Application Specific Analysis Methods for Complex Electronic and Electric Systems

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

Christoph Beyerstedt,

M.Sc.

aus Aachen

Berichter: Univ.- Prof. Dr.-Ing. Stefan Heinen
Univ.- Prof. Dr.-Ing. Tobias Gemmeke

Tag der mündlichen Prüfung: 30.09.2022

Some believed we lacked the programming language to describe your perfect world.

Agent Smith
The Matrix

CONTENTS

ACKNOWLEDGEMENT

This thesis was created during my scientific activity at the Chair for Integrated Analog Circuits and RF Systems of the Rheinisch-Westfälische Technische Hochschule (RWTH) Aachen University and is the sum of many fruitful discussions, experiences and food for thought from friends and colleagues.

The work presented here would not have been possible without the contribution of various persons. Unfortunately, it is not possible to thank everyone to the extent that they deserve. However, I will express my particular gratitude in the next lines.

A special thanks goes to Prof. Dr.-Ing. Stefan Heinen for giving me the opportunity to work on this exciting topic. He always supported my with my work, shared his experience and assisted me with his expertise on RF systems and simulation techniques. Furthermore, I want to thank Dr.-Ing. Ralf Wunderlich for taking care of the 'tedious administrative stuff' and for providing me always with good advices.

The development of the event-driven simulation environment was a common project of Fabian Speicher, Jonas Meier and me. I want to express me deep gratitude towards them for the many helpful discussions, the good ideas and the support they gave me. A special thank also goes to Markus Scholl, Tobias Saalfeld, Vahid Bonehi and Michael Hanhart for sharing their deep knowledge on circuit and system design with me. I appreciate the help of the colleagues who helped me correct this thesis, as there are Michael Hanhart, Johannes Bastl, Christopher Nardi, Jonas Meier and Alexander Meyer. Moreover, I want to say thank you to all other colleagues and students. I was happy to meet you Moritz Schrey, Tobias Zekorn, Eva Schulte-Bocholt, Tobias Piwczyk, Tim Lauber, Kenny Vohl, Lantao Wang, Leon Weihs, Daniel Blase and Alexander Tulov!!!

I would also like to thank Mr Frielingsdorf and Mr Hesse, who made my project with Ford run smoothly.

My wife and my parents always supported me during the busy times of my studies. A special thanks goes to them, as without them I would not have had the necessary strength and motivation to complete this work.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AC	Alternating Current
ADAS	Advanced Driver Assistance Systems
ADC	Analog-to-Digital Converter
APS	Accelerated Parallel Simulator
BB	Baseband
BER	Bit Error Rate
BTLE	Bluetooth Low Energy
BW	Bandwidth
CDC	Clock Domain Crossing
CMOS	Complementary Metal–Oxide-Semiconductor
CORDIC	Coordinate Rotation Digital Computer
CP	Charge Pump
DAC	Digital-to-Analog Converter
DC	Direct Current
DPI	Direct Programming Interface
DUT	Device-Under-Test
ECAD	Electronic Computer Aided Design
EDA	Electronic Design Automation
EM	Electromagnetic
ENOB	Effective Number of Bits
ETHB	Envelope Transient Harmonic Balance
FD	Frequency Divider
FFT	Fast Fourier Transformation
FMEA	Failure Mode and Effects Analysis
FSR	Fullscale Range
GA	Genetic Algorithm
GMSK	Gaussian Minimum Shift Keying
GPS	Global Positioning System
GUI	Graphical User Interface
HB	Harmonic Balance
HDL	Hardware Description Language

HW	Hardware
IAS	Institut für integrierte Analogschaltungen
IC	Integrated Circuit
IF	Intermediate Frequency
IFFT	Inverse Fast Fourier Transformation
IP	Intellectual Property
IPC	Inter-Process communication
ISM	Industrial, Scientific and Medical
KSK	Kundenspezifischer Kabelbaum/customer specific wire harness
LNA	Low Noise Amplifier
LO	Local Oscillator
LSB	Least Significant Bit
LTI	Linear Time-Invariant
MMD	Multi-Modulus Divider
MNA	Modified Nodal Analysis
ODE	Ordinary Differential Equation
OFDM	Orthogonal Frequency-Division Multiplexing
OPAMP	Operational amplifier
PA	Power Amplifier
PFD	Phase-Frequency Detector
PLL	Phase-Locked Loop
PPF	Polyphase Filter
PRBS	Pseudorandom Binary Sequence
PSN	Power Supply Noise
PSO	Particle Swarm Optimization
PSRR	Power Supply Rejection Ratio
PSS	Periodic Steady State
PVT	Process, Voltage and Temperature
PWC	Piece-Wise Constant
PWL	Piece-Wise Linear
QPSS	Quasi-Periodic Steady State
QSS	Quantized-State System
RF	Radio Frequency
RNM	Real Number Modelling
RTL	Register Transfer Level
RX	Receiver
SC	SystemC
SCA	Sneak-Circuit Analysis

SDR	Software-Defined Radio
SNR	Signal-to-Noise Ratio
SoC	System-on-Chip
SPI	Serial Peripheral Interface
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random-Access Memory
STL	Standard Template Library
SV	SystemVerilog
SW	Software
TX	Transmitter
UDN	User-Defined Nettype
VBS	Visual Basic Script
VCO	Voltage-Controlled Oscillator
VHDL	Very High Speed Integrated Circuit Hardware Description Language
VP	Virtual Prototype
WHO	Wire Harness Optimization

CHAPTER 1

INTRODUCTION

Two major events happened in the last century which made it possible that every person has now more computing power in his pocket than the computer that guided the first men to the moon:

- The invention of the transistor in 1947 by John Bardeen, William Shockley und Walter Brattain in the Bell labs [1].
- The invention of the monolithic Integrated Circuit (IC) in 1959 by Robert Noyce at Fairchild Semiconductor [2].

Before these inventions, the circuits were built of vacuum tubes wired together manually using 'flying-wire' connections. This method was time-consuming and expensive and furthermore the circuits were large and heavy. One of the largest computers built out of vacuum tubes was the ENIAC machine which contained about 18,000 tubes, had a power consumption of 140 kW, weighted 27 t and occupied an area of 170 m^2 [3].

The invention of the transistor and the IC changed circuit design rapidly. The planar manufacturing process allowed multiple transistors to be built on a single die and connected during the process steps avoiding the time-consuming and error-prone manually wiring. In 1965 Gordon Moore predicted that the numbers of transistors on a single die will double every 18 to 24 months. This prediction still holds and is known as Moore's Law [4]. But not only the size and weight of the electronic components shrank but also the price and power consumption greatly reduced over the last decades.

The development of Radio Frequency (RF) Complementary Metal–Oxide-Semiconductor (CMOS) technology by Asad Ali Abidi [6] was the next big step in the digital revolution which enabled the production of integrated RF circuits for wireless telecommunication and therefore was the fundament for the rise of the communication era. Instead of purely analog transceivers, more and more functionality was shifted in the digital domain which offers more flexibility and noise suppression than analog signal processing. Moreover, the designs process benefits from the reuse of digital Intellectual Property (IP). The ultimate goal is the ideal Software-Defined

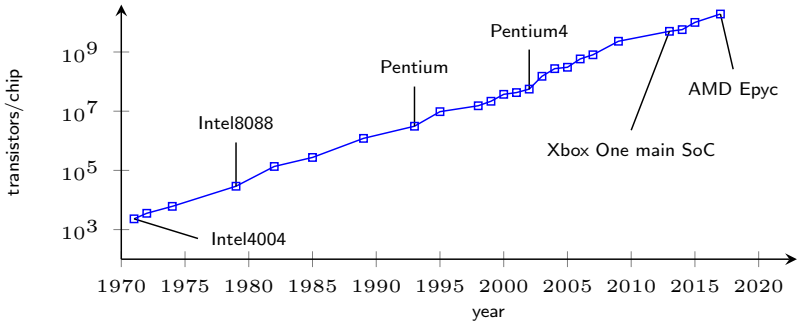


Figure 1.1: Transistor count per chip over the years (data source: [5]).

Radio (SDR), where every analog hardware component is replaced by a digital system or software which would give a maximum of flexibility.

The physical world is analog and therefore there will always be a need for analog circuits which connects the physical world to the abstraction of the digital world. Due to their continuous nature, analog systems are highly complex to design and moreover, analog signals tend to be very sensitive to noise, parasitic circuit effects and process variations. Therefore, analog design requires a high fraction of the overall design time of mixed signal integrated circuits.

Nowadays, the boundary between pure Hardware (HW) and pure Software (SW) development is disappearing more and more and HW/SW co-design is shifted in the foreground. Designing modern systems is not purely HW or SW development but both HW and SW tightly interacts with each other. Both parts are tailored for each other. HW is controlled by SW and the SW is optimized for a specific piece of HW.

But not only the semiconductor industry has to deal with the challenges of rising complexity. Due to the increased use of electrical components in vehicles such as driver assistance systems, the size and complexity of the wire harness used to connect the different subsystems is fast-growing. Depending on the type of car, the harness cable length can be up to 3 km and the weight up to 80 kg [7]. Furthermore, not only electrical properties have to be considered, but also the interaction between electrical and other physical quantities, such as the torque of a motor, the environment temperature, etc. has to be taken into account. Moreover, the trend to customize the configuration of cars for the individual customer leads to a huge variance of harnesses [7].

The massive increase of complexity from a few hundred or thousand vacuum tubes to millions or billions of transistors and the strong interdependency between HW, SW and multiple other non-electrical subsystems demand for performant and sophisticated analysis methods in the design as well as in the verification process. Computer-aided methods and automation of manual steps are necessary in today's design processes to deal with the complexity and to stay competitive.

1.1 Importance of Modelling and Simulation

In the context of this work, the term modelling refers to the creation of mathematical models of a circuit or system of interest. A description of a mathematical model is given by the Dutch mathematician Sjoerd W. Rienstra:

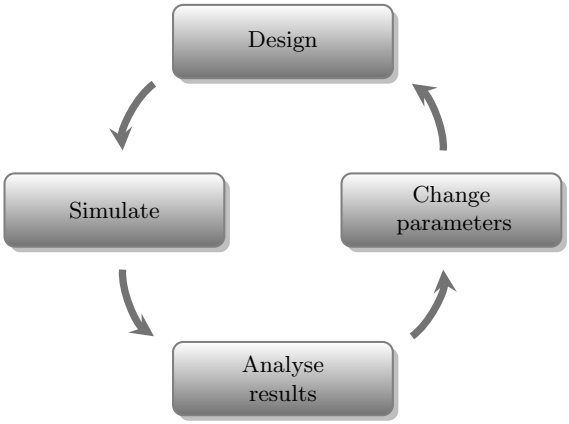
> *Describing a real-world problem in a mathematical way by what is called a MODEL, such that it becomes possible to deploy mathematical tools for its solution. The accuracy of the description should be limited, in order to make the model not unnecessary complex.* [8]

One purpose of a model is to predict the real-world system's behaviour the model represents. Thus, it must be accurate enough so that the prediction is useful but on the other hand it must not be too complex so that the analyst cannot calculate the behaviour in a reasonable time. A well-designed model is a good trade-off between realistic behaviour and simplicity.

The evaluation of a model is often exercised on a computer using a simulation software. Simulations are done to study the behaviour of the system. The reason these studies are done on the model of the system are manifold. It could be too expensive, too time-consuming, impractical or too dangerous to test the real-world system. Furthermore, in some cases simulations of models offer insight into system variables that cannot be accessed in reality.

In engineering context simulating is used for two main reasons. Simulations can be part of an design cycle in which the system is optimized iteratively or they can be used to verify the final design and find bugs before start of production. A typical design process is depicted in Fig. 1.2. The engineer designs a system in a virtual environment, simulates it, analyses the results and changes some parameters to optimize the system. This procedure is repeated until the system's specifications are fulfilled.

When designing complex systems, there will be always bugs in the design. The later the errors are found the more expensive it will be to fix them [9]. In the early



;

Figure 1.2: Iterative design process.

concept stage, it is relatively harmless to change a few things. Later, in the design or verification phase, fixing a fault can be more laboriously and time-consuming because it might result in a redesign of the system concept. After release in the production stage, it will become very expensive to fix bugs and in the worst case it results in a recall and a complete redesign of the product. A famous example of a costly design error was the *Pentium FDIV* bug in the floating-point divider unit of Intel Pentium PCs produced in 1994 [10], which resulted in a large replacement program and costed about 475 million dollars.

Well-designed models and good analysis methods help identifying errors and design bugs earlier in the design process and massively reduces costs by this. Virtual Prototypes (VPs) or *digital twins* allow to explore new concepts and examine architectures before the first physical prototype is produced. Thus, system faults can be detected earlier and their impact is mitigated.

Besides finding errors before production, a VP allows concurrent HW and SW development which saves valuable time and helps reducing the time to market. In Fig. 1.4a and Fig. 1.4b a classical design flow is compared to a flow in which HW and SW is designed concurrently on base of virtual prototypes. It can be seen, that in the concurrent design flow a lot of time can be saved which is indispensable in today's tight time-to-market frame.

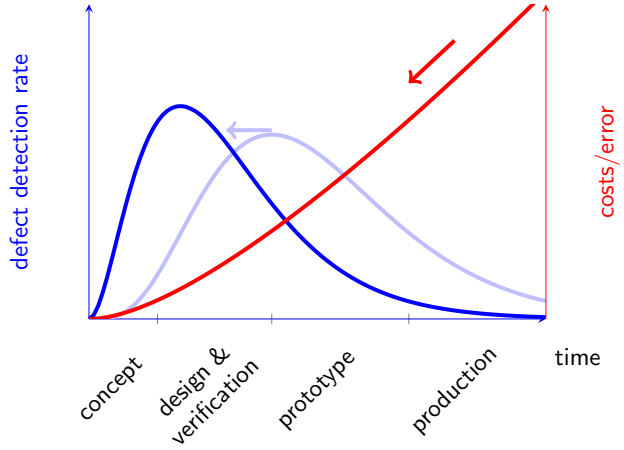
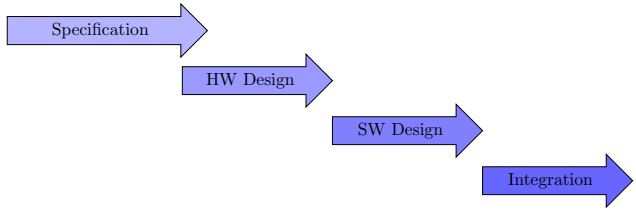
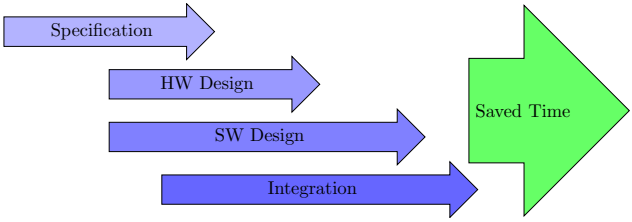


Figure 1.3: Cost reduction by virtual prototyping.



(a) Classical design flow.



(b) Concurrent design flow.

Figure 1.4: Classical vs. concurrent design flow (adapted from [11]).

1.2 Goal of this Work

As ICs began to incorporate hundreds of transistors, the design verification was too laborious and error prone for a purely manual task. In the 1960s Simulation Program

with Integrated Circuit Emphasis (SPICE) was developed in Berkeley. This circuit level simulator was good enough for a few years but with the increasing complexity of the digital systems, the simulation speed was a bottleneck. Hardware Description Language (HDL) in combination with event-driven simulations solved this problem. On a more abstract level, they are able to describe and simulate the systems without the limitations of a SPICE-like simulator.

With decreasing feature sizes, the gap between the number of designed gates and the number of verified gates began to widen because the effort of testing rose faster than the effort of designing the systems. This is known as the verification gap [12] and nowadays verifying a complex design can take as much as 80% of the total product development time [13].

A massive effort is put in the verification of digital circuitry. A variety of tools from different vendors which supports several analysis approaches like event-driven simulations or formal verification are on the market. With SystemVerilog (SV) and SystemC (SC), mighty HDLs for the description of digital systems on different abstraction levels are available. But comparable languages and simulation methods for analog and mixed signal systems are missing. Although VerilogA was developed to describe analog circuits, it still relies on a matrix-based solver for simulation and thus is not performant enough for large systems.

Therefore, simulation methods and hardware description languages which suits both the analog and the digital parts are required. It has been shown that an event-driven simulation of mixed-signal systems with so-called Real Number Modelling (RNM) of the analog blocks is a good trade-off between the simulation speed and accuracy [14] but in the field of RF RNM there are still many white spaces on the map.

A large part of this work deals with the extension of modelling and simulation methods needed for the verification of large mixed signal RF systems. The focus is set especially on simulation speed and accuracy. Different signal descriptions are used in order to reach the best simulation performance. Furthermore, different modelling approaches are investigated regarding the complexity and expressiveness of the resulting models. Since there is always a time-to-market pressure in IC design, ways for a fast model development are important. Therefore, automating the model creation process wherever it is sensible helps saving valuable time. In addition to the modelling methodologies, in this thesis it is investigated on how some system properties can be exploited to achieve a good simulation performance and expressiveness of the results.

Not only in the IC industry new design and simulation approaches are needed but also in other areas. The wire harness has a big influence on electric functions, needed space, weight, and costs. For getting a fully functional and optimized wiring,

a computer aided analysis of the harness design early in the development process is necessary [15]. Methods for analysing and optimizing wire harnesses is the main concern of the second part of this thesis. Since some properties of these systems are hard to grasp with conventional methods, new concepts and tools are necessary to deal with the increasing complexity in the automotive area.

1.3 Thesis Structure

In addition to the introduction, this thesis includes 9 other chapters.

- In chapter 2 fundamentals in simulation techniques, RF systems and wire harnesses are explained for better understanding of the following chapters.
- Chapter 3 deals with efficient ways to describe signals for event-driven simulations and chapter 4 outlines RNM approaches of commonly used system properties and model development methods.
- Ways to fit the model parameters are discussed in chapter 5.
- A small signal noise analysis based on the already developed RF RNM methodology is presented in chapter 6.
- Chapter 7 presents simulation results using the developed techniques applied on a real-world example on a mixed-signal RF IC developed at the Institut für integrierte Analogschaltungen (IAS).
- In chapter 8 a Sneak-Circuit Analysis (SCA) algorithm for the verification of a proper functionality of wire harnesses is proposed.
- Chapter 9 deals with the automated optimization of wire harnesses based on a Genetic Algorithm (GA) and simulations of the system.
- A conclusion of this work and a further outlook is given in chapter 10.

CHAPTER 2

FUNDAMENTALS

When discussing the analysis of electric and electronic systems, it is surely important to understand the difficulties that arises in the analysis process. Therefore, a short introduction to the most important system properties and analysis methods is provided in the following.

2.1 RF Systems

An example of a complex system is a modern RF System-on-Chip (SoC). An RF integrated circuit contains nowadays several functional subsystems on a single piece of silicon, and it is optimized among other things for current consumption, sensitivity and linearity. It needs to be as flexible as possible to support the different wireless protocols and modulation schemes. Thus, the analog part of the SoC can often be configured by digital tuning signals. With shrinking technology nodes, more and more functionality is shifted from the analog part to the digital part [16] because of flexibility reason, area and power consumption. Furthermore, digital circuits can be more easily ported to other technology nodes since they are usually described on an abstract level by an HDL and the mapping on a specific technology node is done automatically by a compiler. Moreover, to reduce integration costs, RF and digital Baseband (BB) blocks are integrated together on a single chip [16].

The shift towards digital-centric circuits leads to a strong interaction between digital control and steering systems and the remaining analog circuits. These kinds of system are called mixed-signal systems. The downscaling and the increasing digital circuitry do not only provide advantages. More digital activity leads to an increased supply and substrate noise [16]. The smaller transistor sizes result in lower supply voltages and less headroom [17] and make it difficult to design for high gain and process stable analog circuits [18] which are needed to transmit and receive signals with complex modulation schemes such as Orthogonal Frequency-Division Multiplexing (OFDM). Furthermore, the wireless spectrum gets more and more crowded [19] which leads to tight linearity specs for the analog frontend [20] to avoid problems coming from the intermodulation of two signals in a nonlinear component. Summing

up, analysis methods for RF systems should cover the nonideal effects as noise and nonlinearity while on the other hand they must be performant enough to deal with large mixed-signal systems. As explained later in this chapter, the simulation of mixed-signal systems is very challenging and with increasing system complexity, new methodologies for analysing such systems are needed.

2.1.1 Noise

Noise is defined as any random or quasi-random unwanted signals which disturb a wanted signal in a system. It origins from a variety of sources and due to its random or quasi-random nature, it is best described, through its statistical properties [21]. There are several types of noise, which are separated by their physical origin or their impacts on the system. The most common ones in RF IC design are listed below, but there are several more types.

- *Thermal noise* origins in randomly moving electrons. Its instantaneous amplitude value has a Gaussian distribution and the power spectral density is flat over frequency with the value $p_n = kT$ where T is the ambient temperature and k is the Boltzmann constant [22].

- The spectral density of *flicker noise* decreases proportional to $\frac{1}{f}$. Because of this phenomenon it limits the performance of circuits operating near Direct Current (DC). *Flicker noise* is related to the material surface properties [22].

- *Phase noise* is the random fluctuation of a signal's phase from its ideal value. It can be seen, e.g., as jitter in the zero-crossings of a clock signal. However, the source of *phase noise* comes from randomly moving electrons and is the same as *thermal noise* or *flicker noise* [22].

- *Quantization noise* arises from the quantization of a signal. The quantized signal does not perfectly match the original signal and the difference between both can be regarded as noise [22].

- *Supply noise* and *substrate* noise are distortions coming from the activity of fast switching circuits which propagates via the supply lines or the common substrate to sensitive blocks. With increasing digital circuitry on RF SoCs these noise types get more and more problematic [23] [24].

2.1.2 Linearity

A system is linear if the transformation T of the linear combination $\alpha_1 x_1 + \alpha_2 x_2$ of two input signals x_1 and x_2 can be decomposed as shown in eq. 2.1.

$$T[\alpha_1 x_1 + \alpha_2 x_2] = \alpha_1 T[x_1] + \alpha_2 T[x_2] \tag{2.1}$$

In linear systems, only phase and amplitude are influenced but the frequency of a signal remains unchanged. For most circuits, it is desirable to design it as linear as possible to avoid distorting the signals but some kind of systems rely on their nonlinearity, e.g., some kind of mixers, because frequency conversion can only occur in nonlinear or time-varying systems. Nonlinear circuits are often classified in two categories, in weakly nonlinear and strongly nonlinear systems [25]. A strongly nonlinear circuit is for example a limiting amplifier where the output signal is clipped. A weak nonlinearity can be found in amplifiers with a small to medium excitation where the transfer function differs only little from an ideal line. The mathematical description of nonlinearities depends on the type of nonlinearity. For weakly linear systems often a polynomial description is chosen. While in analog systems noise is the fundamental lower limit for the signals amplitude, the upper signal level is determined by the nonlinearity. In RF receivers, several nonlinear mechanisms like *compression*, *desensitization* and *intermodulation* degrade their performance [26]. In eq. 2.2 the impact of the nonlinear transfer function $T(x) = ax + cx^3$ on the superposition of two sinusoidal signals is shown.

$$\begin{aligned}
&T[A_1 cos(\omega_1 t) + A_2 cos(\omega_2 t)] = \\
&A_1 \cdot \left(a + \underbrace{\frac{3cA_1^2}{4}}_{compression} + \underbrace{\frac{3cA_2^2}{2}}_{desensitization} \right) \cdot cos(\omega_1 t) \\
&+ A_2 \cdot \left(a + \underbrace{\frac{3cA_2^2}{4}}_{compression} + \underbrace{\frac{3cA_1^2}{2}}_{desensitization} \right) \cdot cos(\omega_2 t) \\
&\underbrace{+ \frac{3c}{4} A_1^2 A_2 cos((2\omega_1 - \omega_2)t) + \frac{3c}{4} A_1 A_2^2 cos((2\omega_2 - \omega_1)t)}_{intermodulation} \\
&+ \frac{3c}{4} A_1^2 A_2 cos((2\omega_1 + \omega_2)t)) + \frac{3c}{4} A_1 A_2^2 cos((2\omega_2 + \omega_1)t)) \\
&+ \frac{c}{4} A_1^3 cos(3\omega_1 t) + \frac{c}{4} A_2^3 cos(3\omega_2 t)
\end{aligned} \tag{2.2}$$

Since the coefficients a and c have different signs in real circuits, the gain of the circuits gets less with increasing amplitude of the signals. The influence of the signal on itself is called *compression* while the influence of another signal on the first one is called *desensitization*. If the amplitude of the second signal is large enough, it blocks the reception of the wanted signal and therefore it is called *blocker*. Furthermore, it can be seen that new spectral components arise at linear combinations of ω_1 and ω_2. It might happen that two strong signals at ω_1 and ω_2 in the same frequency band create a new in-band component at $2\omega_1 - \omega_2$ or $2\omega_2 - \omega_1$ which falls on the same frequency as a weak receive signal. These *intermodulation products* can mask the wanted signal and degrade the receiver performance or even make the reception impossible.

