.accellera.org/downloads/standards/uvm/UVM_1.1_Class_Reference_Final_06062011.pdf.

[Blu10] Bluetooth SIG. *Specification of the Bluetooth System - Version 4.0*. 4th ed. Bluetooth Special Interest Group. June 2010.

[Cad] Cadence Design Systems, Inc. *Virtuoso Accelerated Parallel Simulator*. Cadence Design Systems. URL: http://www.cadence.com/rl/Resources/datasheets/virtuoso_mmsim.pdf.

[Cad03] Cadence Design Systems, Inc. *Creating Analog Behavioral Models - Verilog-AMS Analod Modeling*. Platform Application Note. 2003.

[Cad08] Cadence Design Systems, Inc. *OCEAN Reference*. Product Version 5.1.41. Sept. 2008.

[Cad09] Cadence Design Systems, Inc. *ASSURA PHYSICAL VERIFICATION*. Datasheet. 2009. URL: http://www.cadence.com/rl/Resources/datasheets/4903_VirtuosoALVS_DSfnl.pdf.

[Cad10] Cadence Design Systems, Inc. *Assertion Checking in Simulation*. 10.2. Nov. 2010.

[Cad11a] Cadence Design Systems. *Virtuoso schematic editor user guide*. Product Version 6.1.5. 2011.

[Cad11b] Cadence Design Systems, Inc. *Cadence SKILL Language User Guide*. Cadence Design Systems. Nov. 2011.

[Cad11c] Cadence Design Systems, Inc. *Cadence® Verilog®-AMS Language Reference*. Jan. 2011.

[Cad11d] Cadence Design Systems, Inc. *Incisive Enterprise Specman Products*. Datasheet. 2011. URL: http://www.cadence.com/rl/Resources/datasheets/specman_elite_ds.pdf.

[Cad11e] Cadence Design Systems, Inc. *Virtuoso AMS Designer Simulator User Guide*. Product Version 10.25. Jan. 2011.

[Cad11f] Cadence Design Systems, Inc. *Virtuoso® Spectre® Circuit Simulator Reference*. Product Version 10.1.1. June 2011.

[Cad11g] Cadence Design Systems, Inc. *Virtuoso® Spectre® Circuit Simulator RF Analysis Theory*. Product Version 10.1.1. June 2011.

[Cad12] Cadence Design Systems, Inc. *Virtuoso Hierarchy Editor User Guide*. Product Version 6.1.5. Jan. 2012.

[IEE01] IEEE Design Automation Standards Committee. „IEEE Standard Verilog Hardware Description Language“. In: *IEEE Std 1364-2001* (2001), 0_1–856. DOI: 10.1109/IEEESTD.2001.93352.

[IEE07] IEEE Design Automation Standards Committee. „IEEE Standard VHDL Analog and Mixed-Signal Extensions“. In: *IEEE Std 1076.1-2007 (Revision of IEEE Std 1076.1-1999)* (2007), pp. c1 –328. DOI: 10.1109/IEEESTD.2007.4384309.

[IEE08] IEEE Design Automation Standards Committee. „Std 1076–2008, IEEE Standard VHDL Language Reference Manual“. In: *IEEE, New York, NY, USA* (2008).

[IEE09] IEEE Design Automation Standards Committee. „IEEE Standard for System Verilog-Unified Hardware Design, Specification, and Verification Language“. In: *IEEE STD 1800-2009* (2009), pp. C1 –1285. DOI: 10.1109/IEEESTD.2009.5354441.

[Men07] Mentor Graphics Corporation. *Calibre nmLVS*. 2007. URL: http://www.mentor.com/products/ic_nanometer_design/verification-signoff/circuit-verification/calibre-nmlvs/upload/nmlvs_datasheet.pdf.

[Ope] Open SystemC Initiative. „SystemC 2.2.0“. In: URL: http://www.systemc.org.

[Sys08a] SystemC AMS working group. „OSCI SystemC AMS Users Guide“. In: *Open SystemC Initiative (OSCI)* (Mar. 2008). URL: http://www.accellera.org/downloads/standards/systemc/ams.

[Sys08b] SystemC AMS working group. „Standard SystemC AMS Language Reference Manual“. In: *Open SystemC Initiative (OSCI)* (Mar. 2008). URL: http://www.accellera.org/downloads/standards/systemc/ams.

[Ver09] Verilog-AMS Technical Subcommittee. „Verilog-AMS Language Reference Manual, Release 2.3.1“. In: *Accellera Systems Inititative Standards* (June 2009). URL: http://www.accellera.org/downloads/standards/v-ams/VAMS-LRM-2-3-1.pdf.

# Curriculum Vitae

| | |
|---|---|
| Name | Yifan Wang |
| Date of Birth | 24. February 1981 |
| Place of Birth | Wuhan, V.R. China |

## Professional Experience

| | |
|---|---|
| since 10/2013 | Mixed-Signal verification engineer, Intel Mobile Communications GmbH, Munich, Germany |
| 05/2012 - 10/2013 | Verification engineer for RF-IC, Intel Mobile Communications GmbH, Duisburg, Germany |
| 04/2008 - 05/2012 | Research assistant and Ph.D. student at the Chair of Integrated Analog Circuits, RWTH Aachen University |
| 10/2007 - 02/2008 | Student worker at the Chair of Integrated Analog Circuits, RWTH Aachen university |
| 04/2006 - 10/2006 | Internship at Philips Medical Systems, Böblingen, Germany |
| 05/2004 - 06/2005 | Student worker at Institute of Electronic Materials II, RWTH Aachen university |

## Education

| | |
|---|---|
| 10/2001 - 02/2008 | Studies on Electrical Engineering and Information Technology at RWTH Aachen University, Aachen, Germany. Diploma thesis: “Frontendmodellierungen von HF Funksystemen für die Full-Chip-Verifikation” |
| 06/2001 | Abitur, Duisburg |