2.2 Automotive Wire Harnesses

Another example of complex electric systems is the wire harness of modern automobiles. Due to the increasing electrification of cars, the dimensioning of the wire harness is becoming more and more important. The wiring has a big influence on the proper electrical functionality, the power loss, the costs, the weight and the required space in the cars. Only with the help of suitable analysis and design methodologies an optimal wire harness can be developed with minimum cost in a reasonable time which fulfils the required electrical conditions [7].

A challenge for the voltage stability is for example the increasing value of required peak currents over the last years. It is crucial that safety relevant systems are supplied with a stable voltage in a well-defined range. Furthermore, the demand for Kundenspezifischer Kabelbaum/customer specific wire harness (KSK) increase the complexity of the wire harness design [7]. In addition to electrical properties, also the thermal characteristics of the wire harness and the interaction between thermal and electrical behaviour must be considered. Usually, the thermal response of the wires has a much higher time constant than the response of the electrical system [7], resulting in long simulation times. Examples of such mixed-domain simulations are wire temperature and fuse blow test under various fault situations like short-circuits. The execution of such tests on a real prototype are costly both in time and money and are often so late in the design process that it is very expensive to change the design [9]. Thus, analysis methods relying on a VP are preferred because they can be done earlier in the design process and are less expensive and faster once the models exist. Furthermore, there could be test cases which are too expensive or too dangerous to conduct on a real prototype so that the analysis on base of a VP is the only possibility. Since automotive systems can potentially cause harm in the

road traffic if critical systems have a malfunction, car components have to fulfil high safety requirements before admission. To get an overall high functional safety standard, the ISO 26262 has been created and is valid for electrical/electronic/programmable electronic (E/E/PE) systems in the automotive sector since 2011 [27]. Products developed under this standard have high safety requirements and needs to be analysed and tested thoroughly. Failure Mode and Effects Analysis (FMEA) and Sneak-Circuit Analysis (SCA) need to be done to ensure safety if parts of the system fail [28]. Thus, computer-aided methods are needed which provides on the one hand precise and meaningful predictions on the safety and operability of the systems under normal and failure conditions and which are on the other hand performant enough to realize the analysis in an acceptable time.

2.3 Analysis Methods

Since in most cases tests on a real prototype are either too expensive and time-consuming or impractical, the correct functionality of the systems needs to be verified beforehand using manual or computer-based verification and analysis methods. These methods can be separated into two categories [29]:

1. Formal analysis/verification
2. Simulation based analysis methods

In the following, both analysis types are presented and the advantages and disadvantages are explained.

2.4 Formal Verification

Formal verification tries to prove mathematically the correctness of the design. This can be done manually or with the help of special Electronic Design Automation (EDA) tools (JasperGold®, Spyglass®, ...). Traditionally, hardware bugs were discovered by simulation of many possible test cases. However, for a large SoC the number of possible test scenarios is too large to simulate all of them in a reasonable time. Imagine, a digital circuitry with 400 input pins. If every possible input combination should be checked, 2^{400} input vectors need to be simulated. Even if every particle of the universe ($\sim 10^{89}$ atoms) would be a super computer that could simulate one input combination in 1 ns, still 8.18×10^{14} years are needed to calculate every input combination. Even if the simulation had started with the big bang (~14 billion years ago), only 0.0017% had been simulated until now.

Formal verification methods tackle the problem from another side by proving the correctness or incorrectness of a given design without using stimuli. Two important mechanisms are widely used in the formal verification of ICs, equivalence checking and model checking [30]. Equivalence checking tries to prove the equivalence between two implementations, e.g., the Register Transfer Level (RTL) implementation and the synthesized circuit. Model checking or property checking checks if a model meets the given specification. It requires a precise model of the system and well-formulated properties to be tested. The idea behind the model checking methodology is to regard the system states in a state-space formulation. In its core it can be regarded as reachability analysis which checks if all required states can be reached. The problem is the exponential rise in states for an increasing size of the system [31]. Since analog systems have an infinite state space [32], these methods are mostly unsuitable for circuits that are not purely digital. Yet, there are some approaches for formal analysis of analog circuits [33] [34] which uses the discretization of the continuous analog state space but, this will not be discussed further in this thesis.

2.5 Simulation Based Analysis

As mentioned above formal verification methods are the only way to prove the correctness of a system but they suffer from the problem of the exploding state space for large systems and the lack to verify analog systems. Simulation based approaches use stimuli in a virtual environment to check different scenarios against the specification. Only a handful of these scenarios can be tested due to the tight schedule to bring the design to market and thus they have to be selected carefully so that meaningful conclusions on the system's functionality and performance can be drawn. Depending on the abstraction level there are different simulation methods. With an Electromagnetic (EM) solver, Maxwell equations can be solved but this is only feasible for circuits with only a few components. Large designs cannot be simulated in an EM solver because it would require too much computational power [35]. Here, it will be focussed on so-called circuit simulation which includes matrix-based simulation methods and event-driven simulations.

In Fig. 2.1 a typical simulation environment for a Device-Under-Test (DUT) is shown. A set of stimuli is derived from the specifications for testing different scenarios the DUT needs to pass. In some verification test cases, a *golden model* is built on basis of the specs which serves for checking the results of the simulation against 'perfect' simulation results. A *golden model* can be a reference design from a previous tapeout or a high-level description of the system to be designed. The results from the DUT simulation are compared against the simulation results from the *golden model* in

the monitor [36]. If there is no *golden model*, the probed results need to be analysed whether they match the specifications.

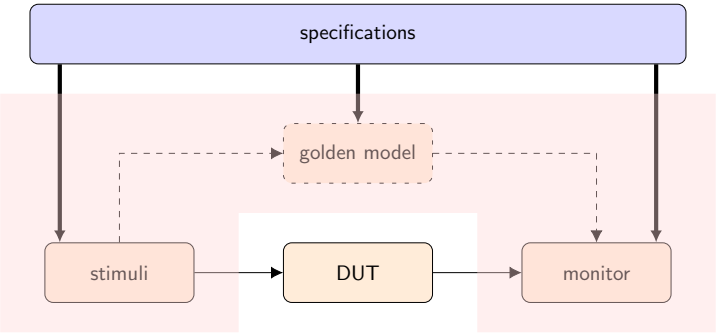


Figure 2.1: Typical testbench.

2.6 Matrix Based Analog Circuit Simulation

Conservative systems are described using conservation laws [37], in which a potential can be defined (an electrical potential, gravitational field). A conservative system can be described using its node potentials (also known as *cross* values) and the flow in the nodes (also known as *thru* values). In an electrical system, the potential and flow correspond to the node voltages and currents [37]. The relationship between voltage and current is described by the characteristic component equation, e.g., Ohm's law for a resistance. In contrast to EM solvers, in matrix based analog circuit simulation it is assumed that the spatial distribution of the elements can be neglected, so that lumped-element models of the components can be used. This assumption allows to ignore all spatial parameters in the equations.

An (integrated) circuit consists of the connections of electronic components which can be linear or nonlinear and static or frequency dependent. Using Kirchhoff's current law which states that the sum of currents in each node is zero, a nonlinear system of differential equations can be formulated on basis of a structural description of the circuit which is called netlist [38]. One way to formalize the creation of the equation system is the nodal analysis. The node voltages are the unknown for which the system of equations is solved and the currents are described in dependence of the voltages. A few exceptions must be made for ideal voltage sources, voltage-controlled voltage source, etc. which are incompatible with this scheme and therefore the

Modified Nodal Analysis (MNA) formalism was created. This systematic approach allows to build the equation systems automatically by a computer. The advantage of the method is, that it is generally applicable on every conservative electric system.

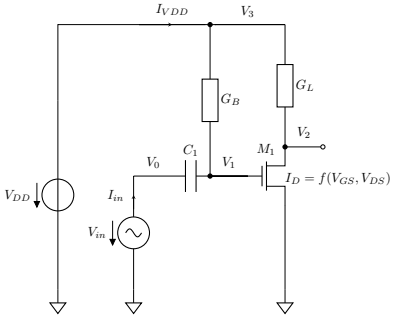
Fig. 2.2 shows an example circuit and its associated equation system. The system can be described by the matrix $\boldsymbol{G}$ which consists of the linear conductances in the circuit and a few other entries coming from the modification due to the ideal voltage sources. Furthermore, the matrix $\boldsymbol{C}$ comprises the linear, energy-storing elements like inductances and capacitances. The vector $\boldsymbol{I_{NL}}$ which contains the nonlinear, controlled current sources is found as additional term on the right side of the equation. If there would be nonlinear capacitances, the term $\frac{d}{dt}\boldsymbol{Q}$ would be attached where $\boldsymbol{Q}$ describes the nonlinear dependence of the charge. The excitation vector $\boldsymbol{I}$ contains all independent sources and the vector $\boldsymbol{V}$ is the solution vector. As can be already seen from this small example, the matrices $\boldsymbol{C}$ and $\boldsymbol{G}$ are not fully populated and for larger circuits, the matrices are often sparse [38] which can be exploited when solving the equations. Since the formulation of the equations requires the definition of a potential it fails when there is a non-negligible changing magnetic flux through a mesh of the circuit. The electric field would be non-conservative because of the magnetic induction and a potential and thus voltages are undefined. For these kinds of problems, an EM simulator is required [38].

In real circuits voltages and currents changes continuously over time and the values they can take are also continuous. Due to the fact, that a computer can only deal with discrete values, the formulation of the Ordinary Differential Equation (ODE) system must be transformed into a form which can be solved by a computer. Based on the mathematical description of the system, several analysis methodologies can be run.

2.6.1 Transient simulation

The most natural way to calculate the solution of the equation system is the transient simulation. In a transient analysis, the equation system is solved in time-domain by converting it to a set of nonlinear difference equations using integration methods [39] such as the backward Euler [40]. In the next step, the nonlinear difference equations are solved numerically using iterative methods like the Newton-Raphson algorithm [41]. The progress in time (time steps) are usually chosen dynamically by the simulator and are dependent on the highest frequency in the system [42]. The advantage of a transient analysis is that it can be applied on nearly every circuit without any restrictions on the periodicity of the signals or the linearity of the system.

$$\boldsymbol{C} \cdot \frac{d}{dt}\boldsymbol{V} + \boldsymbol{G} \cdot \boldsymbol{V} + \boldsymbol{I_{NL}}(\boldsymbol{V}) = \boldsymbol{I}$$



$$\boldsymbol{C} = \begin{pmatrix} C_1 & -C_1 & 0 & 0 & 0 & 0 \\ -C_1 & C_1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \boldsymbol{V} = \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ I_{in} \\ V_3 \\ I_{VDD} \end{pmatrix},$$

$$\boldsymbol{G} = \begin{pmatrix} 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & G_B & 0 & 0 & -G_B & 0 \\ 0 & 0 & G_L & 0 & -G_L & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -G_B & -G_L & 0 & G_B + G_L & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix},$$

$$\boldsymbol{I_{NL}} = \begin{pmatrix} 0 \\ 0 \\ I_D(V_1, V_2) \\ 0 \\ 0 \\ 0 \end{pmatrix}, \boldsymbol{I} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ V_{in} \\ 0 \\ V_{DD} \end{pmatrix} \tag{2.3}$$

Figure 2.2: Example circuit and the corresponding equation system. The transistor is modelled as voltage-controlled current source and all parasitic elements were neglected.

2.6.2 DC simulation

DC simulation computes the static operating point of a DUT. For a DC analysis the steady state of the circuit is assumed and thus all capacitances are regarded as opens and all inductors as shorts [38]. The resulting nonlinear algebraic equation system can be solved with the Newton-Raphson algorithm. DC simulation can be used for a first estimation of the power consumption and the calculated operating point is used as a starting point for other simulations like the transient analysis or the Alternating Current (AC) analysis, but no system dynamics can be simulated using a DC analysis.

2.6.3 AC simulation

An AC or small signal analysis calculates the frequency response of the linearized system as a function of the frequency of the independent sources. It is assumed that the input signal's magnitudes are so small that the nonlinearities in the DUT can be neglected. It can be run on top of a DC simulation that first needs to determine the operating point around which the circuit is linearized. The $\frac{d}{dt}$ operator is replaced by $j\omega$ and the resulting linear equation system can be solved. Using an AC analysis, the circuit's transfer functions can be simulated, e.g., the frequency dependent gain or the Power Supply Rejection Ratio (PSRR) of an amplifier. The drawback of the AC analysis is that it cannot handle circuits with a time varying operation point like mixers or switched-capacitor circuits and is unable to calculate the impacts of nonlinearities on the signals.

2.6.4 RF simulation techniques

As mentioned above an AC simulation cannot calculate the output of circuits with time-varying operating points and the time step of a transient analysis depends on the highest frequency in the system. When dealing with RF circuits, the highest frequency is often in the gigahertz range while the time the circuit needs to get in its final operation region can take up to a few milliseconds due to the high time constants of the biasing networks or high Q resonances [42]. However, if the steady-state solution is needed, a transient simulator needs to simulate as long as the transients from the start-up are vanished which can take an unacceptable long CPU time [42].

RF simulators exploit certain characteristics of RF circuits to directly find the periodic steady state [43]. There are two main RF simulation methods: *Periodic Steady State (PSS)* and *Harmonic Balance (HB)* simulations and various similar methods like the *Circuit Envelope Method* or *Sampling Methods* [43].

In the HB approach it is assumed that all node voltages and currents can be written as sum of sinusoids with the fundamental frequencies $k \cdot \omega, k \in \mathbb{N}_0$. The periodic steady-state can then be directly calculated in the frequency-domain by solving eq. 2.4 with the nonlinear components I and Q, the linear part Y and the excitation U. The variable k denotes which harmonic frequency is looked on.

$$\begin{aligned} &\sum_k F(V,k) = 0 \\ \text{with:}\quad &F(V,k) = I(V,k) + jk\omega Q(V) + Y(k) \cdot V(k) + U(k) \end{aligned} \tag{2.4}$$

To make the equation system calculable on a computer, the number of harmonics k must be truncated. Then, eq. 2.4 can be formulated in a matrix-vector notation. The nonlinear equation system can be iteratively solved by using the Newton-Raphson algorithm. Since not every nonlinearity can be directly calculated in frequency-domain, the nonlinear components I and Q are first transformed in time-domain by the Inverse Fast Fourier Transformation (IFFT), evaluated there and transformed back in the frequency-domain via Fast Fourier Transformation (FFT) [43]. The presented methodology can be extended to more than one fundamental frequency [42] and it can also be applied to autonomous systems like oscillators [43]. Furthermore, after calculating the periodic steady-state, small signal analysis methods like an AC (*hbac*) or a noise analysis (*hbnoise*) can be run on top of the HB simulation to compute the small signal behaviour of the circuit including frequency conversion effects.

Contrary to the HB approach in PSS simulations the steady-state currents and voltages are calculated in time-domain. Again, it is assumed that all currents and voltages are T-periodic and thus in the steady-state. Now, the two-point boundary value problem 2.5 applies.

$$v(t_0) = v(t_0 + T) \tag{2.5}$$

An initial state $v(t_0)$ can be guessed and by transient analysis the state at $t_0 + T$ can be calculated. At the same time, the sensitivity of the final state $v(t_0 + T)$ from the initial state is computed. If the final and initial state are equal, the PSS analysis is stopped and the steady-state is found. Otherwise, with the help of the sensitivity, a new initial state is calculated and the the next final state is calculated. This process is repeated until the steady-state is found [43]. As for the HB analysis, there exist extensions for the PSS methodology to simulate circuits with more than one fundamental frequency (*Quasi-Periodic Steady State (QPSS)*), autonomous systems and the periodic steady-state found by PSS can be used to apply small-signal analysis simulations (*pac* and *pnoise*). Which RF simulation technique to use depends on the application and it cannot be said that either HB or PSS technique is better than the other. If the circuit contains distributed components [43] or only a small number of harmonics are relevant [42], usually HB is preferred whereas if the system is excited with strongly discontinuous signals the PSS can be a better choice [43].

2.7 Event-Driven Simulation

The above-mentioned matrix-based simulation techniques are well-suited for small to medium sized circuits, but they suffer from the fact, that for the calculation of the voltages and currents equation systems must be solved. The computational effort

to solve a fully populated equation systems with n equations is $O(n^3)$ [44] in the Landauer notation. Usually, the admittance matrices describing electronic circuits are sparse and therefore with sophisticated solving algorithms, the time complexity can be reduced up to $O(n^{1.1}) - O(n^{1.5})$ for fast or parallel SPICE [45], but still it does not scale well for large systems. Furthermore, the simulator time steps are defined by the fastest changing node. All node voltages, also the slowly changing ones, are recalculated at every time step which makes matrix-based simulators inefficient for systems with hugely differing time constants, e.g., a Phase-Locked Loop (PLL).

As digital circuits became so large that SPICE-like simulators could not handle them any longer, a new simulation methodology was developed, event-driven simulations. In event-driven simulations the concept of signal flow is applied. Unlike in conservative systems, in signal flow systems only the potential and not the flow gets assigned to the node [37]. Furthermore, only unidirectional signal flow is allowed and so there is no effect from a change at the output of a component on the input. For digital gates, these constraints are acceptable since the feedback from a logic gate to its input is usually negligible.

A second abstraction in digital circuit simulation is the reduction in value resolution. Since digital systems use Boolean algebra, it is only important that it can be differentiated between the values 'logic 0' and 'logic 1'. To cover situations with tri-state logic and multiple output drivers, the values 'high impedance (z)' and 'unknown/invalid (x)' are needed. Using these four values, digital signals for event-driven simulations can be described. This abstraction allows the fast evaluation of digital circuitry because now Boolean algebra can directly be applied in the simulation.

Systems which are simulated with an event-driven simulator are described in an HDL such as Very High Speed Integrated Circuit Hardware Description Language (VHDL), SV or SC. Components modelled in an HDL possesses *sensitivity lists* that contains the conditions on which the output of a process can change its value. E.g., a register only changes its output on the positive/negative edge of the clock signal and the positive/negative edge of the reset signal but not on a change of the data signal.

An example of event-driven simulation of an inverter connected to an XOR-gate is shown in Fig. 2.3. The inputs q and r change values at t_0 and t_1, respectively. The calculation of the inverter's output signal is triggered by the input signal and the recalculation of the XOR-gate's output is done on the edges of signal q and signal w.

A problem arises now that the two signals arrive at the same time at the gate. Since real hardware runs concurrently but is simulated on a computer which can only handle sequential program code, the output depends on which process the simulator

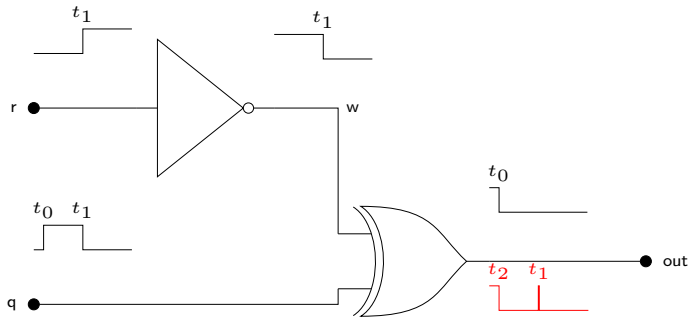


Figure 2.3: Exemplary digital circuit.

calculates first. If the XOR-gate is evaluated first, the new output of the inverter has not been calculated yet and so w is still '1' but if q already changed to zero, *out* will be 1 and only after the inverter output is calculated *out* will change back to '0'. But if the inverter is evaluated first, at t_1 the inputs of the XOR-gate are both zero and the output does not change. To avoid unpredictable, unrepeatable results the Δ-cycle concept was created. A Δ-cycle is an infinitesimal progress in simulator time [46]. Events, like a signal change, are scheduled to be updated after all Δ-cycles at this time point has been processed. In this case, at the critical point t_1, the change in q and r creates Δ-cycles after which the inverter and the XOR output will be updated. Since the inverter output does not depend on any other signal, after the Δ-cycle the output is updated. The new value of w creates another Δ-cycle after which *out* is updated. With the new input values, *out* is calculated and because no other event happens at t_1, the output of the XOR-gate is written one Δ-cycle later to the output port. Since the output is still zero, no further events will be produced. Following blocks connected to *out* would not be triggered at t_1. Fig. 2.4 visualizes the concept of the event-driven time progress with Δ-cycles.

The advantage of event-driven simulations is that on a change of a signal only parts which are sensitive of this signal change need to be recalculated and the rest of the systems stays in the state it was before. Furthermore, no equation systems need to be solved but the processes calculate their output locally in a single step. Therefore, event-driven simulators overcome the problems of matrix-based simulations regarding the computational performance. But the gain in CPU time is treated for some restrictions and disadvantages. First, the systems are described on a more abstract level and thus the simulation results lose some accuracy compared to SPICE-like simulations [47]. Furthermore, closed loop systems without loop delay cannot be modelled because it would result in an endless loop in simulation and last,

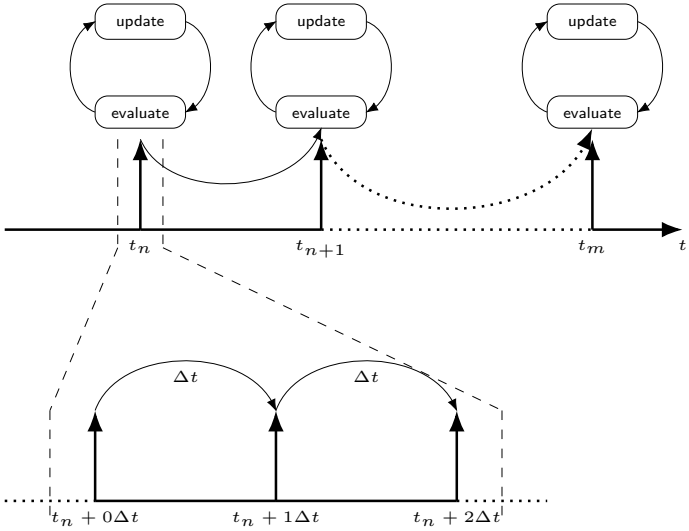


Figure 2.4: Event queue with Δ-cycles.

models for the different blocks in the systems need to be created. For synthesized digital systems, the standard libraries contain models for event-driven simulations but for custom designed circuits, a model in an HDL must be created which is done in most cases manually.

2.8 Simulation of Mixed-Signal Systems

A system consisting of both digital and analog components is called a mixed-signal system. Due to the difference in simulating digital and analog circuits, the analysis of a mixed-signal system is a very challenging part of the verification process of modern SoCs [45].

There are several approaches to simulate such a system. A naive way would be to simulate the digital part with an event-driven simulator, extract the control signals for the analog part and feed the analog simulator with these stimuli for simulating the analog circuitry. This approach only works if there are no feedback signals from analog to digital, otherwise the digital simulation is incorrect because the stimuli from analog are missing.

A mixed-mode simulator is a tool which can handle both analog matrix-based simulation and digital event-driven simulation [47] which is a challenging task because it has to deal with the great differences the simulation methods of digital and analog circuits. First, the time concept in event-driven simulators is different to SPICE-like ones. In digital systems time is discrete set of events with a minimum time step while in analog systems time is continuous. A mechanism must be implemented to synchronize both time-domains. There are essentially two methodologies. In the *backtracking* algorithm [47] the time of both the analog and the digital simulator advances independently from each other. If an event in the analog or digital domain occurs which influences the other domain, the simulator has to step back to the time point of the event and recalculate the signals from this point on. The *lockstep* method [48] uses a fixed time grid for both simulators. The analog solver can decrease the time steps if it runs in convergence issues. In addition to the difference in the timing concept, the signal representation in analog and digital systems is different. In matrix-based analog simulators, both voltages and currents are continuous values (represented as 64-bit floating point values) whereas in event-driven digital simulations the signals have discrete values ('0','1','z','x'). At the interface points between analog and digital system, it can be made use of so-called *interface elements* or *connect modules* [49] which translates digital signal representation to voltages and currents and vice-versa. These elements can be inserted either manually or automatically by the mixed-mode simulator. A major issue when going from digital to analog domain is, that the logic states do not possess the current/impedance information which is needed for the analog solver which can lead to a loss of accuracy. Special derivatives of HDLs have been developed for mixed-mode simulations such as *Verilog AMS* and *VHDL AMS* which can model systems in analog as well as in digital domain. They can help to speed up simulation when replacing a transistor-level implementation in a simulation or they can be used for virtual prototyping.

For small to medium sized mixed-signal systems, a mixed-mode simulator is well-suited but unfortunately, for really large systems, the analog solver slows down the entire simulation and is the bottleneck [45]. In Fig. 2.5 the performance issue for mixed-signal simulations is shown. Since the analog solver scales superlinear with the complexity of the analog circuit but the run time of the digital simulator increases only linear with the size of the digital system, the mixed-mode simulator needs much more time for solving the analog part than for the digital part with increasing system complexity.

Therefore, the concept of developing macromodels of analog and RF circuits and using RNM techniques to simulate the complete mixed-signal system in an event-driven simulator has been widely applied over the last years [50] [45] [51].

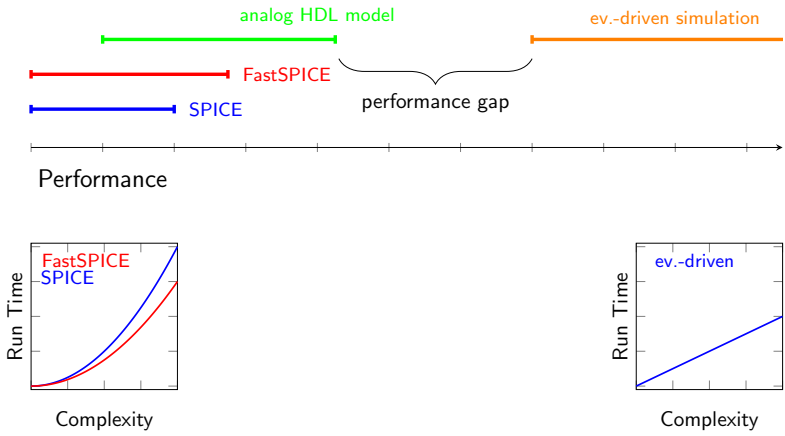


Figure 2.5: Performance gap in mixed-mode simulations [50].

2.8.1 Real Number Modelling

In Real Number Modelling (RNM) analog circuits are modelled as a signal flow model [50]. The analog block is abstracted in such a way that the relation between in- and outputs can be described explicitly by equations so that no matrix inversion and iterative equation solving is needed. Furthermore, the time-continuous operations are replaced or approximated by sampled operations. Taking the amplifier from Fig. 2.2 as example, it could be replaced under some constraints (neglecting the capacitance for the moment and assuming a fixed operating point) by a simple multiplication of the input voltage with the gain A of the amplifier (see eq. 2.6 and listing 2.1).

$$V_2(t_{ev}) = A \cdot V_{in}(t_{ev}) \tag{2.6}$$

The gain could be either calculated analytically or it can be extracted from a stand-alone circuit simulation of the amplifier. Since, when neglecting all capacitances, there are no energy storing elements in the circuit, and it reacts immediately on an input event. Thus, it can be described as a block which is sensitive on the input signal V_{in} and updates the signal V_2 every time the input changes.

Several HDLs like SV [52] and VHDL [53] support the use of double precision floating point variables. Using the User-Defined Nettype (UDN) implementation, which was

first established in the SV 2012 standard, custom made functions for resolving the nets, which define e.g., what should happen if more than one block drives a net, can be created. The SV *EE_pkg* package for example implements in- and output resistances and differs between current and voltage [50] to get a more realistic electrical behaviour.

Of course, the above-described amplifier model is an extremely simplified description of the real circuit. Depending on the purpose of the simulation, more or less complex models are feasible. If only functional verification is required, it is better to use fast and therefore less complex models [51] since only the system's functionality is of importance, e.g. the correct connectivity and whether the control bytes have the right endian, whereas for a behavioural simulation the model behaviour should be as close to the real circuit behaviour as possible. The disadvantage of event-driven RNM is, that the feedback from output to input cannot be modelled because it is limited to an unidirectional signal flow. This is no big drawback for digital systems but can create difficulties when modelling analog circuits because there are circuits where the output strongly interacts with the input, e.g., a passive mixer. Moreover, for every analog module a model needs to be creates which is in general done manually which creates a large effort.

The following two chapters will discuss signal representations and modelling techniques suited for event-driven analog models which on the one hand allow a good simulation performance in terms of simulation speed and on the other hand enable accurate behaviour description compared to the transistor-level implementation.

```
1 module amplifier (V_in,V_2);
2
3 input real V_in;
4 output real V_2;
5 const real A = 10.0;
6
7 always @(V_in) V_2 <= A*V_in;
8
9 endmodule
```

Listing 2.1: RNM of an amplifier in SV.

CHAPTER 3

SIGNAL DESCRIPTION

In signal processing, a signal is a function that provides information about a system's behaviour. In the context of electrical engineering it often refers to a time-varying voltage, charge, current or magnetic flux but it can also hold non-electrical properties like phase, acoustic sound levels or the humidity of the ground [54]. The information, the signal carries, can come from the measurement of a physical process, e.g. the humidity, but the information can also be impressed by a modulation in order to transmit information. Furthermore, a distinction must be made between continuous-time and discrete-time signals. Discrete-time signals are sequences of values whereas continuous-time signals are described by functions over the continuous time t. Analog systems process continuous-time signals, while a digital system is a good example of a discrete-time system.

When modelling systems on a higher abstraction level, it can be beneficial to describe a system's property in the domain which is best suited for this special kind of system, either because the modelling will become easier or because this special property is the most expressive one. For example, a Phase-Locked Loop (PLL) is strongly nonlinear in the voltage domain but only weakly linear in the phase domain [55]. Therefore, it is beneficial to describe a PLL in the phase domain to avoid handling the strong nonlinearities which are difficult to describe. Furthermore, the relevant output property of a PLL is the phase and not the voltage. The signal abstraction level plays another important point in high-level modelling. While for an analog system it is important that the resolution of the signal is high, for digital systems only the differentiation between two levels is important. And lastly, when dealing with nonlinear systems it might be beneficial to separate between signals with large amplitudes and signals with amplitudes so small that nonlinear effects are negligible.

By Fourier transform or Laplace transform, the signal description can be moved from a time dependent series to a function depending on the frequency f or the Laplace variable s. These transformations can be utilized when describing different systems. E.g., a nonlinearity is best described in time domain whereas it is beneficial to describe a linear system in frequency domain.

Since the modelled systems should be simulated on a computer, the time and value continuous analog signals first must be discretized in order to be representable there. Continuous values are approximated as real floating-point type with 64 bits resolution (a C/C++ *double* data type). The quantization error which is left is usually so small that it does not influence the simulation results in most cases. More care has to be taken for the quantization of time. The Nyquist–Shannon theorem [56] gives the sufficient condition for sampling a continuous time signal with a finite bandwidth without losing information. For equidistant sampled signals it states that the sampling rate f_s must be higher than half of the bandwidth BW of the signal. When describing a time continuous system on with a sampled model it has to be ensured that the Nyquist criterion is not violated.

For simulation, a C++ class *AMS_Signal* was implemented (see listing 3.1). It serves as parent class for several other signal descriptions and defines what property (*signalDomain*) the signal describes.

```
1 class AMS_Signal : public AMS_Object {
2 public:
3 typedef enum {VOLTAGE_CURRENT, PHASE, FREQUENCY, UNDEFINED} domainOfSignal;
4 [...]
5 protected:
6 domainOfSignal signalDomain;
7 };
```

Listing 3.1: Implementation of the *AMS_Signal* class.

The whole signal class structure from the simulation tool chain is shown in Fig. 3.1. The different signal descriptions and their application area will be discussed in the following.

3.1 Signal Sampling and Fundamentals of Signal Description

As mentioned before, the Shannon-Nyquist theorem states that a band-limited time continuous signal can be fully described by a series of sampled signal values if the sampling rate is high enough. It can be distinguished between equidistant sampling and non-uniformly sampling (see. Fig. 3.2).

[1] The *AMS_SmallSignal* signal description is discussed in chapter 6.

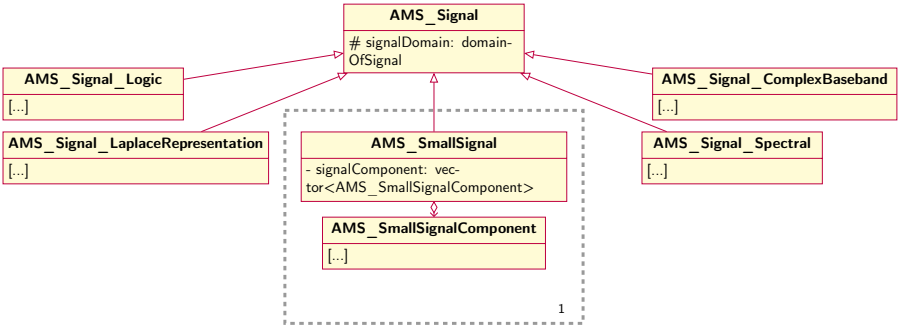
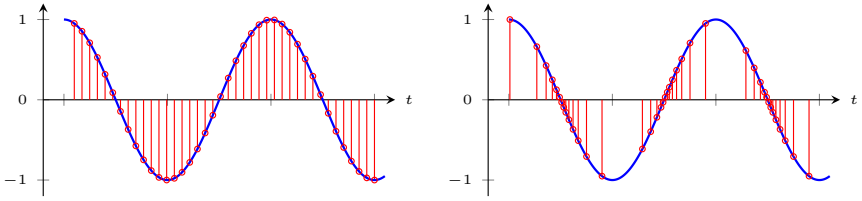


Figure 3.1: Different signal descriptions on C++ level and their relations.



(a) Uniformly sampled signal.

(b) Non-uniformly sampled signal.

Figure 3.2: Signal sampling.

Both ways of sampling have their advantages. When dealing with equidistant sampled signals, the event control is easier to implement and a Fast Fourier Transformation (FFT) can be applied directly to transform the result in the frequency domain.

Non-uniformly sampling can for example be more efficient but at cost of a more complex event-generation. In [57] a PLL model using a Quantized-State System (QSS) approach is proposed where events are generated if a state of the system varies by a threshold value from its previous state which results in non-equidistant sampling. The QSS description is especially well-suited for simulating heavily discontinuous models [58]. The disadvantage of this approach is that it relies on the absolute value of a state change so that it is difficult to find a threshold in state change ΔQ for the next event generation when handling small and large signals together. If the value ΔQ is too large, small signals will not trigger events and information is lost. If choosing a value too small, large signals will trigger an unnecessary high amount of events and slow down the simulation. This issue can be solved by using step sizes depending on the value of the states as proposed in [59]. Also, for handling

stiff systems [58] proposed the backward QSS methodology inspired by stable ODE solvers like the backward Euler method.

To reduce the sampling rate for a faster simulation, the way of describing the signals can also be improved. The simplest way is to use a Piece-Wise Constant (PWC) description. It is assumed, that the signal value is constant until the next sampling point, resulting in a signal approximated by step functions. A more sophisticated solution is to describe the signal in a Piece-Wise Linear (PWL) fashion [60] or to use a higher order polynomial approximation [61]. An approach using a different set of basic functions instead of polynomials is the *XREAL* signal description which is in detail described in section 3.4.

If a signal description uses no a-priori knowledge on the signal's properties and simply approximates it as series of signal values obeying the Nyquist–Shannon theorem, it is called passband signal description. Other ways of describing signals are discussed in the following.

3.2 Baseband Equivalent Signal Description

If no a-priori knowledge is available on the signal properties, the passband signal description is a good way for the use with RNM. But RF signals are usually band-pass signals with the spectrum centred around a high carrier frequency f_c and the bandwidth of those signals is typically small compared to f_c (see Fig. 3.3). If an RF signal is naively described as PWC or PWL passband signal, many sampling points, thus many events are needed. As an example, a signal with the bandwidth of 1 MHz

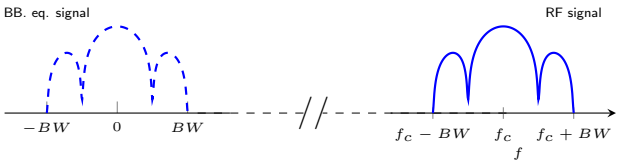


Figure 3.3: Spectra of RF and corresponding BB.eq. signal.

and the carrier frequency of 2.5 GHz must be sampled at least with a sampling rate of 5 GHz to avoid aliasing effects. Since every modulated RF signal $x(t)$ can be described as shown in eq. 3.1 [14], the slowly varying complex envelope $BB(t)$ can be separated from the fast-oscillating carrier $\exp(j\omega_{RF}t)$, which is utilized by the baseband equivalent signal description.

$$x(t) = I(t) \cdot \cos(\omega_{RF} t) + Q(t) \cdot \sin(\omega_{RF} t) = \Re\left\{BB(t) \cdot \exp(j\omega_{RF} t)\right\}$$
$$\text{with:} \quad BB(t) = I(t) + jQ(t) \tag{3.1}$$

Only the three real-valued parameters $I(t)$, $Q(t)$ and ω_{RF} needs to be passes between the models. Since the bandwidth of the signal is low and ω_{RF} is fixed, $I(t)$ and $Q(t)$ will only vary slowly and there is no need for a high sampling rate. To stay with the example above, a sampling rate of only 2 MHz would be necessary which is a factor of 2500 lower than for the passband signal description. The modulated RF signal can be imagined as phasor rotating very fast in a fixed coordinate system. But if the coordinate system itself is rotating with the RF frequency, the projection of the phasor on the axis of the rotating system, which equals I and Q, is only slowly varying (see Fig. 3.4).

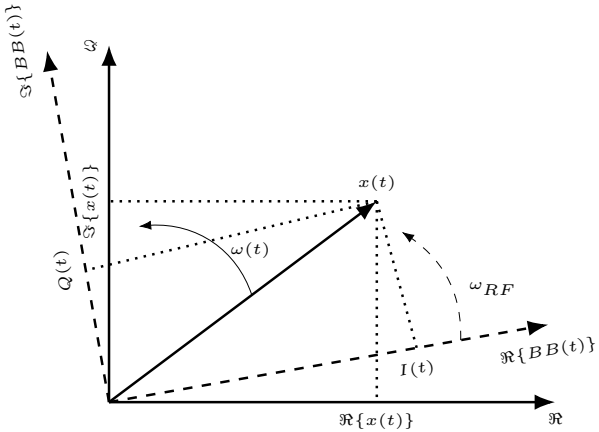


Figure 3.4: Graphical representation of a baseband equivalent signal.

3.3 Spectral Signal Description

The baseband equivalent signal description enables a high simulation speed-up compared to the passband description, but it comes with the drawback, that only signals which are centred around one frequency can be described. This has the disadvantage that nonlinear operations cannot be depicted, since nonlinearities generate harmonic frequency components which do not fit in this signal description.

Furthermore, scenarios with multiple input tones are impossible to describe. The models cannot cover intermodulation products.

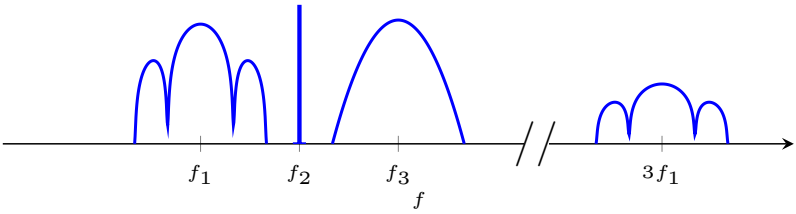


Figure 3.5: Exemplary RF spectrum with multiple centre frequencies.

To overcome this issue, the idea of a spectral signal was proposed in [62] and [63]. Here, the baseband equivalent approach was extended to more than one carrier frequency, allowing the usage of nonlinear operations on the signal. The spectral signal can be formulated in the form

$$x(t) = \sum_k c_k(t) \cdot \exp(j\omega_k t), \tag{3.2}$$

where k is the number of centre frequencies. This number depends on the number N of incommensurate fundamental tones λ_i and the number of harmonics per tone $K(n)$. The set of carrier frequencies ω_k is the set of all frequencies that can be constructed like eq. 3.3.

$$\begin{aligned} \omega_k &= m_1 \cdot \lambda_1 + m_2 \cdot \lambda_2 + ... + m_N \cdot \lambda_N \\ -K(1) \leq m_1 \leq K(1), -K(2) &\leq m_2 \leq K(2), ..., -K(N) \leq m_N \leq K(N) \\ m_1, m_2, ..., m_N &\in \mathbb{Z} \end{aligned} \tag{3.3}$$

The weights $c_k(t)$ set the time-varying complex-valued amplitude for each centre frequency. A small example should demonstrate how nonlinear scenarios can be covered using a spectral signal description. A nonlinear amplifier with the transfer function $y(x) = a_1 x + a_2 x^2$ is fed by the sum of two sinusoidal signals $x(t) = A_1 \cdot cos(\lambda_1 t) + A_2 \cdot sin(\lambda_2 t)$. The input can be described as spectral signal as shown

in eq. 3.4.

$$x(t) = \sum_{k=1}^{4} c_k(t) \cdot \exp(j\omega_k t) \quad \text{with:}$$
$$\omega_1 = -\omega_2 = \lambda_1, \omega_3 = -\omega_4 = \lambda_2, c_1 = c_2 = \frac{A_1}{2}, c_3 = c_4^* = \frac{jA_2}{2} \tag{3.4}$$

The output signal y of the nonlinearity can be written as:

$$y(t) = \sum_{k=0}^{12} c_k(t) \cdot \exp(j\omega_k t) \quad \text{with:}$$
$$\omega_0 = 0, \omega_1 = -\omega_2 = \lambda_1, \omega_3 = -\omega_4 = \lambda_2, \omega_5 = -\omega_6 = \lambda_1 - \lambda_2,$$
$$\omega_7 = -\omega_8 = \lambda_1 + \lambda_2, \omega_9 = -\omega_{10} = 2\lambda_1, \omega_{11} = -\omega_{12} = 2\lambda_2, \tag{3.5}$$
$$c_0 = \frac{a_2(A_1^2 + A_2^2)}{4}, c_1 = c_2 = \frac{a_1 A_1}{2}, c_3 = c_4^* = \frac{j a_1 A_2}{2}, c_5 = c_6^* = \frac{-j a_2 A_1 A_2}{2},$$
$$c_7 = c_8^* = \frac{j a_2 A_1 A_2}{2}, c_9 = c_{10}^* = \frac{a_2 A_1^2}{4}, c_{11} = c_{12}^* = \frac{a_2 A_2^2}{4}$$

It becomes clear, that since only real-valued signals are regarded, there is no need to store the coefficients from both sides of the spectrum. A single-sided representation contains all information needed since the coefficients on the other side of the frequency axis are the complex conjugated values.

While it is possible to calculate the output for some classes of nonlinear transfer functions directly in the spectral domain, for a general evaluation of nonlinearities the computation of the operation in the time domain is necessary (see Fig. 3.6) as it is done in most HB algorithms [43].

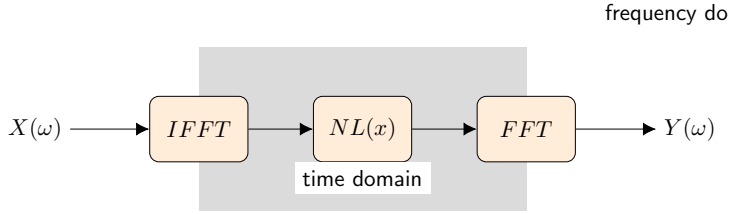


Figure 3.6: Evaluation of nonlinearities in the time domain.

When regarding c_k as time-varying Fourier coefficients of the signal, the spectral signal can be transformed via a multidimensional IFFT in time-domain. Then, the

nonlinearity is evaluated and finally the result is transformed back with a FFT in the frequency-domain. Of course, the number of harmonics arising from some nonlinear operations can theoretically be infinite. To make the operation numerically tractable, the result needs to be truncated after a defined number of harmonics. Furthermore, it must be noticed that, when using the FFT it is automatically assumed that the signal is in its quasi-periodic steady-state because the multidimensional FFT only produces error free results for these kind of signals. Usually, this is no big constraint for RF signals because their Bandwidth (BW) is in most cases much lower compared to the carrier frequency which allows a steady-state assumption for nonlinearity evaluation.

The spectral signal description was implemented in C++ as *AMS_Signal_Spectral* class. The coefficients c_k are stored a multidimensional array with the dimension number equals the number of fundamental tones. The array itself is stored as linear array in the in row-major order. Fig. 3.7 gives an example for the coefficient arrangement of a spectral signal with two incommensurate fundamental frequencies. The coefficient storage was chosen this way so that the multidimensional FFT algorithm *FFTW* [64] can efficiently work with the spectral signal.

When adding two spectral signals with different fundamental frequencies, first, the resulting signal structure must be organised in a way that the resulting coefficient array can hold all coefficients for all frequency combinations. Then the coefficients of both signals will be copied in the new array. If components of the same centre frequency occur, they are added.

3.3.1 Application Area

Unlike the baseband equivalent signal description, the spectral signals can describe a wide range of signals. Since the nonlinear operations on them requires the use of the multidimensional FFT, the more Fourier coefficients needed to properly describe the signal, the more computational power is needed and the slower the simulation becomes. The same is true for the number of incommensurate fundamental frequencies. The representation is best suited for the description of a superposition of low to medium BW modulated RF signals because they can be described by only a few fundamental frequencies and usually a small number of harmonics when dealing with weakly nonlinear systems. If the BW is small, the coefficients update rate is small which increases simulation speed. Since it is assumed that the signal is in its quasi-periodic state between the sampling points, it is ill-suited for describing fast transient events. For systems where the transient behaviour is of interest, other signal descriptions are better fitting.

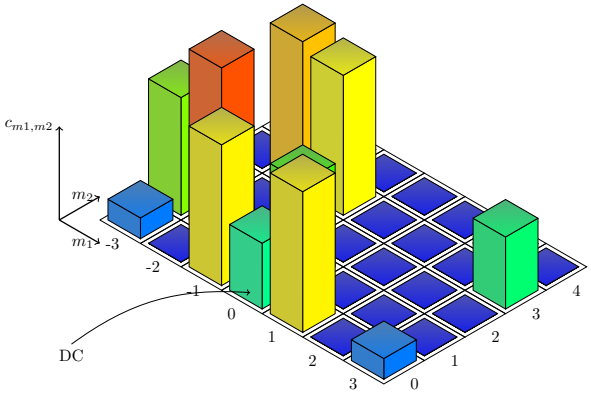


Figure 3.7: Coefficients of a spectral signal with two incommensurate fundamental frequencies. The indices m_1 and m_2 denote the harmonics of the fundamentals. The *FFTW* algorithm requires that from the last fundamental frequency coefficients only the non-negative indexed coefficients are stored.

3.4 *XREAL* Signal Description

In order to reduce the number of events, a PWC signal description can be replaced by a PWL [60] or a higher order polynomial approximation of the signal. But still, these descriptions are only approximating the real signal and the accuracy of the description depends on the distance between two sampling points. The higher the number of grid points, the more accurate the approximation is.

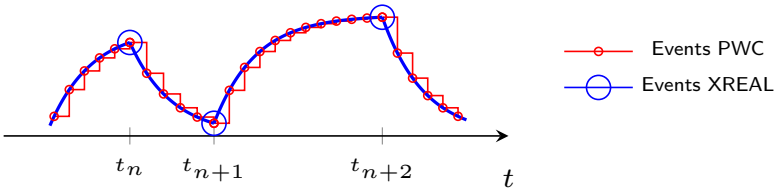


Figure 3.8: Number of events needed for a PWC and *XREAL* signal description.

A different approach to describe signals with only a few grid points was first described in [65]. It is assumed, that for simulation and verification purpose, most analog signals can be sequence-wise decomposed in sums of step, polynomial, sinusoid, and

exponential functions. By reducing the signal space on these functions, sets of only four parameters are necessary for the signal description which allows a compact implementation on a computer. The *XREAL* signal descriptions are only updated if the shape of the waveform changes which lowers the number of events drastically.

Each sequence between to grid points can be expressed as waveform with the shape:

$$x(t) = \sum_{i=0}^{K} \frac{b_i}{(m_i - 1)!} \cdot (t - t_{0,i})^{m_i - 1} \cdot \exp(a_i(t - t_{0,i})) \cdot u(t - t_{0,i}), \tag{3.6}$$

with the complex amplitudes b_i, the poles a_i, the pole multiplicities m_i and the times of occurrence $t_{0,i}$. $u(t)$ denotes the step function, which is 0 for $t < 0$ and 1 otherwise. In the s-domain, a sequence of the signal is described by

$$X(s) = \sum_{i=0}^{K} \frac{b_i}{(s - a_i)^{m_i}} \quad \text{occuring at } t_{0,i}. \tag{3.7}$$

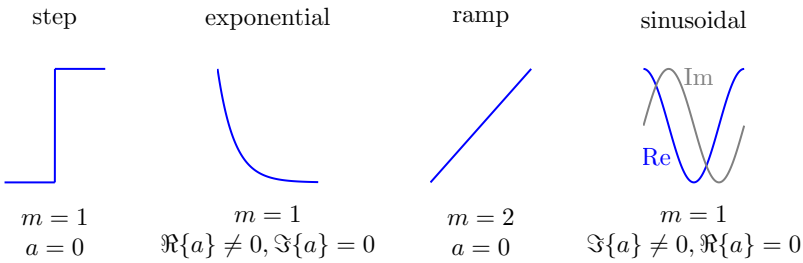


Figure 3.9: Commonly used signals described by the *XREAL* signal description.

The advantage of this formulation is, that it is not only an approximation of the real signal but exactly describes it. Furthermore, the *XREAL* signal description only needs to be updated if the shape of the signal changes. This means, the number of events needed is reduced to an absolute minimum. The value of the signal at a specific point of time t_x can be calculated by replacing t with t_x in eq. 3.6 and evaluating the expression. Also, switching between a time-domain and an s-domain view on the signal is straightforward which allows modelling the systems in the domain which fits best, e.g., filters can be described in the s-domain whereas strong nonlinear blocks like comparators can be modelled in time-domain.

The *XREAL* signal description is implemented using a vector of parameter sets

(eq. 3.8) which is passed between the blocks.

$$x(t) =< [b_1, a_1, m_1, t_{0,1}], .., [b_i, a_i, m_i, t_{0,i}], .. > . \tag{3.8}$$

The parameter sets are organised as *struct* while the collection of all parameters for an *XREAL* signal are stored in a *vector* from the Standard Template Library (STL). This has the advantage that the number of parameter sets can be dynamically varied during the simulation phase depending on the described signals.

```
class AMS_Signal_LaplaceRepresentation: public AMS_Signal {
public:

typedef struct {
complex<double> a,
b;
double m;
double t_0;
} parameter_set;

std::vector<parameter_set> signal_representation;

[...]
```

Listing 3.2: Implementation of the *AMS_Signal_LaplaceRepresentation* class.

3.4.1 Operations on the Signal

To use *XREAL* signals for simulating systems, a few operations like addition and multiplication with a constant need to be defined which are commonly used when modelling circuits. Since each update describe the signal for a complete sequence and because a parameter vector instead of a scalar is changed, some operations are not as straightforward as for a PWC signal description.

The addition of two *XREAL* signals is done by merging the two vectors of parameter sets to a resulting one. Parameter sets can be combined if they only differ by the amplitude b (3.9).

$$\begin{aligned} &< [b_1, a_1, m_1, t_1], .., [b_i, a_i, m_i, t_i], .. > + < [g_1, a_1, m_1, t_1], .., [b_j, a_j, m_j, t_j], .. > \\ &= < [b_1 + g_1, a_1, m_1, t_1], .., [b_i, a_i, m_i, t_i], [b_j, a_j, m_j, t_j], .. > . \end{aligned} \tag{3.9}$$

A *XREAL* signal is multiplied with a constant by multiplying the amplitude b of every parameter set with the constant factor c (3.10).

$$\begin{aligned} c\cdot &< .., [b_i, a_i, m_i, t_{0,i}], [b_{i+1}, a_{i+1}, m_{i+1}, t_{0,i+1}], .. > \\ = &< .., [c \cdot b_i, a_i, m_i, t_{0,i}], [c \cdot b_{i+1}, a_{i+1}, m_{i+1}, t_{0,i+1}], .. > . \end{aligned} \tag{3.10}$$

Sometimes it might happen, that an *XREAL* signal is composed of sets with different starting times $t_{0,i}$, e.g., if the signal was composed of two different sources. For some operations, it is necessary that all sets in the signal starts at the same time. Therefore, re-timing must be applied to some sets. Eq. 3.11 shows how a set is modified when replacing the time of occurrence t_1 by a different starting point t_2 while the two signal representations stay equivalent.

$$\begin{aligned} &\frac{b}{(m-1)!} \cdot (t-t_1)^{m-1} \cdot \exp(a(t-t_1)) \\ &= \sum_{k=0}^{m-1} \binom{m-1}{k} \frac{b\exp(a(t_2-t_1))}{(m-1)!} (t_2-t_1)^{m-1-k} \cdot (t-t_2)^k \cdot \exp(a(t-t_2)). \end{aligned} \tag{3.11}$$

As eq. 3.11 reveals, the single set is replaced by a sum of sets which looks on implementation level as follows:

$$\begin{aligned} &[b, a, m, t_1] \\ \rightarrow &<[\frac{b\exp(a(t_2-t_1))}{(m-1)!}(t_2-t_1)^{m-1}, a, 0, t_2], \\ &[\frac{(m-1)b\exp(a(t_2-t_1))}{(m-2)!}(t_2-t_1)^{m-2}, a, 1, t_2], \\ &\ldots \\ &[\frac{b\exp(a(t_2-t_1))}{(m-1)!}, a, m-1, t_2] > \end{aligned} \tag{3.12}$$

For the sake of completeness, also the multiplication of two *XREAL* signals will be discussed even if it is rarely used in the practical modelling of RF systems since mixer and multipliers are best described with the spectral signal description and nonlinearities applied on *XREAL* signals are modelled avoiding multiplication as shown later. For multiplying two *XREAL* signals, it is necessary that the times of occurrence of the parameter sets are equal. If they are not, a re-timing must be

applied first. For the sake of clarity, the multiplication of two single parameter sets will be derived. The product of two *XREAL* signals with more than one set each can be calculated as in eq. 3.13.

$$s_1 \cdot s_2 = \sum_k set_k \cdot \sum_l set_l = \sum_k \sum_l set_k \cdot set_l \tag{3.13}$$

The multiplication of two parameter sets is explained in eq. 3.14.

$$\begin{aligned} & set_1 \cdot set_2 \\ &= \frac{b_i}{(m_i-1)!} \cdot (t-t_0)^{m_i-1} \cdot \exp(a_i(t-t_0)) \cdot \frac{b_j}{(m_j-1)!} \cdot (t-t_0)^{m_j-1} \cdot \exp(a_j(t-t_0)) \\ &= \frac{\tilde{b}}{(\tilde{m}-1)!} \cdot (t-t_0)^{\tilde{m}-1} \cdot \exp(\tilde{a}(t-t_0)). \\ &\text{with:} \quad \tilde{a} = a_i + a_j, \quad \tilde{m} = m_i + m_j - 1 \quad \tilde{b} = b_i b_j \frac{(m_i+m_j-2)!}{(m_i-1)!(m_j-1)!} \end{aligned} \tag{3.14}$$

The calculation of the time derivative of an *XREAL* signal can be explicitly computed since it is built up from a sum of analytic functions for which the derivatives are known (see. eq. 3.15). Higher order derivatives can be computed recursively.

$$\frac{dx}{dt} = \begin{cases} \sum_{i=0}^{K} \frac{b_i \cdot (m_i-1)}{(m_i-1)!} \cdot (t-t_{0,i})^{m_i-2} \cdot \exp(a_i(t-t_{0,i})) & \text{for } m \geq 2 \\ \quad + \frac{b_i \cdot a_i}{(m_i-1)!} \cdot (t-t_{0,i})^{m_i-1} \cdot \exp(a_i(t-t_{0,i})). & \\ \sum_{i=0}^{K} b_i \cdot a_i \cdot \exp(a_i(t-t_{0,i})) & \text{for } m = 1 \end{cases} \tag{3.15}$$

Besides the basic operations shown above, another important operation on *XREAL* signals is the calculation of the point where a signal crosses a specific value x_{thres}. This is needed, e.g., for the description of comparators or frequency dividers modelled in the phase domain. Since *XREAL* signals are built up from a sum of complex exponentials and polynomials, the analytical calculation of a zero crossing is not

possible in most cases but on a section where the signal is either monotonic rising or falling, an approximated solution can be calculated with the Newton algorithm (eq. 3.16, Fig. 3.10). The crossing time t_{cross} can be iteratively calculated. The iteration might be stopped when the absolute value $|x(t_n) - x_{thres}|$ is below a certain error bound or the change in t_n from one iteration to the next is less than the simulation time resolution.

$$t_{n+1} = t_n + \frac{x(t_n) - x_{thres}}{x'(t_n)} \tag{3.16}$$

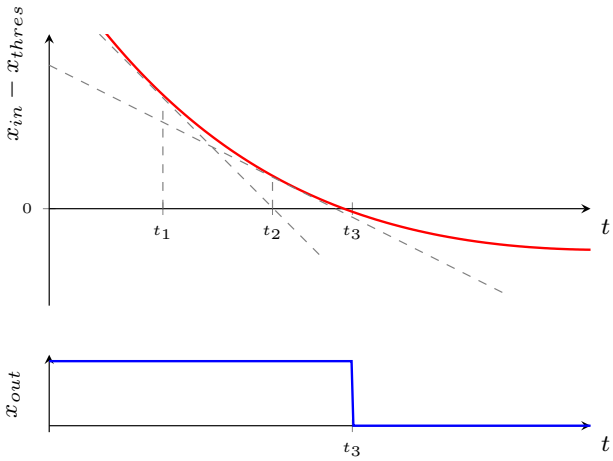


Figure 3.10: Calculation of crossing point.

Since for some signals there cannot be guaranteed that it is monotonic, the search for the crossing point is first started with a modified bisection method until an interval with a zero crossing and where the sign of the derivatives at start and endpoint is the same is found. Then, the Newton algorithm is applied. The step sizes and interval sizes are chosen depending on the poles of the *XREAL* signal, because they determine how fast the signal can change.

All mentioned operations were implemented in the *AMS_Signal_LaplaceRepresentation* class so that the other C++ models can revert on them. From this set of basic operations, more complex operations can be derived. In chapter 4 the implementation of nonlinearities and filters in combination with *XREAL* signals are described.

3.4.2 Application Area

Since linear operations occurring in analog mixed-signal systems will always map *XREAL* compatible signals on *XREAL* compatible signals (proof in A.1), the *XREAL* signal description is really well suited for systems where the input waveforms are already present in the *XREAL* signal description and the system is linear or can be approximated by a (piece-wise) linear system. Furthermore, *XREAL* signals are superior to baseband equivalent or spectral signal descriptions if the transient behaviour is from interest because baseband equivalent as well as spectral signals assume a steady-state and are therefore ill-suited for such systems.

It has been shown that this is the case for systems which are driven by hard switching signals (signal is a sequence of step functions) or systems with other a-priori known input waveform shapes, e.g., a DC-DC converter or a PLL which is nearly linear in the phase domain. For circuits with arbitrary signal shapes like the input of a LNA, the *XREAL* signal offers no advantage in comparison to other description because an arbitrary signal is not necessarily describable by a sum of *XREAL* parameters sets, thus the signal must be sampled and can be piece-wise approximated by an *XREAL* description. This results in similar sampling rates as needed for PWC or PWL description but with the computational overhead of the *XREAL* operations.

As shown above some operations like re-timing of two *XREAL* signals cause the number of parameter sets to increase rapidly if the pole multiplicity of some sets are high which enlarges the computational effort and slows down the simulation. Luckily, these kinds of signals are very rare in most systems and mostly only m-factors of one or two occurs.

3.5 Logic Signals

For describing value-discrete signals, a special signal description similar to the Verilog *wire* or SV *logic* datatype is used. The signal can take four different values:

- '0', if the signal represents a logical value of 0.
- '1', if the signal represents a logical value of 1.
- 'Z', if the signal represents a high ohmic state, e.g., when a node is not connected or the high ohmic state of a tristate logic gate.
- 'X', if the signal value is unknown, e.g., before initialization or when a node is driven by two different signal values '0' and '1'.

Besides the signal value, also the logic domain and the logic (voltage) level are stored in the signal description. This helps identifying domain crossing problems and show points where level-shifters are missing. At the start of the simulation, the domain and the level of the signal is checked against the domain and level of each connected port. An error is thrown if they do not match.

3.6 Transformation Between the Different Signal Descriptions

When using different signal descriptions in one simulation, it is important, that they can be transformed into each other so that the different parts of the system model understand each other. For some pairs of representations, e.g., the translation between a 'logic' and the analog passband signal description, the transformation is straightforward and will not be discussed here. For reasons of space and clarity, only the most used transformation will be discussed in the following.

Since the spectral signal description is a generalization of the baseband equivalent signal representation, the transformation from baseband equivalent signal to spectral signal is straightforward. The baseband equivalent signal is just a spectral signal with one fundamental frequency and no harmonics. The other way around is more complicated because all frequency components from the spectral signal have to be mapped on $I(t)$ and $Q(t)$ entries around a specific centre frequency ω_c. The spectral signal can be regarded as sum of baseband equivalent signals and since the baseband equivalent signal representation is not unambiguous, all components of this sum can be expressed as baseband equivalent signal with the carrier frequency ω_c and the different signal parts can be added (3.17). A proper oversampling must be guarantee because $BB_i(t)$ is now changing at least with the frequency difference $\omega_i - \omega_c$ which can be much higher than the changing rate of the coefficients c_i.

$$\Re\left\{\sum_i c_i(t) \cdot \exp(j\omega_i t)\right\} = \Re\left\{\sum_i \tilde{BB}_i(t) \cdot \exp(j\omega_c t)\right\}$$
$$\text{with:} \quad \tilde{BB}_i(t) = c_i \cdot \exp(j(\omega_i - \omega_c)t) \tag{3.17}$$

Due to their description by sequence-wise functions, *XREAL* signals can be translated analytically into spectral signals. First, the fundamental frequency and the relating harmonics ω_k for the resulting spectral signal are determined. If it is known that the *XREAL* signal has a certain periodicity it is sensible to calculate the spectral values at multiples of its fundamental frequency.

Starting from a short-time Fourier transformation with a rectangular window function over the interval $T = \frac{2*\pi}{\omega_1}$, the derivation of the calculation of a timely varying spectral coefficients $c_k(t) = S(\omega_k, t_x)$ is shown in (3.18).

$$\begin{aligned} S(\omega_k, t_x) &= \frac{1}{T} \int_{t_x}^{t_x+T} \sum_{i=0}^{K} \frac{b_i}{(m_i-1)!} \cdot (t-t_{0,i})^{m_i-1} \cdot \exp(a_i(t-t_{0,i})) \cdot \exp(-j\omega_k t) dt \\ &= \frac{1}{T} \int_{t_x}^{t_x+T} \sum_{i=0}^{K} \frac{\tilde{b_i}}{(m_i-1)!} \cdot (t-t_{0,i})^{m_i-1} \cdot \exp(\tilde{a_i}(t-t_{0,i})) dt \\ \text{with:} \quad & \tilde{b_i} = b_i \cdot \exp(-j\omega_k t_{0,i}); \quad \tilde{a_i} = a_i - j\omega_k. \end{aligned} \tag{3.18}$$

All needed operations can be efficiently implemented using already defined operations on *XREAL* signals. The poles a_i of each parameter set needs to be replaced by $a_i - j\omega$ which costs only one subtraction and the amplitudes needs to be multiplied by $\exp(-j\omega t_{0,i})$. The resulting signal has to be integrated and scaled with $\frac{1}{T}$, which corresponds to a convolution with the weighted step function ($< [\frac{1}{T}, 0, 1, 0] >$) (see sec. 4.4.1.3). Then, the resulting *XREAL* signal is sampled with the rate $\frac{1}{T}$ and the difference between the actual value and the last value one period earlier is calculated. By calculating the Fourier integral using *XREAL* signals, the result is exact by definition and oversampling is not required.

The reverse transformation from baseband equivalent or spectral signal to an *XREAL* signal representation is shown in eq. 3.19 exemplarily for the baseband equivalent representation but the concept can be transferred to spectral signals as well.

$$\mathfrak{R}\left\{BB_i(t_{ev}) \cdot \exp(j\omega_c t_{ev})\right\} \rightarrow < \left[\frac{BB_i(t_{ev})}{2}, j\omega_c, 1, 0.0\right], \left[\frac{BB_i(t_{ev})^*}{2}, -j\omega_c, 1, 0.0\right] > \tag{3.19}$$

Since sinusoidal waveforms are depictable by the *XREAL* representation, on every event of the input baseband equivalent signal, the *XREAL* signal responds with two parameter sets. The poles of the sets are at $j\omega_c$ and $-j\omega_c$, respectively, and the complex conjugated amplitudes $\frac{BB_i(t_{ev})}{2}$ and $\frac{BB_i(t_{ev})^*}{2}$.

CHAPTER 4

MODELLING

4.1 Model structure

There exist a variety of HDLs to describe purely digital electronics like VHDL, Verilog or SystemC but when it comes to analog or mixed-signal systems the description of the circuits gets more difficult. Even though there are extensions for the mentioned HDLs which allows the modelling of analog behaviour, these analog parts are later simulated on a node-based simulator which is not suitable for large mixed-signal systems due to high computational effort needed to solve the equation systems. Therefore, the RNM approach like in [14] is extended in this work to enable a detailed and accurate description of analog and mixed-signal circuits without losing the advantage of an event-driven simulation. The model structure is based on the simulation framework proposed in [63] and extended here. The model creation in this method is split into two parts and the advantage of this approach is explained in the following. One part of the model is written in SystemVerilog (SV) because this HDL is natively supported in commonly used design and verification environments and thus, SV code can be easily embedded there. Furthermore, a notion of time and notations for expressing concurrency is implemented in every HDLs, so there was no need to take care of it during the development of the simulation framework. Since analog signal processing capabilities of SV is rather limited, the behavioural description of the blocks is sourced out to C++. SV allows to bind pre-compiled C++ code via the Direct Programming Interface (DPI) [66]. Thus, an SV model is able to access C++ functions and vice-versa.

C++ is a mighty language for complex calculations due to its vast amount of libraries and is therefore well-suited for all types of analog circuit description. Furthermore, its runtime performance is very powerful [67]. In Fig. 4.1 a performance comparison between four operations implemented once purely in SV and once in C++ interfaced via the DPI from SV side is shown. As can be seen, for simple operations like the addition of two real (double) numbers, the overhead from calling the function via the DPI clearly reduces the performance of the mixed-language approach. For more complex mathematical operations like the calculation of the sine-function, the

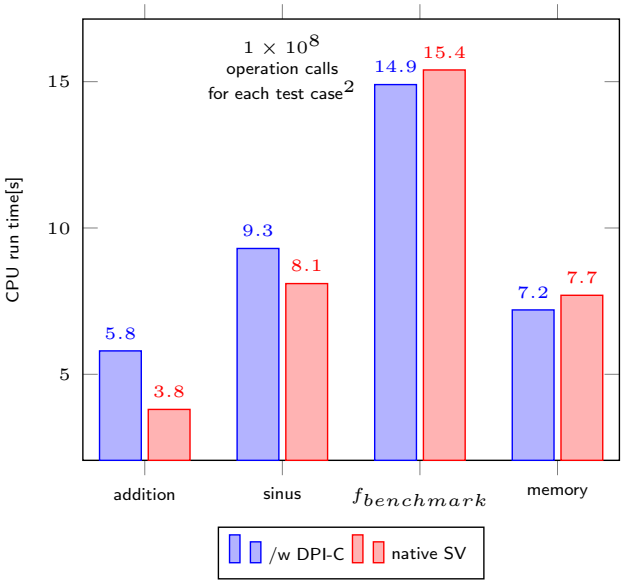


Figure 4.1: Comparison between a pure SV and a mixed SV/C++ implementation.

performance difference of both implementations is not so big anymore and when the computational complexity of the function gets even higher, the DPI approach is faster than the pure SV implementation. Also, in terms of memory management, the mixed-language implementation is superior to the native SV one. Thus, regarding runtime it is sensible to do all calculations with high complexity and memory management tasks in C++. Especially, for operations on spectral and *XREAL* signal description the C++ speed performance is advantageous.

In Fig. 4.2 the model structure is shown graphically. Every signal, net, port and block will be mapped on an object of a corresponding C++ class. The C++ classes are structured hierarchically which allows a standardized process of creating the objects from SV side and therefore offers the possibility to automatically generate the SV part of the models. Furthermore, the advantage of the class structure is that a big part of the code can be reused in the child classes and does not need to be recoded which offers a fast code adaption and is less error-prone.

[2]Each implementation was tested with 1×10^8 operation calls. The function $f_{benchmark}$ is defined as follows: $f_{benchmark}(x) = \exp((sin(x)^2 + cos(x)^2)/x)$. In SV the system functions were used. For benchmarking the memory management capabilities of the approaches the SV *queue* datatype was compared against the C++ std::queue container since they allow similar operations on them.

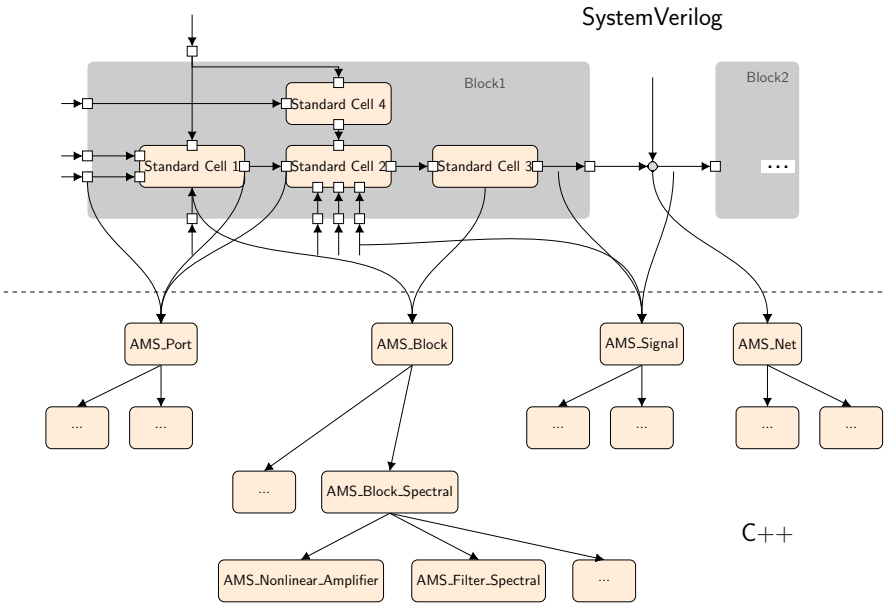


Figure 4.2: Model framework.

It has been shown, from personal experience and also from literature [68] [69], that it is advantageous to implement a library of generic modelling cells which describe fundamental analog behaviour like a nonlinearity, a filter, a differential to single-ended block, etc. These blocks can be customized by setting parameters to adjust to the desired behaviour. Based on these standard cells, more complex modules can be built by connecting the building blocks in a way they describe the behaviour of the modelled circuit. This way to build up models has the advantage that after the library of standard modelling cells is created and tested, it can be relied on the correct functionality of these building blocks. Furthermore, since the process of creating models is split into a SV and a C++ part, every time a new block is modelled, both the code on SV and C++ side must be touched. But once a complete library of standard cells exists, the creation of new models can be done completely on SV side because only the building blocks needs to be connected in the right way. This way the model creation can be done much faster and less error-prone compared to other approaches because only the connectivity between the standard models must be defined in SV. Since SV is a dedicated language to describe hardware, it is perfectly fitted for the structural description of the interconnection of the standard cells. For the same reason, the time control is managed in SV.

Listing 4.1 shows exemplarily the connection of standard modelling cells in a frequency-dependent, nonlinear LNA model. The differential RF inputs *RF_IN* and *RF_INX* are connected to differential to single-ended interface block (*AMS_ControlUnit_Spectral*). This block also checks the correct supply voltage, biasing and the digital enable signal and switches off the signal if the operating conditions do not apply. The signal then passes a filter (*AMS_SpectralFilterReference*) and a nonlinear amplifier cell (*LNA_Spectral_SE*) and is finally mapped to the differential output signals *LNARF_RFOUT* and *LNARF_RFOUTX* again. The different standard library cells are customized by setting parameters. For example, the nonlinear amplifier cell is defined by the transfer function $5.2469717645x - 5.6183091123x^3$ (*.nonlin_coeffs('0.0,5.2469717645,0.0,-5.6183091123*)).

During the verification of several RF transceiver chips developed at the IAS, the library of standard cells was created, enhanced and improved. The challenge is to develop parametrized generic models which describes all possible implementations of a class of circuits. E.g., an amplifier can be nonlinear, frequency-dependent and the gain can be controlled digitally by a bias current. All possibilities need to be taken into account to create a basic model cell which can cover all these effects. If a circuit cannot be described by cells of the standard library, the library must be enhanced by a cell which can cover the wished functionality. It has shown that most observed circuits can be described by connecting cells of the analog standard library and only a handful of blocks needed a completely manual creation of a new model from scratch.

Since over 90% of bugs responsible for a silicon re-spin are so-called C2I3 (control, configuration, interface, interconnect, and integration) errors [45], it is important to verify the genuine connectivity of the system toplevel and not an artificial created model of the system built for simulation purpose. Therefore, it is necessary to create models which are pin-compatible to the real underlying circuit. Furthermore, for complexity and simplicity reason, the real circuit implementation should be replaced on a low hierarchical level by their models. This approach guarantees that the connectivity between the building blocks on higher hierarchical levels is not changed by accident due to mistakes when creating the model and the complexity of the modelled blocks is kept low which decreases modelling errors.

For a faster model development and to avoid errors from manual work, a model template generator was developed. The generator collects the names of the block ports, asks the user which type of signal is carried by the port (spectral, logic, bias, ...) and which ports are differential ones. Furthermore, additional information like the supply and logic domain and tolerance values for the bias signals can be given. From this information, the generator creates a SV model template with the correct ports and port settings. Also, interface blocks mapping the differential signals on

```
[...]
signal_net net_spectral_in1[1:0];
alias net_spectral_in1[0] = RF_IN;
alias net_spectral_in1[1] = RF_INX;
signal_net net_spectral_out1[1:0];
alias net_spectral_out1[0] = LNA_RFOUT;
alias net_spectral_out1[1] = LNA_RFOUTX;
[...]
AMS_ControlUnit_Spectral #(.control_function("enable_high"), .controlType("DIFF2SE"),
.digital_domain_in(digital_domain), .digital_level_in(digital_level),
.num_spec_in(2), .num_spec_out(1), .num_logic(9), .num_bias(2), .num_supply(2),
.supply_domains('{supply_domain_AVDD,supply_domain_AVSS}), .supply_levels('{supply_level_AVDD,supply_level_AVSS}), .ref_values('{CSC_BIAS_5USRC_value,LNA_BIAS_25USRC_value}), .ref_tolerances('{CSC_BIAS_5USRC_tolerance,LNA_BIAS_25USRC_tolerance}), .ref_types('{CSC_BIAS_5USRC_type,LNA_BIAS_25USRC_type}), .v_cm(v_cm_out)
) ControlUnit_in1(.in(net_spectral_in1), .out(net_intern1), .supply(net_supply), .logic_in(net_logic), .bias_in(net_bias));

AMS_SpectralFilterReference #(.coefficient_file("LNA_CG_TO/LNA_2G4_TF.csv"),
.As('{0.0,1.26e4}), .Bs('{1.58e12,1.26e4,1.0}), .approx_degree(0))
Filter1(.in(net_intern1[0]), .out(net_intern2[0]));

LNA_Spectral_SE #(.noiseFile(noiseFile), .nonlin_order(3),
.nonlin_coeffs('{0.0000000000,5.2469717645,0.0,-5.6183091123}))
LNA1 (.in(net_intern2[0]), .out(net_intern3[0]));

AMS_ControlUnit_Spectral #(.control_function("default_on"), .controlType("SE2DIFF"),
.digital_domain_in(digital_domain), .digital_level_in(digital_level),
.num_spec_in(1), .num_spec_out(2), .num_logic(0), .num_bias(0), .num_supply(0),
.supply_domains('{"no_domain"}), .supply_levels('{0.0}), .ref_values('{10e-6}), .ref_tolerances('{1.0}), .ref_types('{"no_type"}), .v_cm(v_cm_out)
) ControlUnit_out1(.in(net_intern3), .out(net_spectral_out1), .supply(supply_dummy), .logic_in(logic_dummy), .bias_in(bias_dummy));
[...]
```

Listing 4.1: Connecting standard modelling cells in SV.

single-ended ones for internal processing are inserted. This template can then be manually completed by the user by inserting and connecting cells from the analog standard library. This procedure avoids extensive debugging of manually created models due to typos in the port names and speeds-up the model creation process. An example for a model of an LNA with the help of the template generator is shown in sec. A.2. Without counting the blank lines and comments, the template generator created 76 lines of code automatically while only 6 lines were manually written.

4.2 Description of common analog functionality

Due to time and space limitations in this chapter not every model created in the process of writing this thesis is discuss but the focus is put on the most used and most interesting functionality to model.

4.3 Modelling of memoryless nonlinearities

For RF circuits one key parameter of a circuit is its linearity or nonlinearity. In most cases, a linear transfer characteristic is aspired, e.g., for LNAs or filters, since nonlinear distortions create addition frequency components which might be difficult to remove but there are also circuits which relies on a nonlinear transfer function, e.g., some kind of mixers or envelope detectors. The focus in this section will be on memoryless nonlinearities. Systems with a nonlinear response which also depends on events happened in the past will be addressed in sec. 4.5.

There exist several methods [70] [71] [72], to characterize nonlinear transfer functions, but since most circuits in RF transceivers are only weakly nonlinear [72], an approach to approximate the nonlinearities with polynomials was chosen. The functions to describe nonlinearities are in the latter called *describing functions*.

The usage of polynomials as *describing functions* has several advantages. First, only the coefficients of the polynomial needs to be stored to fully describe the transfer function. Moreover, with the Horner's scheme [73], there exist an efficient algorithm for polynomial evaluation. Furthermore, the differentiation of a polynomial is straightforward and the derivative is once again a polynomial. Finally, for parameter extraction from simulation or measurement data there exist several well-tested methods for polynomial fitting, e.g., in MATLAB®. For a more general class of describing functions, either the transfer function needs to be hard coded on C++ side which would contradict the standard library approach or a parser for mathematical

expressions must be implemented which would most probably cause a performance loss. The disadvantage of this approach is that a polynomial approximation is only well-suited for weakly nonlinearities and small to medium excitations around the quiescent point, but polynomials are poor global approximators [74]. Since in some cases a polynomial description is unsuitable, e.g., when dealing with *XREAL* signals or if the modelled block is strong nonlinear, a PWL describing function is another approach to model nonlinear behaviour [74]. The accuracy of the fit depends on the number of 'pieces' the function is divided into.

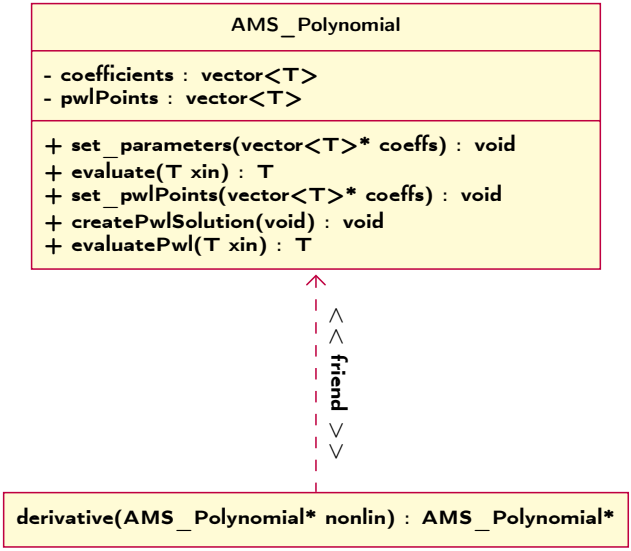


Figure 4.3: AMS_Polynomial class.

```
T evaluate(T xin) const{
T value;
value = static_cast<T>(0);
for (typename vector<T>::const_reverse_iterator iter = this->coefficients.rbegin(); iter != this->coefficients.rend(); iter++){
value = ((*iter) + xin * value);
}
return value;
}
```

Listing 4.2: Evaluation of a polynomial using Horners scheme.

Since nonlinearities appear in more than one block of the standard library, the C++ template class *AMS_Polynomial* was created for handling polynomial nonlinearities. The class contains a *vector* for storing the coefficients, a method to set the coefficients and an evaluation function (see Listing 4.2). For the calculation of a derivative, the *friend* method *derivative* was written, which creates an *AMS_Polynomial* object containing the derivative of the argument. For PWL describing functions, the class was extended by methods to set interpolation points, create a PWL description of the nonlinearity from a polynomial describing function on base of these points and to evaluate the PWL function. Since the creation of the PWL solution is only done once in the initialization step, a high degree polynomial can be chosen to describe the nonlinearity without losing performance and so a good global approximation can be ensured. If needed, the class can also be extended by a method for directly setting the PWL function without the detour via the polynomial description. Fig. 4.3 depicts the class diagram of the *AMS_Polynomial* class.

4.3.1 Multivariate Polynomials

Until now it was implicitly assumed that the nonlinear function is only dependent on one variable but of course there are models which require a nonlinear function description which depends on multiple variables. Therefore, an approach using multivariate polynomials was chosen. A multivariate polynomial with the total degree $\leq m$ is defined as shown in eq. 4.1.

$$P(x_0, x_1, ...x_N) = \sum a_k \cdot \prod_{l=0}^{N} x_l^{\alpha_l} \quad \text{with: } \sum \alpha_l \leq m; \alpha_l \in \mathbb{N}_0 \tag{4.1}$$

As can be seen, with this kind of nonlinearity approximation intermodulation between the different signals can be described since there are terms where the product of the variables $x_0, ..., x_N$ appears.

As for the one-dimensional case, the multivariate polynomial description was implemented in the C++ class *AMS_MultivariatePolynomial* and a PWL description was added to this class.

4.4 Description of LTI systems

When modelling analog or RF ICs, Linear Time-Invariant (LTI) systems are from great importance because many circuits can be described exactly or with sufficient

accuracy by these kinds of systems. As the name implies, in LTI systems there is a linear relationship between input $x(t)$ and output $y(t)$, which means that the system can be described by an ordinary linear differential equation (eq. 4.2). Furthermore, time-invariance means that the coefficients in the differential equation are not time-dependent and therefore constant.

$$a_0 y(t) + a_1 \frac{dy(t)}{dt} + a_2 \frac{d^2 y(t)}{dt^2} + \ldots = b_0 x(t) + b_1 \frac{dx(t)}{dt} + b_2 \frac{d^2 x(t)}{dt^2} + \ldots \tag{4.2}$$

The fundamental result in LTI system theory is that every LTI system can be characterized completely by a so-called system's impulse response. The output $y(t)$ of a system can be calculated by the convolution of the input $x(t)$ and the impulse response $h(t)$ (eq. 4.3).

$$y(t) = x(t) \star h(t) = \int_{-\infty}^{\infty} x(\tau) \cdot h(t - \tau) d\tau \tag{4.3}$$

LTI systems can also be described in the Laplace domain by a transfer function $H(s)$, which is the Laplace transform of $h(t)$. In frequency domain, the output $Y(s)$ is the product of $H(s)$ and the Laplace transform $X(s)$ of the input signal $x(t)$.

The transfer function $H(s)$ can be derived from eq. 4.2 by applying the derivation rules of the Laplace transform and it can be shown that $H(s)$ is a fraction of two polynomials, thus a rational function.

$$\begin{aligned} a_0 y(t) + a_1 \frac{dy(t)}{dt} + a_2 \frac{d^2 y(t)}{dt^2} + \ldots &= b_0 y(t) + b_1 \frac{dx(t)}{dt} + b_2 \frac{d^2 x(t)}{dt^2} + \ldots \\ \Rightarrow a_0 Y(s) + a_1 s Y(s) + a_2 s^2 Y(s) + \ldots &= b_0 X(s) + b_1 s X(s) + b_2 s^2 X(s) + \ldots \\ \Rightarrow H(s) = \frac{Y(s)}{X(s)} &= \frac{b_0 + b_1 s + b_2 s^2 + \ldots}{a_0 + a_1 s + a_2 s^2 + \ldots} \end{aligned} \tag{4.4}$$

There are different notations for transfer functions, but it has been shown that aside from the representation as one fraction of two polynomials for the modelling of LTI systems especially in combination with *XREAL* and spectral signals the Jordan normal form is the most beneficial one.

```
class AMS_TransferFunction : public AMS_Object {
public:

    enum tfRepresentationType {LAPLACE,JORDAN};
    typedef struct {
        complex<double> p;      // pole
        complex<double> g;      // gain
        int m;              // pole multiplicity
    } basicTFcomponent;
    [...]
    void read_from_file(const std::string& coefficient_file, ↩
        ↪ tfRepresentationType tfRepresentationForm);

    dcmplx evaluate(double frequency) const {
        dcmplx value = 0.0;
        if (this->FourierLaplace_isValid){
            value = evaluate_rational_function(this->numerator_f
                                ,this->denominator_f,frequency);
        }
        else{
            for (std::vector<basicTFcomponent>::const_iterator it = ↩
                ↪ this->JordanNormalForm.begin(); it != ↩
                ↪ this->JordanNormalForm.end(); it++) {
                value += it->g / pow(dcmplx(0.0,2*M_PI*frequency) + it->p,it->m);
            }
        }
        return value;
    }
    //calculate derivative of transfer function
    friend AMS_TransferFunction* calculate_derivative(AMS_TransferFunction ↩
        ↪ const * tf);
    [...]

private:

    [...]

    std::vector<dcmplx> numerator_f, denominator_f;
    std::vector<basicTFcomponent> JordanNormalForm;

    bool FourierLaplace_isValid;
};
```

Listing 4.3: Transfer function class in C++.

To get the Jordan normal form $H(s)$ is decomposed into partial fractions and the resulting representation form looks like eq. 4.5

$$H(s) = \frac{g_0}{(s+p_0)^{m_0}} + \frac{g_1}{(s+p_1)^{m_1}} + \frac{g_2}{(s+p_2)^{m_2}} + ..., \tag{4.5}$$

with the poles p_i, the gain of the partial fractions g_i and the pole multiplicity m_i.

Listing 4.3 shows the C++ class which serves as container for transfer function objects. It contains a vector of $< p, g, m >$ triple for a Jordan normal form description of the transfer function and two vectors of complex values, one vector for the numerator and one vector for the denominator, for a description in the standard form. A Boolean variable signals which representation is valid. Furthermore, the class contains methods for importing the transfer function from a *.csv*-file and an evaluation method, which evaluates the transfer function for a given frequency. As shown later, for some operations the derivative of the transfer function with respect to the Laplace variable s is needed. Therefore, the function *calculate_derivative* symbolically derives the transfer function.

4.4.1 LTI system modelling for different signal descriptions

Due to the fact that the signals and impulse responses in analog systems are continuous but computers can deal only with a sampled version of the signals, algorithms are needed which can calculate with a high accuracy a sampled version of the output of the LTI systems using the sampled input signal. For the various signal descriptions, different methods to calculate the output signal y exist and are derived in the following sections.

4.4.1.1 Filter model for baseband equivalent signal description

As shown in section 3.2, the baseband equivalent signals allow the efficient description of modulated signals by passing only the complex amplitude $I + jQ$ and the carrier frequency f_c. When dealing with modulation signals with a small bandwidth compared to the time constants of the system it is sufficient to regard the transfer function as constant in the signal bandwidth. Therefore, the approximated response of the system is the multiplication of the input x with the transfer function H evaluated at f_c (eq. 4.6).

$$y(t) = H(f_c) \cdot x(t). \tag{4.6}$$

For different centre frequencies the weighting factor $H(f_c)$ is different and can be easily calculated with the *evaluate* function of the *AMS_TransferFunction* class. But if

the bandwidth of the signal is in the range of the systems natural frequencies or even higher a constant transfer function approximation is not accurate enough. Fig. 4.4

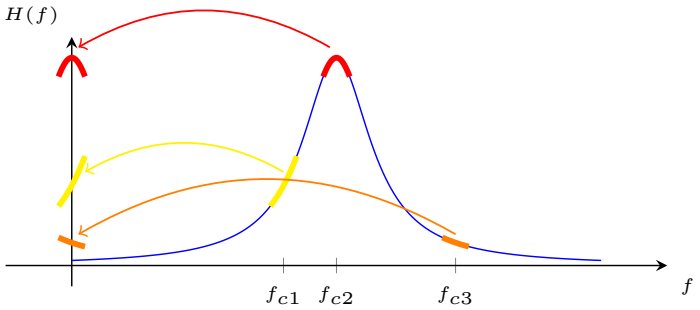


Figure 4.4: Generic bandpass transfer function.

represents this case graphically. It can be seen that the variation of the transfer function in the signal bandwidth cannot be neglected anymore because the deviation of the model from the genuine system is no longer close to zero. For baseband equivalent signals, the filter characteristic around the signal's centre frequency must be mapped to baseband to cover the influence of the LTI system on the signal's complex amplitude $I(t) + jQ(t)$.

In [75] an approach to cover the in-band frequency dependent effects is presented. It is based on an idea formulated in [42] as part of the Envelope Transient Harmonic Balance (ETHB) algorithm. A linear network is approximated for small frequency deviations F by a Taylor series around the carrier frequency f_c of the signal (see eq. 4.7).

$$H(f_c + F) = \underbrace{H(f_c)}_{c_0} + \underbrace{\frac{1}{1!} \cdot \left.\frac{dH(f)}{df}\right|_{f=fc}}_{c_1} \cdot F + \underbrace{\frac{1}{2!} \cdot \left.\frac{d^2H(f)}{df^2}\right|_{f=fc}}_{c_2} \cdot F^2 + \ldots \tag{4.7}$$

The LTI response to a bandwidth-limited input signal around f_c can be formulated in the frequency domain as:

$$\begin{aligned} Y(f) &= H(f) \cdot X(f) = H(f) \cdot X_{eq.BB}(f - f_c) \\ &\text{with: } X(f) = X_{eq.BB}(f - f_c) \end{aligned} \tag{4.8}$$

In eq. 4.8 $X_{eq.BB}$ is the baseband equivalent signal of the input X. Furthermore, for simplicity reasons, $F = f - f_c$ is introduced. The output signal y is calculated in time

domain by the inverse Fourier transformation.

$$\begin{aligned} y(t) &= \mathcal{F}^{-1}\{Y(f)\} = \int_{f_c-BW/2}^{f_c+BW/2} Y(f)\cdot e^{j2\pi ft}df \\ &= \int_{-BW/2}^{BW/2} H(F+f_c)\cdot X_{eq.BB}(F)\cdot e^{j2\pi(f_c+F)t}dF \\ &= \underbrace{e^{j2\pi f_c t}}_{\text{carrier}}\cdot \underbrace{\int_{-BW/2}^{BW/2} H(F+f_c)\cdot X_{eq.BB}(F)\cdot e^{j2\pi Ft}dF}_{\text{baseband equivalent signal}} \end{aligned} \tag{4.9}$$

Now $H(f_c+F)$ in eq. 4.9 can be replaced by the Taylor series from eq. 4.7. Only the calculation of the baseband equivalent output signal will be regarded, since f_c is fixed. $y_{eq.BB}(t)$ can be calculated straightforwardly with the help of the Taylor coefficients c_i of the H_f and the time derivatives of the input signal x.

$$\begin{aligned} y_{eq.BB}(t) &= \int_{-BW/2}^{BW/2} \left(c_0 + c_1F + c_2F^2 + ...\right)\cdot X_{eq.BB}(F)\cdot e^{j2\pi Ft}dF \\ &= c_0\cdot \int_{-BW/2}^{BW/2} X_{eq.BB}(F)\cdot e^{j2\pi Ft}dF \\ &+\frac{c_1}{j2\pi}\cdot \int_{-BW/2}^{BW/2} j2\pi F X_{eq.BB}(F)\cdot e^{j2\pi Ft}dF \\ &+\frac{c_2}{(j2\pi)^2}\cdot \int_{-BW/2}^{BW/2} (j2\pi F)^2 X_{eq.BB}(F)\cdot e^{j2\pi Ft}dF \\ &+... \\ &= c_0 x_{eq.BB}(t) + \frac{c_1}{j2\pi}\frac{dx_{eq.BB}(t)}{dt} + \frac{c_2}{(j2\pi)^2}\frac{d^2x_{eq.BB}(t)}{dt^2} + ... \end{aligned} \tag{4.10}$$

The signal derivate can be approximately calculated using the difference quotient (eq. 4.11). The calculation of higher order derivatives is recursively implemented. The Taylor coefficients are determined using the symbolically derived transfer functions of $H(s)$ and evaluating them for the given centre frequency f_c.

$$\frac{dx(t)}{dt} \approx \frac{x(t)-x(t-\Delta t)}{\Delta t} \tag{4.11}$$

The presented filter description algorithm does not depend on a predetermined sampling rate and is therefore flexibly applicable in different simulation setups. Only

if the derivatives with a higher order than 1 are used, an equidistant sampling is required. For smooth transfer functions which vary only slowly over frequency, this method is sufficient to compute the output y of an LTI systems. The drawback of the algorithm is that it becomes inaccurate when the transfer function is not smooth enough to be described by a Taylor polynomial. Also, for high modulation bandwidths, the method fails because a polynomial is only a local approximation. Therefore, a second algorithm was developed with better approximation properties.

As can be seen from Fig. 4.4, for the description of the influence of the LTI system on the complex amplitude $I(t) + jQ(t)$, the transfer function needs to be shifted from the baseband equivalent signal's centre frequency f_c to baseband. This operation can be efficiently implemented when the system's transfer function is in the Jordan Normal form.

$$\begin{aligned} H_{BB.eq.}(s) &= H(s + j2\pi f_c) \\ \Rightarrow H_{BB.eq.}(s) &= \underbrace{\frac{g_0}{(s + p_0 + j2\pi f_c)^{m_0}}}_{H_{p,0}(s)} + \underbrace{\frac{g_1}{(s + p_1 + j2\pi f_c)^{m_1}}}_{H_{p,1}(s)} + \underbrace{\frac{g_2}{(s + p_2 + j2\pi f_c)^{m_2}}}_{H_{p,2}(s)} + \ldots \end{aligned} \tag{4.12}$$

Eq. 4.12 shows that to get the transfer function $H_{BB.eq.}(s)$ for the baseband equivalent signal, only the poles of each partial fraction have to be shifted by $j2\pi f_c$ which can be straightforwardly implemented.

The output $y(t)$ is calculated by summing up the output signals from each partial filter $H_{p,i}(s)$. Since the output signals $y_{p,i}$ are determined by the multiplication of the input signal with the terms $g_i \cdot (\frac{1}{(s+p_i+j2\pi f_c)})^{m_i}$, an explicit rule for evaluation in the time domain can be derived. If the factor m_i is one, the calculation is reduced to the convolution of the input signal with a complex exponential $\exp(-\tilde{p_i}t)$ where $\tilde{p_i} = p_i + j2\pi f_c$. This convolution product can be numerically computed (eq. 4.13) using the trapezoidal integration rule [44].

$$
\begin{aligned}
y_{p,i}(t) &= x_{BB.eq.}(t) \star \exp(-\tilde{p}_i t) \\
\Rightarrow y_{p,i}(t_n) &\approx k \cdot (x_{BB.eq.}(t_n) + x_{BB.eq.}(t_{n-1})) + p_z \cdot y_{p,i}(t_{n-1}) \\
\text{with: } k &= \frac{1}{2.0/(t_n - t_{n-1}) - \tilde{p}_i} \\
\text{and: } p_z &= \frac{1.0 + \tilde{p}_i \cdot 0.5 \cdot (t_n - t_{n-1})}{1.0 - \tilde{p}_i \cdot 0.5 \cdot (t_n - t_{n-1})}
\end{aligned}
\tag{4.13}
$$

If the multiplicity m_i is bigger than 1 the output $y_{p,i}$ is calculated by a repetitive convolution with the complex exponential. As the first algorithm, this method does not rely on a predetermined sampling rate. Moreover, since only the trapezoidal integration rule is used, the filter model also works with non-equidistant sampling rates.

In Listing 4.4 the C++ class which implements the numeric convolution of an arbitrary input signal with a transfer function of the form $\frac{gain}{s+pole}$ is shown. A pole multiplicity m greater than one can be achieved by cascading objects of the *AMS_NumericLossyIntegrator* class m times.

In Fig. 4.5 both filter modelling methods are compared using models of a bandpass filter with the centre frequency 10 MHz and a quality factor $Q = 100$. The sampling rate for all signals is 100 MHz. One model was created using the Taylor series approximation method (method 1) while the other model uses the integration method (method 2). As input signal, a sinusoidal signal with the frequency of 9.4 MHz was chosen. To show the difference in accuracy of both modelling methods, the signals were once represented by a baseband equivalent signal with the carrier frequency $f_{center} = 9.5$ MHz and a rotating phasor $I(t) + jQ(t)$ with the frequency −100 kHz and once by a baseband equivalent signal with the carrier frequency $f_{center} = 9.75$ MHz and a rotating phasor $I(t) + jQ(t)$ with the frequency −350 kHz. Both are representations of the same signal and the models should ideally output same signals.

For small deviations of 100 kHz from the carrier frequency, the output signals from the models using method 2 and method 1 with a approximation degree of 2 matches the ideal output signal very well. Using method 1 with an approximation degree of 0 (constant transfer function with $H_{model}(f) = H(f_{carrier})$) leads to a visible error because the transfer function varies over the BW of 100 kHz so much, that the assumption of a constant transfer function is not valid. When the deviation gets larger (350 kHz), it can be seen that the ideal output and the output of the model using modelling method 2 still matches very well but the errors of the models

```cpp
//using the the trapezoidal rule
template <class T>
class AMS_NumericLossyIntegrator: public AMS_Object {
public:
    [...]
    T evaluate (T x, double sv_time){
        T y,k,pz;
        double delta_t = sv_time - this->t_last;
        //calculate parameters k and pz
        k = this->gain / (static_cast<T>(2.0)/static_cast<T>(delta_t) - this->pole);
        pz = (static_cast<T>(1.0) + (this->pole)*static_cast<T>(0.5*delta_t)) / (static_cast<T>(1.0) - (this->pole)*static_cast<T>(0.5*delta_t));
        //calculate the output of the lossy integrator
        y = k*(x+this->x_last) + pz*this->y_last;
        //overwrite old values
        this->y_last = y;
        this->x_last = x;
        this->t_last = sv_time;

        return y;
    }

private:
    //gain and pole of lossy integrator
    T gain;
    T pole;
    //old values of input/output signal and time
    T x_last;
    T y_last;
    double t_last;
};
```

Listing 4.4: Integrator in C++.

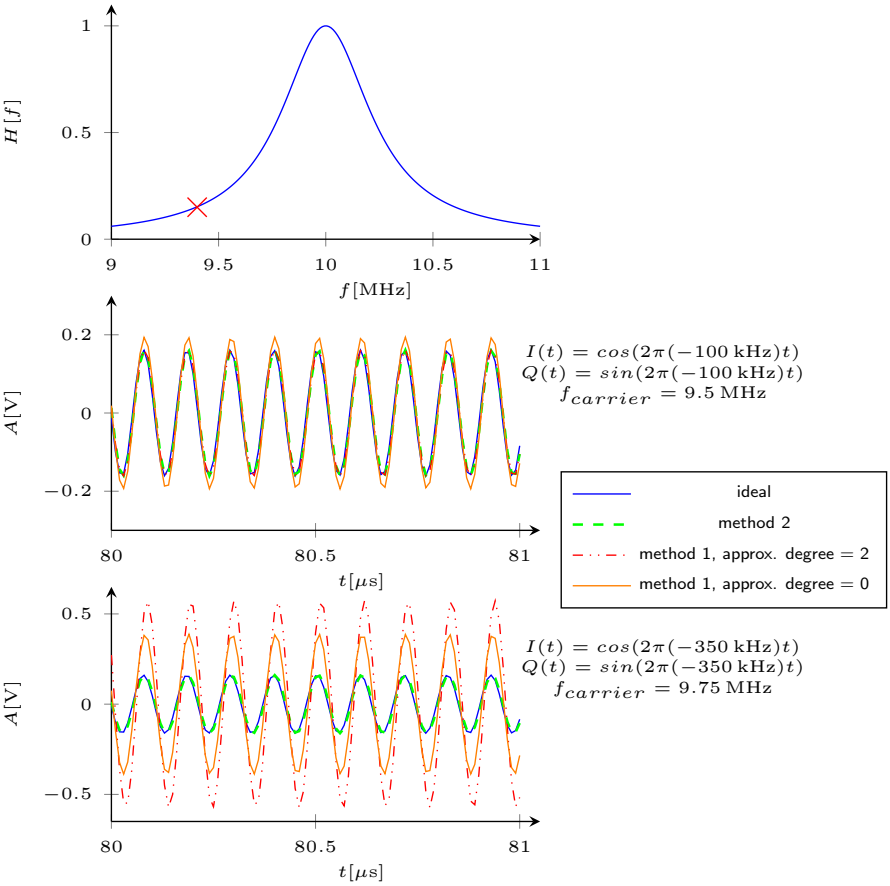


Figure 4.5: Comparison of filter modelling methods.

using method 1 are increasing. What is also noticeable is that method 1 with an approximation degree of 2 delivers worse results than using an approximation degree of 0. That is because the Taylor series is only a good local approximator, but the global behaviour is completely different to that of the real transfer function.

The expressiveness of this academic example can be extended to general bandpass signals which covers a certain BW. In conclusion it can be said that modelling method 1 is sufficient accurate when dealing with low signal BWs compared to the BW of the linear system but method 2 is accurate under every condition. Thus, this method is favoured over method 1 especially since it does not bear crucial disadvantages

compared to method 1.

4.4.1.2 Spectral signal description

A spectral signal can be regarded as sum of baseband equivalent signals with different centre frequencies $f_{c,i}$. To calculate the output signal of a LTI system, each spectral component can be treated as baseband equivalent signal passing the system undisturbed by the other components due to the system's linearity. On implementation level, for each spectral component a baseband equivalent linear operator is instantiated. The spectral input signal is split into the baseband equivalent signals which pass their linear operator. Afterwards, the baseband equivalent output signals are packed together to the spectral output signal. Fig. 4.6 shows the class structure for the implemented spectral and baseband equivalent LTI modelling.

4.4.1.3 *XREAL* signal description

One big advantage of the *XREAL* description is the possibility to calculate the convolution of two signals algebraically without using numeric integration methods. Furthermore, there is no need to introduce finer time steps and additional events because the resulting signals do not rely on time-step integration and approximation errors introduced by numerical methods are avoided.

Like for the LTI modelling in combination with baseband equivalent signals, the Jordan Normal form of $H(s)$ is needed. The output $y(t)$ can be calculated by summing up the output signals from each partial fraction $H_{p,i}(s)$ due to the linearity of the convolution.

The output for a filter of the form $H_{p,i}(s) = \frac{g}{(s-p)^m}$ and an input signal $X(s)$ is calculated in eq. 4.14.

$$Y(s) = \underbrace{X(s) \cdot \frac{g}{(s-p)^m}}_{\text{response to input}} + \underbrace{\sum_{k=0}^{m} \sum_{j=0}^{m-k-1} \binom{m}{k} (-1)^k p^k y_0^{(j)} \frac{s^{m-k-j-1}}{(s-p)^m}}_{\text{response to initial conditions}}, \tag{4.14}$$

with the initial output signal and its derivatives $y_0^{(j)}$.

To calculate the output, which fits in the *XREAL* description, eq. 4.14 has to be split into *XREAL* compatible waveform sequences again. While $X(s)$ is in the form

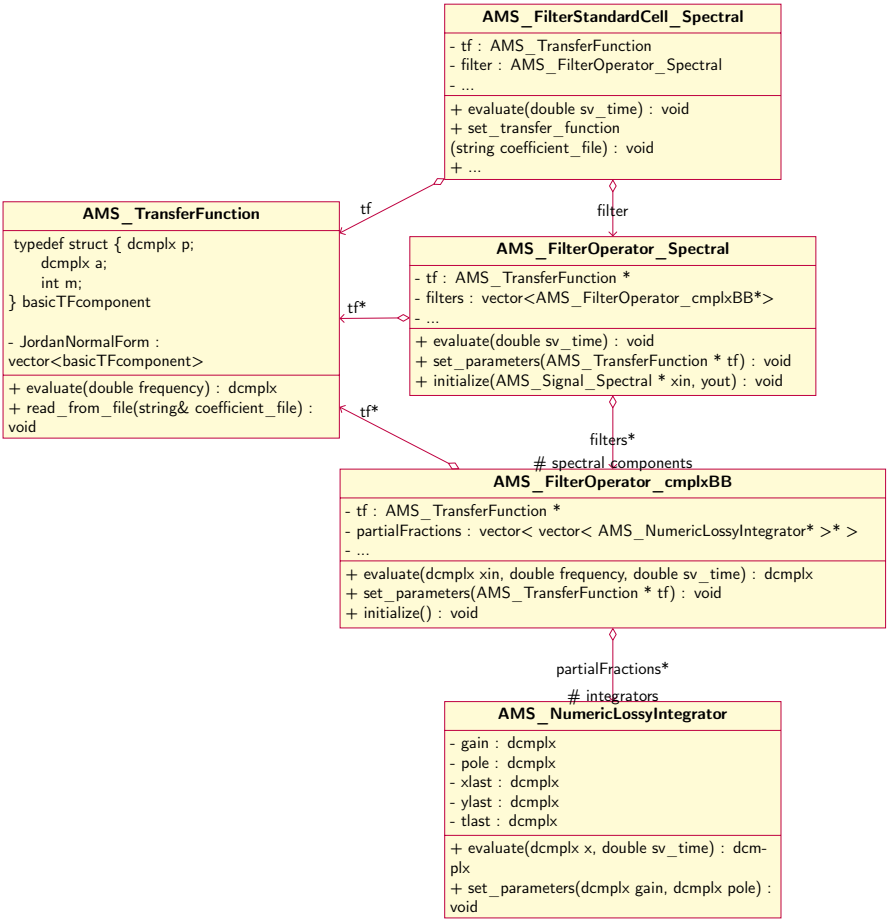


Figure 4.6: Filter Modelling.

$\sum_{i=0}^{K} \frac{b_i}{(s-a_i)^{n_i}}$, it is sufficient to regard the special case $K = 0$ for a single basic waveform component. The other cases can be derived from that case by the superposition principle. Different cases have to be regarded when splitting the product of two basic partial fractions. If the poles of the partial fractions are equal, they can be simply merged together and the pole multiplicities add up. Thus, in the following only cases with parameter sets with different poles are looked at. The simplest case is shown in eq. 4.15 where two parameter sets with pole multiplicities of 1 are convolved. The product can be rewritten as sum of basic waveform components which fit the

XREAL description.

$$\frac{b_1}{s-a_1}\cdot\frac{b_2}{s-a_2}=\frac{\frac{b_1b_2}{a_1-a_2}}{s-a_1}+\frac{\frac{b_1b_2}{a_2-a_1}}{s-a_2}$$
$$\Updownarrow$$
$$<[b_1,a_1,1,t_{0,1}]>\cdot<[b_2,a_2,1,0]>=<[\frac{b_1b_2}{a_1-a_2},a_1,1,t_{0,1}],[\frac{b_1b_2}{a_2-a_1},a_2,1,t_{0,1}]> \tag{4.15}$$

In the next case the product of one parameter set with a pole multiplicity bigger than 1 and one with the pole multiplicity of 1 is calculated (eq. 4.16).

$$\begin{aligned}\frac{b_1}{(s-a_1)^{m_1}}\cdot\frac{b_2}{s-a_2}&=\frac{b_1b_2}{(s-a_1)^{m_1-1}}\cdot\left(\frac{\frac{1}{a_1-a_2}}{s-a_1}+\frac{\frac{1}{a_2-a_1}}{s-a_2}\right)\\&=\frac{\frac{b_1b_2}{a_1-a_2}}{(s-a_1)^{m_1}}+\frac{\frac{b_1}{a_2-a_1}}{(s-a_1)^{m_1-1}}\cdot\frac{b_2}{s-a_2}\end{aligned} \tag{4.16}$$

The first term on the right side of eq. 4.16 is already in an *XREAL* compatible form and the second term resembles the left equation side, but the pole multiplicity of the first partial fraction is reduced by one. Thus, the splitting of the parameter sets for this case can be recursively repeated until all parts are either in an *XREAL* compatible form or in the form of eq. 4.15. The last case is the case where two parameter sets with pole multiplicities bigger than 1 are convolved (eq. 4.17).

$$\begin{aligned}&\frac{b_1}{(s-a_1)^{m_1}}\cdot\frac{b_2}{(s-a_2)^{m_2}}\\=&\frac{\frac{b_1}{a_1-a_2}}{(s-a_1)^{m_1}}\cdot\frac{b_2}{(s-a_2)^{m_2-1}}+\frac{\frac{b_2}{a_2-a_1}}{(s-a_2)^{m_2}}\cdot\frac{b_1}{(s-a_1)^{m_1-1}}\end{aligned} \tag{4.17}$$

As for the second case, by recursion the original term can be transferred into a form in which only *XREAL* compatible partial fractions occur. The term $\frac{s^{m-k-j-1}}{(s-p)^m}$ in eq. 4.14 can be best solved in time instead of the Laplace domain. In time domain it is the $(m-k-j-1)$-th derivative of the expression $\frac{1}{(m-1)!}\cdot t^m\cdot\exp(pt)$. The resulting terms can be written as *XREAL* signal. The above derived operations for filtering an *XREAL* signal were implemented in the C++ class *AMS_LaplaceFilter* for the usage in LTI models.

4.5 Modelling of nonlinear systems with memory

One disadvantage of memoryless nonlinearities is -as the name says- that they lack the ability to describe the system's output with regard to events that happened in the past [42]. Therefore, they are not suited to describe circuits which are at the same time nonlinear and have a frequency-dependent transfer characteristic. For some components, the approximation with a memoryless or static nonlinearity is good enough if the frequency response can be regarded as constant over the bandwidth of interest but there are also effects which cannot be explained by a simple static nonlinearity or a linear frequency dependent model. An example is shown in Fig. 4.7.

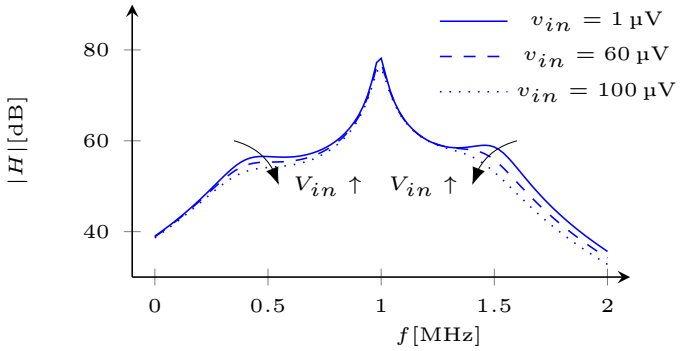


Figure 4.7: Transfer characteristic of the AixRF ΔΣ-ADC's loop filter for different input levels.

It can be seen that with increasing input voltage v_{in} the transfer function of the ADC loop filter changes its shape. This behaviour cannot be modelled using a linear system and also it cannot be explained by a static nonlinearity.

One approach to describe these effects is an approximation of the behaviour using the Volterra series [42]. A Volterra series can be imagined as a Taylor series with memory. In continuous time system, it is defined as follows [42]:

$$\begin{aligned} y(t) &= \sum_{n=1}^{\infty} y_n(t) \\ \text{with: } y_n(t) &= \int_{-\infty}^{\infty} \dots \int_{-\infty}^{\infty} h_n(\tau_1, \dots, \tau_n) x(t-\tau_1) \cdot \dots \cdot x(t-\tau_n) d\tau_1 \dots d\tau_n \end{aligned} \tag{4.18}$$

The expressions $h_n(\tau_1, ..., \tau_n)$ in eq. 4.18 are called n-th order Volterra kernels. For $n = 1$ it corresponds to the impulse response of the linear system. A drawback of the Volterra approach is the large number of parameters to describe a system [76]. Furthermore, Volterra series are ill-suited for large excitations [77] since in terms of nonlinear behaviour they are similar to the static polynomial description.

Another approach to describe nonlinear systems with memory are so-called block-orientated models in which the systems are modelled by connecting static nonlinear blocks and LTI blocks. The simplest ones are Wiener (LN-models) models where a LTI block (L) is cascaded with a memoryless nonlinear block (N) and Hammerstein models (NL-models) where a memoryless nonlinear block is followed by an LTI block [78]. For describing a wider range of nonlinear systems, the cascading and parallel connection of multiple static nonlinear and LTI blocks can be applied. At least for time-discrete systems it can be shown that any finite-dimensional nonlinear system can be approximated arbitrarily well by a finite number of parallel Wiener–Hammerstein models or by cascade Wiener models in parallel [76]. Also, it was proven that a large class of continuous-time systems can be represented exactly by a finite sum of NLNL cascades [79]. Therefore, [76] states that block-orientated models are more suited to describe general nonlinear systems. Another reason to use rather block-orientated models than Volterra models is that they can be built-up from basic cells from the analog standard library and just needs to be connected on SV side which fits into the proposed modelling process.

In addition to the Volterra and the block-orientated models, there exists several other approaches like describing a system with neural networks or black box approaches. The main drawback of these descriptions is, that they do not allow any insight and rely only on the input of measurement data. Since one cannot be sure if the measurement data covers the complete behaviour of the system, these approaches can lead to badly identified parameters or behaviour that is not covered by the model [80]. Therefore, they are not further covered in this work.

As the main focus lies on modelling RF systems, a block-orientated structure which suits the special characteristics of such systems best has to be chosen. Since for most RF systems a frequency domain view is desirable, the frequency domain behaviour of the system should be modelled accurately. Furthermore, small-signal properties should be preserved because often RF signals have low amplitudes. The choice fell on a structure shown in Fig. 4.8, so-called S_M-type models [81]. These models consist of M parallel path. Each path has a sandwich structure where a static nonlinear block (N) is located between two LTI blocks (L). The nonlinearity in the n-th path is the monomial x^n. The reasons for choosing this structure are listed in the following:

- The model is easy to interpret because the nonlinearities consist only of monomials.

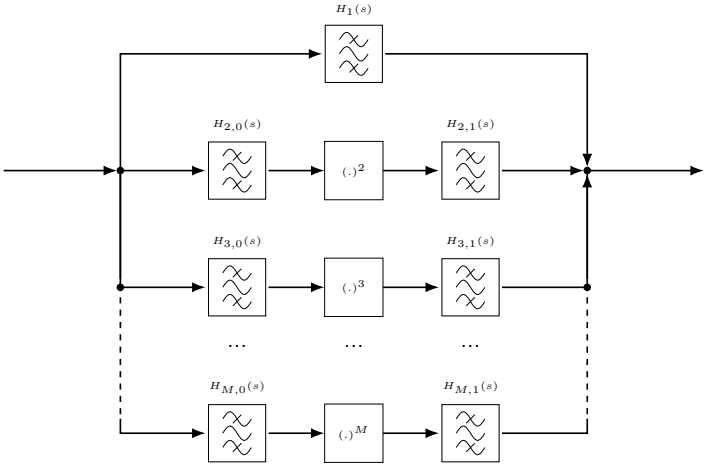


Figure 4.8: Structure of a S_M-type system.

- Small-signal properties are preserved. When dealing with small signals, all path except the linear one, which equals the small signal transfer function, can be neglected.

- A change of frequency response shape as in Fig. 4.7 is possible (example in Fig. 4.9) which is not guaranteed by all block-orientated models. A Hammerstein (NL) model for example cannot produce a change in the frequency response. Since the static nonlinearity comes first, the amplitude of the nonlinear block's output signal does depend on the input level but not on the input frequency. The following LTI is responsible for the frequency dependency but is independent from the input amplitude. Therefore, the shape of the transfer characteristic is fixed is only shifted along the y-axis depending on the input level.

- The parameter identification can be done from a steady-state frequency response [81] and is easy to implement.

[2]The example system consists of the linear path with the transfer function $H_1(s) = \frac{1}{1+1e-7\cdot s}$ and the 3rd order path with the transfer functions $H_{3,0}(s) = \frac{1}{1+1e-8\cdot s}$ and $H_{3,1}(s) = \frac{1e-7\cdot s}{1+1e-7\cdot s}$. The transfer characteristic of the S_M system was extracted once for an input amplitude of 10 mV and 1 V, respectively.

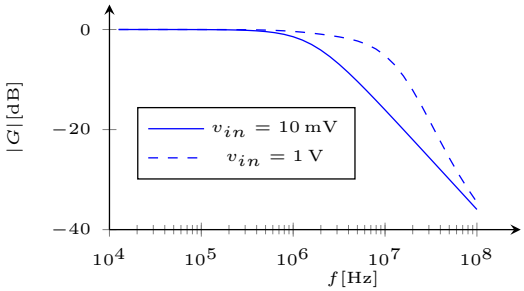


Figure 4.9: Frequency response for different input level.

4.6 Influence of supply and substrate noise

Crosstalk from digital aggressor circuits to sensitive analog circuits is a major concern in large mixed-signal SoCs [23] [24]. There are several mechanisms that create substrate or supply noise. $\frac{dI}{dt}$-noise arises when digital circuits are switching. Sharp rising currents are drawn from the supply. A voltage drop results over the bond wire inductivities which depends on the size of the inductivity and the derivative of the current [82] (eq. 4.19).

$$v_n(t) = L \cdot \frac{dI(t)}{dt} \tag{4.19}$$

This means that the faster the current rises or falls the higher the peak Power Supply Noise (PSN) voltage is. Other mechanisms are impact ionization currents from hot electrons creating a current from drain to the substrate, capacitive coupling into the substrate, gate induced drain leakage, photon-induced currents and diode leakage currents [83].

The noise couples through the substrate or via the supply lines to the sensitive analog blocks and degrades the performance of the system. The substrate coupling can be modelled using lumped R or RC circuits [84] [85] or using transfer functions [86].

The focus of this section lies on the model of the analog or RF circuit. PSN and substrate noise can have different impacts on sensitive circuits. In [68] the influence on the delay and the slope of digital cells is examined. Functions which describe the timing in dependency of the supply voltage are used to calculate the corresponding values. Meier et al. [87] extended the methodology to building blocks of a PLL like a Phase-Frequency Detector (PFD), a FD and a VCO. The effect of supply ripple on circuits in an ΔΣ-ADC are treated in [88].

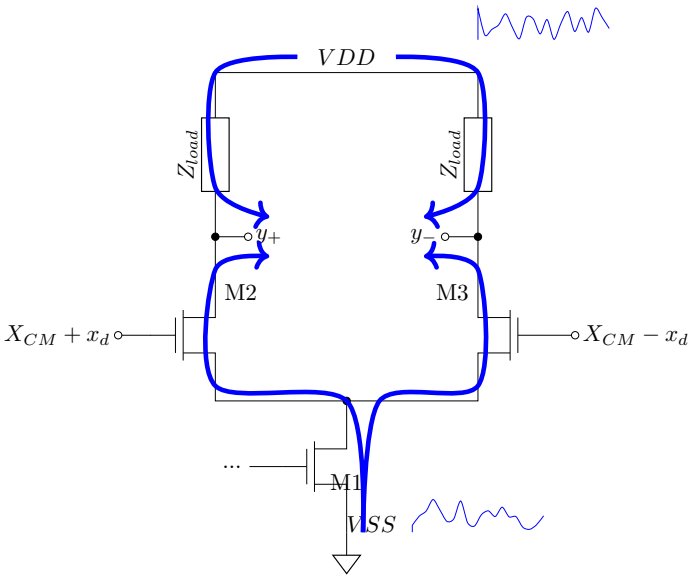


Figure 4.10: PSN in differential structures.

$$
\begin{aligned}
y_+ &= F(X_{cm} + x_d, B + x_b) \quad \wedge \quad y_- = F(X_{cm} - x_d, B + x_b) \\
y_+ &= F(X_{cm}, B) + \frac{\partial F}{\partial x}\Big|_{x=X_{cm}} \cdot x_d + \frac{\partial F}{\partial b}\Big|_{b=B} \cdot x_b \\
&\quad + 0.5\frac{\partial^2 F}{\partial x^2}\Big|_{x=X_{cm}} \cdot x_d^2 + 0.5\frac{\partial^2 F}{\partial b^2}\Big|_{b=B} \cdot x_b^2 + \frac{\partial^2 F}{\partial x \partial b}\Big|_{x=X_{cm}} \cdot x_d x_b + \ldots \\
y_- &= F(X_{cm}, B) + \frac{\partial F}{\partial x}\Big|_{x=X_{cm}} \cdot (-x_d) + \frac{\partial F}{\partial b}\Big|_{b=B} \cdot x_b \\
&\quad + 0.5\frac{\partial^2 F}{\partial x^2}\Big|_{x=X_{cm}} \cdot (-x_d)^2 + 0.5\frac{\partial^2 F}{\partial b^2}\Big|_{b=B} \cdot x_b^2 + \frac{\partial^2 F}{\partial x \partial b}\Big|_{x=X_{cm}} \cdot (-x_d) x_b + \ldots \\
\Rightarrow y &= y_+ - y_- = 2\frac{\partial F}{\partial x}\Big|_{x=X_{cm}} \cdot x_d + 2\frac{\partial^2 F}{\partial x \partial b}\Big|_{x=X_{cm}} \cdot x_d x_b + \ldots
\end{aligned}
\tag{4.20}
$$

A different approach can be used for blocks in the analog frontend in receiver chains. Since most circuits in integrated RF receivers are built using differential signalling a differential LNA is taken as example. Theoretical considerations and measurements validating the theory on the example of a differential LNA used in a CMOS Global Positioning System (GPS) receiver were done in [89]. For perfectly

matched differential branches, in the first approximation the Power Supply Rejection Ratio (PSRR) is infinite because the paths from the positive supply node respectively the negative supply node to the differential output nodes are equal. Therefore, the noise is cancelled at the output when subtracting the negative output signal from the positive one. The effects of supply noise in differential structures can only be seen if an asymmetry is assumed e.g., due to mismatch or if a second order approximation is used. When neglecting frequency dependent effects, the transfer functions from the inputs to the output y_+ resp. y_- of amplifier in Fig. 4.10 can be described by the function $F(x, b)$ which depends on the input signal x and the supply signals b. The input signals x are composed of a common mode part X_{CM} and the differential signal x_d. The supply signals consist of the static part B and the supply noise x_b. The output signals y_+ and y_- and the resulting signal y can be approximated with a Taylor series around the common-mode input signal X_{CM} and the nominal supply B as shown in eq. 4.20.

It can be seen that for a second order approximation apart from the influence of the differential input signal x_d there is a dependency from the supply noise x_b which modulates the signal x_d. This means that the noise at the output of the differential circuit depends on the level of the input signal and of course its own level. Furthermore, because of the multiplication of the supply noise with the input signal, there will be a frequency translation between the supply noise and the noise seen at the output. This effect can be quite harmful for some receive scenarios. Using the example of a large SoC where the digital part is for example clocked with $f_{clk} = 32\,\text{MHz}$ the arising problems will be explained. Due to the switching of the digital circuit, there will be PSN around f_{clk}. A large blocker at 32 MHz distance from the wanted receive signal will mix the supply noise in the receive band and therefore degrade the performance of the system.

When extending the Taylor series to higher degrees, also harmonics of the input signal's frequency, harmonics of the supply signal's frequency and more intermodulation products between supply noise and input signal can be found at the output.

Modelling analog (differential) blocks including PSN can be perfectly done by using multivariate polynomials presented in sec. 4.3.1. In chapter 5 a method to extract the transfer function from schematic simulations will be shown.

4.7 Modelling domain

As mentioned in chapter 3 it can be beneficial to describe a system in the domain which is most meaningful for this special component. As example a VCO is re-

garded.

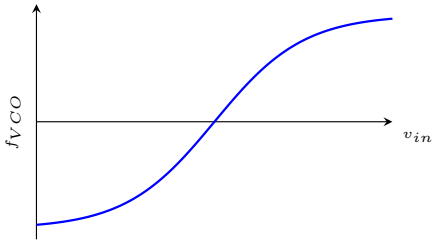


Figure 4.11: Typical VCO transfer function.

In the voltage domain the circuit is described by a nonlinear differential equation but in the phase domain the systems can be modelled with low effort. The relation between VCO input signals and its output frequency can be described as $f_{VCO} = g(x_{in,1}, x_{in,2}, ..., x_{in,n})$ which is an algebraic instead of a nonlinear differential equation. The phase of the oscillator can be derived from the frequency by integration. Since the oscillator's amplitude can be considered constant for the relevant system level simulations, modelling the frequency (phase) of the oscillator is sufficient. Moreover, in typical scenarios the VCO's input signals vary much slower than the oscillator's output signal. Therefore, the number of events needed to sample the VCO's output frequency (phase) is much smaller than the number of events needed to sample the VCO's output voltage. Hence for performant system level simulations, it is beneficial to model the frequency (phase) of oscillators and not their output voltages.

It is not sensible to model the oscillator in the phase domain to spare events if the connected blocks directly transform the VCO signal back to the voltage domain where much more events are needed. Therefore, the signal description should stay in the phase domain as long as possible. Luckily, for parts of systems with a constant envelope, the modelling is unproblematic. Some blocks are even better describable in the phase domain since they are designed to be linear in this domain but are strongly nonlinear in the voltage domain. A FD for example just divides the input frequency (phase) by the divider factor, an upconversion mixer adds the input frequencies/phases and a buffer simply passes the input to the output. To get even more precise models, linear transfer functions can be extracted in the frequency/phase domain directly from a circuit level simulation [55] which helps characterizing second order effects like delay in the circuits or other effects influencing the phase.

At some points in the system, it might be necessary to switch from frequency/phase domain to the voltage domain. For example, in a PLL to cover sampling effects

coming from the reference clock it is necessary to model the PFD and the Charge Pump (CP) in the voltage domain. Therefore, the output signal of the FD in the feedback path of the PLL should be transformed in this domain. Since the FD's output signal is binary, only the rising and falling edges are important. These transitions can be calculated from the FD's output phase by determine the time points where the phase crosses the corresponding value, e.g., if the duty-cycle of the output signal is 0.5 at every phase increment of π the output signal changes between '0' and '1'. Depending on the signal description used, the crossing points can be calculated explicitly or iteratively as proposed in sec. 3.4.1. Since the phase in a PLL is monotonic rising, Newton's method can be safely used. Fig. 4.12 graphically explains how the transformation from the frequency/phase domain to the voltage domain works. The mixed phase domain/voltage domain concept was applied to model the PLL in the AixRF transmitter with a high accuracy and high simulation performance [90].

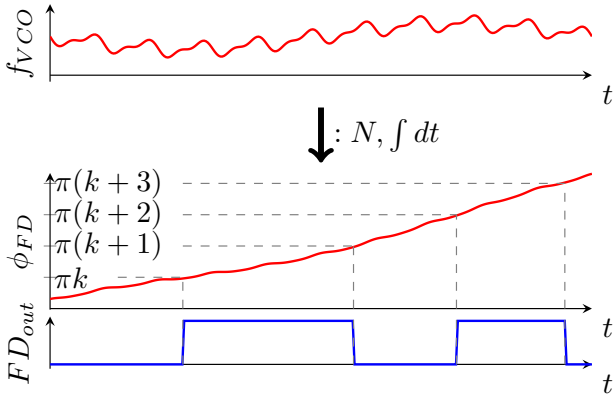


Figure 4.12: Concept for modelling the FD.

For some circuit models it is beneficial to directly calculate a spectral signal description in the voltage domain from a phase signal. Some properties from a typical clock signal can be considered to efficiently translate the signal description into each other. It is assumed that the signal has a nearly constant frequency $\omega_{sig}(t)$ over a specific interval and that the phase deviation from the ideal phase $\phi_{ideal} = \omega_{ideal} \cdot t$ is small. The signal in the voltage domain can then be calculated with the known amplitude

A_{sig} to:

$$\begin{aligned} v_{sig}(t) &= A_{sig} \cdot cos(\phi_{sig}) = A_{sig} \cdot cos\left(\int_{-\infty}^{t} \omega_{sig}(\tau) d\tau \right) \\ &= A_{sig} \cdot cos(\omega_{ideal} \cdot t + \phi_{error}) \\ &\text{with:} \quad \phi_{error} = \phi_{sig} - \phi_{ideal} \end{aligned} \tag{4.21}$$

Since the deviation ϕ_{error} from the ideal phase is small, eq. 4.21 can be rewritten and approximated to

$$\begin{aligned} v_{sig}(t) &= A_{sig} \cdot cos(\omega_{ideal} \cdot t + \phi_{error}) \\ &= A_{sig} \cdot cos(\omega_{ideal} \cdot t) \cdot \underbrace{cos(\phi_{error})}_{\approx 1} - A_{sig} \cdot sin(\omega_{ideal} \cdot t) \cdot \underbrace{sin(\phi_{error})}_{\approx \phi_{error}} \\ &\approx A_{sig} \cdot cos(\omega_{ideal} \cdot t) - A_{sig} \cdot sin(\omega_{ideal} \cdot t) \cdot \phi_{error} \\ \Rightarrow V_{sig}(\omega) &\approx \pm \frac{A_{sig}}{2} \delta(\omega \pm \omega_{ideal}) + \mp \frac{j A_{sig}}{2} \Phi_{error}(\omega \pm \omega_{ideal}) \end{aligned} \tag{4.22}$$

The spectrum therefore consists of Dirac pulses at $\pm\omega_{ideal}$ and the spectrum of the phase error folded to $\pm\omega_{ideal}$. $\Phi_{error}(\omega)$ can either be calculated by a FFT or when dealing with *XREAL* signals it can be computed as shown in sec. 3.6. As shown, the spectral representation in the voltage domain of a clock signal can be calculated with low computational effort from its phase domain description. This is especially useful at domain crossings like in the downconversion mixer in the receive part where the RF and BB signals are modelled as spectral voltage signal and the Local Oscillator (LO) signal in the phase domain.

As mentioned in sec. 3.4 it has been shown that *XREAL* signals are well-suited for circuits which are nearly linear and thus they are superior to other signal description forms when it comes to the modelling of clock generating and distribution circuits because in contrast to spectral signals the transient start-up behaviour is covered and the number of events can be reduced to a minimum.

CHAPTER 5

PARAMETER EXTRACTION

A model-based simulation is only as good as the models are. Therefore, suitable methods are needed to extract parameters for the models. This section deals with the topic how to find proper values for the parameterized models developed in the last chapter.

There exist different approaches to get a valid model. The white-box approach transfers a-priori knowledge about a system's structure directly into differential equations. The advantage is that the resulting model is highly interpretable and always delivers physically sensible results because the underlying physical laws are directly implemented in the model. The disadvantage of this approach is that for complex systems the effort to correctly model the physics behind the systems increases rapidly [80]. Therefore, the white-box approach is not suitable for most circuits in RF systems.

The complete opposite of a white-box model is the black-box approach where no structural knowledge is used but only measurement data. The parameter of a general model, e.g., a neural network, are optimized so that they fit to the measured data. A black-box model gives no inside in the modelled system and therefore is not interpretable [80]. The problems with black-box models are that the measured data must cover all possible situations otherwise not all effects of the underlying real system are covered which can lead to wrong simulation results.

A good compromise are grey-box models. In the grey-box approach knowledge about the modelled circuit is exploited, e.g., whether the block is linear or nonlinear and which type of circuit is modelled (filter/amplifier/VCO). Measurement or simulation data is then used to tune some parameters of the equations describing the system.

5.1 Parameter Extraction of Linear Systems

An analog linear system consisting of lumped elements can be described by a rational function

$$H(s) = \frac{b_0 + b_1 s + b_2 s^2 + ... + b_l s^l}{a_0 + a_1 s + a_2 s^2 + ... + a_m s^m}. \tag{5.1}$$

The goal of the parameter extraction is to find values b_i and a_i so that the modelled transfer function fits to the sampled values from measuring or simulating the linear system. In case of creating a macro model from an existing circuit on transistor-level the easiest way is to run an AC simulation in a matrix-based circuit simulator and use the frequency-domain output results, e.g., the amplitude and phase response of the block for parameter extraction. In [91] the vector fitting method for finding optimal parameters of $H(s)$ is proposed. For simulated frequencies $\omega_0, \omega_1, ..., \omega_n$ and their corresponding transfer function values $H(\omega_0), H(\omega_1), ..., H(\omega_n)$ the unknown a_i and b_i of the overdetermined linear equation system 5.2 are estimated.

$$\begin{aligned}
\left(a_0 + a_1 j\omega_0 + ... + a_m (j\omega_0)^m\right) \cdot H(\omega_0) &= b_0 + b_1 j\omega_0 + ... + b_l (j\omega_0)^l \\
\left(a_0 + a_1 j\omega_1 + ... + a_m (j\omega_1)^m\right) \cdot H(\omega_1) &= b_0 + b_1 j\omega_1 + ... + b_l (j\omega_1)^l \\
&... \\
\left(a_0 + a_1 j\omega_n + ... + a_m (j\omega_n)^m\right) \cdot H(\omega_n) &= b_0 + b_1 j\omega_n + ... + b_l (j\omega_n)^l
\end{aligned} \tag{5.2}$$

Based on [91] and further works the *tfest* algorithm was proposed in [92] and implemented in MATLAB®. The routine allows an accurate out-of-the-box usable transfer function parameter fitting. Similar MATLAB® functions like the *sset* algorithm which gives a state space representation of the linear system achieves even more accurate results when it comes to transfer functions with narrow spikes or complex valued systems like a Polyphase Filter (PPF). As explained in chapter 4, for filter modelling with spectral as well as with *XREAL* signals it is beneficial to have the transfer function in the Jordan normal form as sum of partial fractions. Some of MATLAB®'s fitting algorithms directly delivers this form and otherwise, a partial fraction decomposition is required after fitting.

A MATLAB® GUI (Fig. 5.1) was implemented for a user-friendly application of the fitting algorithms.

The MATLAB® tool imports the simulated waveforms and automatically decides on basis of the symmetry of the waveforms whether it is a real or a complex system. In the next step, the transfer function is fit with a suitable MATLAB® algorithm. A tolerance and a number of system states can be specified by the user. The fitted

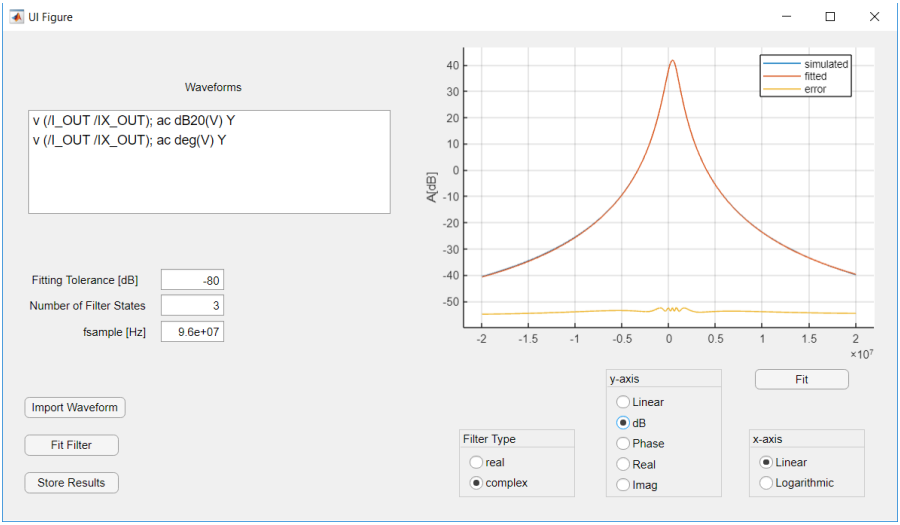


Figure 5.1: MATLAB® GUI for parameter extraction of linear systems.

transfer function is plotted together with the original data plus the fitting error and the view can be adjusted to different settings ('linear', 'dB', 'phase', ...).

5.2 Parameter Extraction of Nonlinear Systems

The parameter extraction of nonlinear systems can be split between the extraction process for memoryless nonlinearities and extracting parameters of nonlinear systems with memory.

5.2.1 Memoryless Nonlinearities

Since memoryless nonlinearities are not frequency-dependent, measurement or simulation results of the DUT from different input voltages/currents is sufficient to construct a nonlinear curve of specified form (polynomial, PWL, ...) through the data set. Nonlinear curve fitting methods like the Levenberg–Marquardt algorithm [93] or the Trust region approach [94] can be used to minimize the fitting error. The MATLAB® *fit* function provides a fully implemented curve fitting method where the user can choose the type of curve and the fit algorithm. Furthermore, the MATLAB®

routine is not limited to functions of one variable, but also multivariate nonlinearities can be fit.

5.2.1.1 Example

In Fig. 5.2 the output frequency of the AixRF's VCO is shown which depends on the two analog voltages v_{mod} and v_{in}. v_{in} is connected to the loop filter output and v_{mod} controls the direct frequency modulation path.

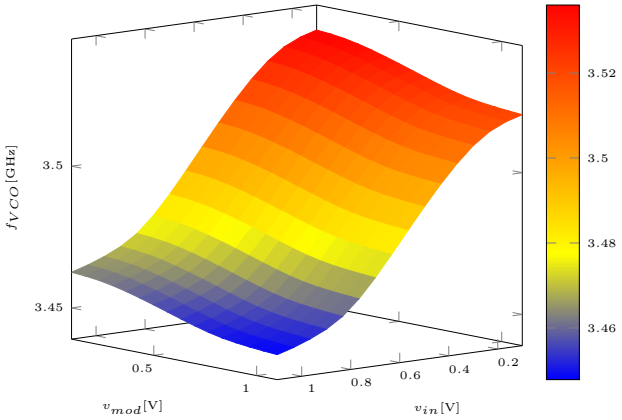


Figure 5.2: Dependency of the VCO frequency from v_{in} and v_{mod}.

With the help of MATLAB®'s *fit* function, a bivariate polynomial (eq. 5.3) of total degree ≤ 3 can be fit to the simulated data. Furthermore, the relative fitting error can be calculated which is shown in Fig. 5.3. The error ranges between -4×10^{-4} and 4×10^{-4} which is accurate enough for system-level simulations.

$$
\begin{aligned}
f_{VCO} = {} & p_{00} + p_{10} \cdot v_{in} + p_{01} \cdot v_{mod} + p_{20} \cdot v_{in}^2 \\
& + p_{11} \cdot v_{in} \cdot v_{mod} + p_{02} \cdot v_{mod}^2 + p_{30} \cdot v_{in}^3 \\
& + p_{21} \cdot v_{in}^2 \cdot v_{mod} + p_{12} \cdot v_{in} \cdot v_{mod}^2 + p_{03} \cdot v_{mod}^3
\end{aligned}
$$

with: $p_{00} = 3534267844.636903$, $p_{10} = 42095151.692583$, $p_{01} = 6355753.815542$,
$p_{20} = -268459779.851982$, $p_{11} = 541759.654017$, $p_{02} = -52261106.689708$,
$p_{30} = 156119851.830017$, $p_{21} = 148611.070302$, $p_{12} = 164383.477808$,
$p_{03} = 30739187.673398$

(5.3)

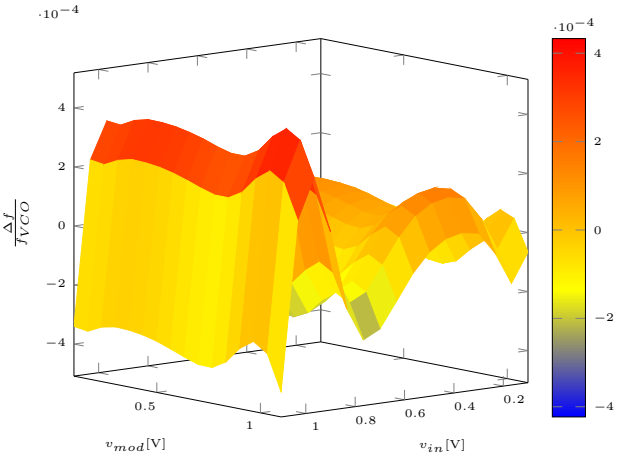


Figure 5.3: Relative fitting error.

5.2.2 S_M-type models

As mentioned in chapter 4 one advantage of modelling nonlinearities as S_M-type parallel L-N-L block structure is the straightforward parameter identification using steady-state system responses. In [81] and [95] the approach to determine the nonlinearity and the filter coefficients is described. Since in each L-N-L branch the nonlinearity consists only of a monomial, the steady-state response of the k-th

branch can be calculated to

$$y_k(t) = A^k \cdot \sum_{q=0}^{k} \binom{k}{q} H_{k,0}^{k-q}(\omega) H_{k,0}^{q}(-\omega) H_{k,1}((k-2q)\omega) \exp(j(k-2q)\omega t) \quad [81]. \tag{5.4}$$

The output of the complete S_M system is the sum of all M branch responses y_k from eq. 5.4. The result can be ordered by its frequency components and rewritten as shown in eq. 5.5.

$$\begin{aligned} y &= h(A,\omega) + \sum_{k=1}^{M} f_k(A,\omega)\exp(jk\omega t) + f_k(A,-\omega)\exp(-jk\omega t) \\ \text{with:}\quad h(A,\omega) &= \sum_{q=1}^{\lfloor M/2 \rfloor} A^{2q}\binom{2q}{q} |H_{2q,0}(\omega)| H_{2q,1}(0) \\ \text{and}\quad f_k(A,\omega) &= \begin{cases} A^k \cdot \sum_{r=0}^{(M_k-k)/2} A^{2r} v_{2r+k,(k-2)/2}(\omega); & k \text{ is even} \\ A^k \cdot \sum_{r=0}^{(M_k-k)/2} A^{2r} v_{2r+k,(k-1)/2}(\omega); & k \text{ is odd} \end{cases} \end{aligned} \tag{5.5}$$

M_k corresponds to the highest power in S_M with the same parity as k. The functions $v_{k,q}$ are dependent on the linear transfer functions $H_{k,0}$ and $H_{k,1}$ which embrace the nonlinearities and are defined as follows:

$$v_{k,q}(\omega) = \begin{cases} \binom{k}{(k-2)/2-q} H_{k,0}^{(k+2)/2+q}(\omega) H_{k,0}^{(k-2)/2-q}(-\omega) H_{k,1}((2q+2)\omega); & k \text{ is even} \\ \binom{k}{(k-1)/2-q} H_{k,0}^{(k+1)/2+q}(\omega) H_{k,0}^{(k-1)/2-q}(-\omega) H_{k,1}((2q+1)\omega); & k \text{ is odd} \end{cases} \tag{5.6}$$

From eq. 5.5 it is clear that for a fixed value of $\omega = \omega_1$ the functions f_k are polynomials in A^2 and the functions $v_{k,q}$ are constant. Thus, using measured or simulated spectral output values at the frequency $k\omega$ from $1 + \frac{M_k - k}{2}$ different input amplitude values A is enough to determine the values $v_{k,q}(\omega_1)$ through polynomial interpolation [81]. The function $v_{k,(k-1)/2}$ for odd k, resp. $v_{k,(k-2)/2}$ for even k for $r = 0$ have the following

form:

$$\begin{aligned} v_{k,(k-1)/2} &= H_{k,0}^{k}(\omega)H_{k,1}(k\omega); \quad k \text{ is odd} \\ v_{k,(k-2)/2} &= H_{k,0}^{k}(\omega)H_{k,1}(k\omega); \quad k \text{ is even.} \end{aligned} \tag{5.7}$$

Repeating the polynomial interpolation for enough frequency points is sufficient to determine the function $v_{k,(k-1)/2}$ ($v_{k,(k-2)/2}$) by e.g. vector fitting as explained in sec. 5.1. Assuming that the pole and zero multiplicities in $H_{k,0}$ and $H_{k,1}$ are not greater than 1, $v_{k,(k-1)/2}$ ($v_{k,(k-2)/2}$) can be decomposed in $H_{k,0}$ and $H_{k,1}$ by the multiplicity of poles and zeros in $v_{k,q}$. For example when regarding the third branch and a pole of $v_{k,q}$ appears three times, it belongs to $H_{k,0}$ whereas if the pole has the multiplicity of 1 it belongs to $H_{k,1}$. A pole with the multiplicity of 2 is a pole of $H_{k,0}$ and a zero of $H_{k,1}$, etc.

This system identification method was implemented in a MATLAB® script. Results from HB frequency and input voltage sweeps are used as input data. Since most vector fitting algorithms are ill conditioned for transfer function fitting with pole multiplicities bigger than 1, poles and zeros from $H_{k,0}$ are often not on the exact same point but distributed as slightly different poles and zeros over a small area in the complex plane. Thus, before mapping the poles to either $H_{k,0}$ or $H_{k,1}$, a clustering algorithm merges poles and zeros which are close together to a pole (zero) with a higher multiplicity. To improve the nonlinear system identification method, the vector fitting algorithm described in [96] could be implemented, which solves the conditioning issue for poles with higher order multiplicities.

5.2.2.1 Examples

The parameter extraction was first tested on an artificial S_M-system of order 5. Table 5.1 shows the original and the fitted parameters.

As can be seen, the poles and zeros of the linear subsystems are identified with high accuracy. The gain of a branch is shifted completely to the second linear subsystem which is allowed without altering the overall transfer function. Both original and fitted system were excited with a sinusoidal signal $s_{in}(t) = 500\,\text{mV} \cdot cos(2\pi 1\,\text{MHz}t)$. The output spectra are compared in Fig. 5.4. The tones at DC, 1 MHz, 2 MHz, 3 MHz and 5 MHz match perfectly. There is no signal at 4 MHz since the fourth branch is missing in the S_M-system.

Next, a real-world example is examined. The LNA in the 2.4 GHz path of the AixRF receiver is regarded and a third order S_M-type model is created. The resulting model was compared in a single-tone and an intermodulation scenario with the original

Table 5.1: Comparison of original vs. fitted S_M example system.

original	fitted
$H_1 = \frac{1e7}{s+1e7}$	$H_1 = \frac{1e7}{s+1e7}$
$H_{20} = \frac{5e6}{s+5e6}, H_{21} = \frac{2e6}{s+2e6}$	$H_{20} = \frac{1}{s+5e6}, H_{21} = \frac{3.5e18}{s+2e6}$
$H_{30} = \frac{1e8}{s+1e8}, H_{31} = \frac{s}{s+1e7}$	$H_{30} = \frac{1}{s+1e8}, H_{31} = \frac{5e23s-4.8e23}{s+1e7}$
$H_{40} = 0, H_{41} = 0$	$H_{40} = 0, H_{41} = 0$
$H_{50} = \frac{s}{s+3.3333e6}, H_{51} = \frac{1e6}{s+1e6}$	$H_{50} = \frac{s+1.225}{s+3.333e6}, H_{51} = \frac{2.963e6}{s+1e6}$

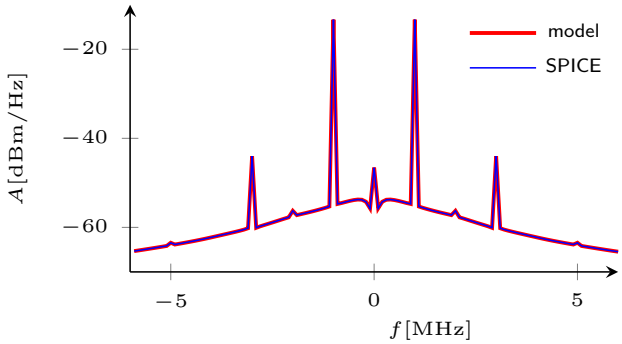
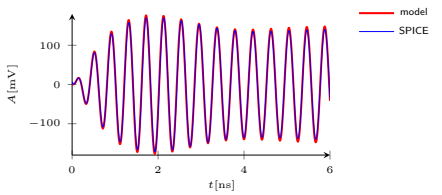


Figure 5.4: Output spectra of original and fitted S_M example system.

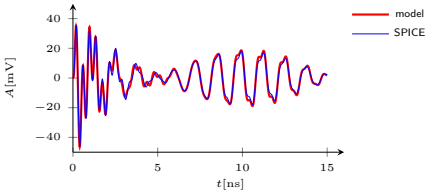
implementation on transistor-level. Fig. 5.5a and Fig. 5.5c show the response to the excitation with the input signal $s_{in}(t) = 20\,\text{mV} \cdot cos(2\pi 2.4\,\text{GHz}t)$. In Fig. 5.5b and Fig. 5.5d the output signal resulting from the input signal $s_{in}(t) = 40\,\text{mV} \cdot cos(2\pi 700\,\text{MHz}t) + 40\,\text{mV} \cdot cos(2\pi 800\,\text{MHz}t)$ is plotted. Despite the large BW and the wide dynamic range, in both cases the error at the harmonics and intermodulation products are less than 2 dB with only one exception where the model and the circuit differs by 3.5 dB.

5.2.3 Extraction of Supply and Substrate Noise Dependencies

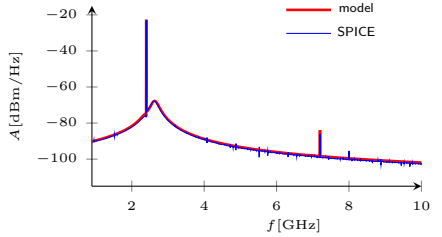
As discussed in sec. 4.6, the influence of supply and substrate noise on analog circuits can be modelled using multivariate polynomials. For the sake of simplicity, a differential system is regarded so that all even coefficients of the Taylor series



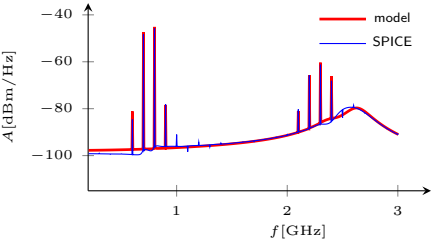
(a) Time-domain response to single tone excitation.



(b) Time-domain response to intermodulation scenario.



(c) Frequency-domain response to single tone excitation.



(d) Frequency-domain response to intermodulation scenario.

Figure 5.5: Model vs. schematic simulations.

are zero. Similar to extracting the coefficients of a S_M-system the amplitudes of the input and the supply signal are varied over the range of interest and the results of two-tone HB simulations are used for parameter extraction. It is assumed that the input signal's frequency is much higher than the frequency of the supply noise. Thus, the frequency dependent effects affects the main signal's fundamental component at $f_1 = f_{in}$ about the same as the signal components including the supply noise at $f_{2,3} = f_{in} \pm f_{supply}$. Similar to the extraction of nonlinear coefficient in sec. 5.2.1, the coefficients of the multivariate polynomial can be found by polynomial fitting using the fundamental frequency component and the intermodulation components at $f_{2,3}$ as input data. An example of the dependency of the first supply intermodulation component at $f_2 = f_{in} + f_{supply}$ from the input signal's and supply noise' signals amplitude is shown in Fig. 5.6 for the 2.4 GHz receiver LNA in the AixRF frontend with $f_{in} = 2628$ MHz and $f_{supply} = 32$ MHz.

In Fig. 5.7 a comparison of a supply noise simulation of the original transistor level LNA to the model implementation is done. The LNA sinusoidal differential input signal has a magnitude of 5 mV and a frequency is 2.65 GHz. As supply noise a sinusoidal interferer with an amplitude of 50 mV and a frequency of 40 MHz is chosen. As can be seen the model matches the fundamental and third harmonic coming from

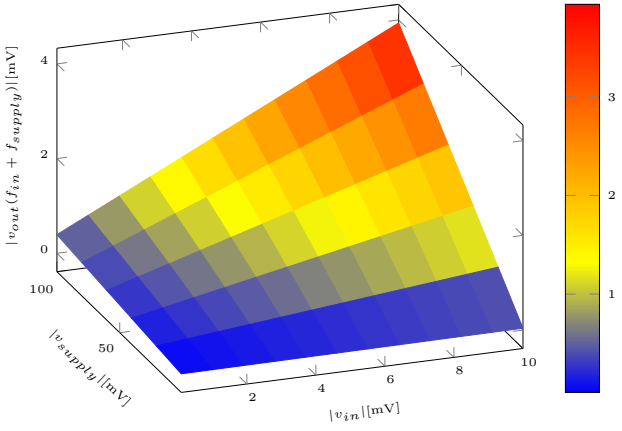


Figure 5.6: Supply intermodulation component at 2660 MHz.

the input signal. The intermodulation products at 2.65 GHz ± 40 MHz differs about 0.62 dB, respectively about 1.6 dB, whereas the intermodulation products at higher order intermodulation products are not correctly reproduced by the model but since their amplitude is low compared to the main signal, they can be neglected without introducing a big error.

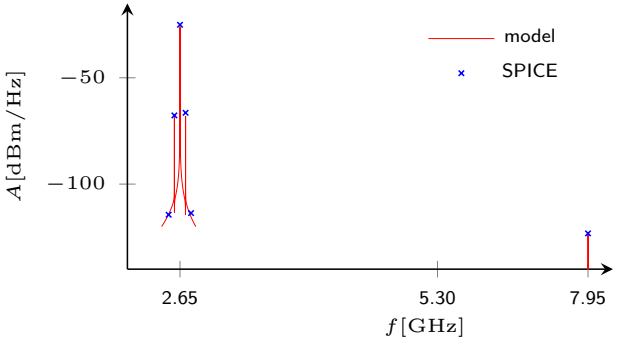


Figure 5.7: Output spectra of original LNA vs. fitted model.

5.3 Extraction of Phase-domain Parameters

As mentioned in chapter 4 instead of describing highly nonlinear circuits in the voltage/current domain it can be advantageous to utilize their description in the phase regime. A linear system is best modelled using the transfer function from in- to output. In sec. 5.1 of this chapter a method is described to extract the transfer function of linear systems from AC simulation results, but the AC simulation can only be applied to linearized circuits in the voltage/current domain. In [55] a method is described to extend a small signal frequency-domain analysis to circuits that behaves nonlinear in voltage or current, but that are linear in other regimes such as phase or frequency. Since circuit simulators choose node voltages and branch currents as variables, they can only determine the transfer functions in the voltage or current domain. In [55] the concept of variable domain translators is introduced which can translate the perturbation of one domain to another. The variable domain translators are pseudo circuits which can for example be implemented in an HDL like VerilogAMS. For extracting a transfer function in the phase domain, a testbench as shown in Fig. 5.8 can be used. A PSS followed by a *PAC* simulation is performed to simulate the system which is nonlinear in the voltage/current domain.

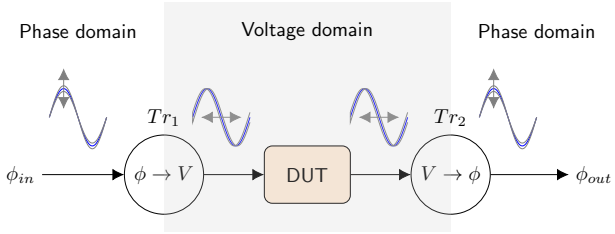


Figure 5.8: Phase domain simulation.

In the testbench, the source serves as small signal excitation in the phase domain. The small signal perturbation is translated in the voltage/current domain using the variable domain translator Tr_1, propagates through the DUT and is translated back in the phase domain with translator Tr_2. There, the periodically time-varying transfer functions can be evaluated. Since the system behaves linear in the phase domain, the zero-order sideband is equal to the phase transfer function and all other sidebands are ideally zero.

Fig. 5.9 shows the results of a small signal phase domain simulation over a frequency range from 0 to 5 MHz applied to a frequency divider by 6. The divider is driven by a periodic large-scale signal of 192 MHz. The input small signal level is chosen to 1 and a setup as in Fig. 5.8 is used. It can be seen that sideband 0 has a constant gain of

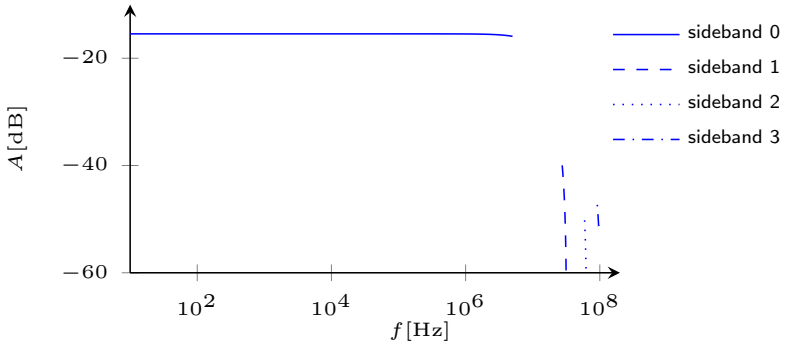


Figure 5.9: Phase domain simulation of a FD.

−15.6 dB which corresponds to the linear factor $\frac{1}{6}$. The low frequency dependence and the deviation of the other sidebands from zero are only artefacts of the nonideal translator elements.

In addition to the phase transfer function, also the influence of supply variations on the output phase can be simulated in a similar fashion. Therefore, the input small signal AC source is turned off and the supply is chosen as small signal excitation. At the output of translator Tr_2, the influence of the supply on the output phase can be observed.

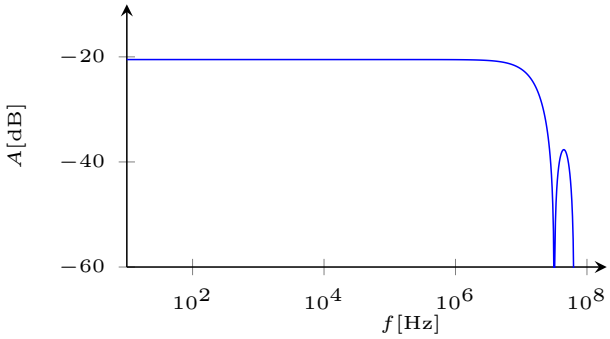


Figure 5.10: Simulation of supply variation influence on the output phase in a FD.

Because the FD is a sampled system, the transfer function has dips around multiples of the output frequency of 32 MHz. For small frequencies the transfer function is nearly constant with a gain of −20 dB.

CHAPTER 6

EVENT-DRIVEN SMALL-SIGNAL ANALYSIS

Up to now, the discussed event-driven simulation methods all take place in the time-domain and treat signals with amplitude levels that are not negligible with regards to the nonlinearities. The advantage of a transient analysis is its general applicability. Its disadvantage is that one transient analysis with one special excitation covers only one particular case and the expressiveness of the results cannot be easily transferred to other test cases. AC analysis on the other hand cannot be applied to nonlinear or time-varying circuits but it is the most efficient way to simulate LTI systems. Simulating enough frequency points allows the complete characterization of a linear circuit and the calculation of the response to an arbitrary excitation. Therefore, AC analysis can be regarded as a formal verification technique for linear, time-invariant analog circuits [55]. As mentioned in sec. 2.6.4 AC simulation methods can be transferred to periodically varying operation points if the small-signal does not disturb the biasing conditions. For SPICE-like simulators these simulation methods have been available for a long time but for an event-driven simulator a small-signal analysis is missing. Furthermore, classical HB or PSS simulations fail when dealing with mixed-signal systems since many of those systems exhibit a quasi-random behaviour e.g., a $\Delta\Sigma$-ADC. Thus, a periodic AC analysis cannot be executed because it is based on the operation point calculated from a HB or PSS analysis. An approach for a stochastic small-signal analysis of mixed-signal systems using Markov chains was implemented in [97] but it relies on solving equation systems for transition probabilities and therefore it does not fit in the proposed event-driven simulation framework. Since noise is a major problem in RF circuits, this chapter focusses on a noise analysis but with some small changes the methodology can also be applied as periodic AC analysis as shown later. The here presented simulation method takes advantage of the small amplitudes that noise usually has and it is assumed that the noise does not change the operation points of the system.

6.1 Small-signal Transfer Functions

For developing a small-signal event-driven simulation method which is capable to depict frequency conversion due to nonlinear or time-varying circuits, the small-signal behaviour of a nonlinear block excited with a large-scale sinusoidal signal is discussed using the example of the amplifier in Fig. 6.1. The circuit is excited with a sinusoidal signal of the frequency f_1 and the output noise at a frequency f_0 is observed.

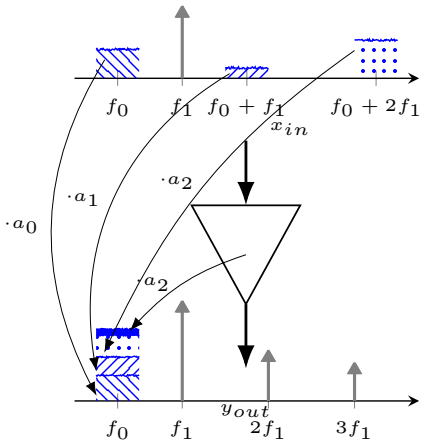


Figure 6.1: Different noise contributors in a nonlinear amplifier.

Due to the nonlinearities, not only the fundamental tone but also harmonics can be found at the output. The output noise centred around a frequency f_0 can be derived using the *describing function* F of the amplifier and assuming that the noise level is so small that it does not influence the periodically varying bias point of the circuit. The calculation is done in eq. 6.1- 6.4.

$$Y(t) = F(X(t)) = F(S(t) + s(t)). \tag{6.1}$$

The input signal X is composed of the large-scale signal S and a small-scale signal s, in this case the noise. Using the first order Taylor series approximation, the output signal Y can be approximated as shown in eq. 6.2.

$$Y(t) \approx F(S(t)) + \frac{dF(\xi)}{d\xi}\Big|_{\xi=S(t)} \cdot s(t). \tag{6.2}$$

In the frequency domain, eq. 6.2 can be rewritten as eq. 6.3.

$$\begin{aligned} Y(f) &= \mathcal{F}\{Y(t)\} \\ &= \mathcal{F}\{F(S(t))\} + \underbrace{\mathcal{F}\left\{ \frac{dF(\xi)}{d\xi}\Big|_{\xi=S(t)} \right\}}_{T(f)} \star \mathcal{F}\{s(t)\} \\ \Rightarrow s_{out}(f) &= T(f) \star s(f), \end{aligned} \tag{6.3}$$

Thus, the output small-scale signal $s_{out}(f)$ is the convolution of the transfer function $T(f)$ which can be calculated from the derivative of the block describing function F evaluated for the large-scale signal S and the input small-scale signal $s(f)$. Since the large-scale input signal is sinusoidal the transfer function $T(f)$ can be described as a Fourier series with the coefficients a_n, the output small-scale signal is the weighed sum of the input small-scale signal s downconverted to the band of interest from different input frequencies (eq. 6.4).

$$s_{out}(f) = \left(\sum_n a_n \delta(f - f_n) \right) \star s(f) = \sum_n a_n s(f - f_n) \tag{6.4}$$

In case of a noisy circuit, the internal noise will also be added to the output noise (see Fig. 6.1).

The exemplarily presented noise folding calculation is not only valid for nonlinear blocks with a single input but can be generalized to circuits with multiple inputs as well, e.g., mixers. In eq. 6.1, the function F and the signals X, S and s are simply replaced by their vectorial representation which leads to eq. 6.5. Moreover, the large-scale input is not restricted to sinusoidal signals but instead any periodic or quasi-periodic waveform which can be expressed as sum of discrete Dirac pulses in the frequency domain is possible.

$$
\begin{aligned}
& Y(t) = F(\underline{X}(t)) = F(\underline{S}(t) + \underline{s}(t)) \\
\Rightarrow\ & Y(t) \approx F\left(\begin{bmatrix} S_1(t) \\ S_2(t) \\ .. \\ S_n(t) \end{bmatrix}\right) + \begin{bmatrix} \frac{\partial F(\underline{\xi})}{\partial \xi_1} \\ \frac{\partial F(\underline{\xi})}{\partial \xi_2} \\ .. \\ \frac{\partial F(\underline{\xi})}{\partial \xi_n} \end{bmatrix}\Bigg|_{\underline{\xi}=\underline{S}(t)} \cdot \begin{bmatrix} s_1(t) \\ s_2(t) \\ .. \\ s_n(t) \end{bmatrix} \\
\Rightarrow\ & Y(f) = \mathcal{F}\left\{F\left(\begin{bmatrix} S_1(t) \\ S_2(t) \\ .. \\ S_n(t) \end{bmatrix}\right)\right\} + \mathcal{F}\left\{\begin{bmatrix} \frac{\partial F(\underline{\xi})}{\partial \xi_1} \\ \frac{\partial F(\underline{\xi})}{\partial \xi_2} \\ .. \\ \frac{\partial F(\underline{\xi})}{\partial \xi_n} \end{bmatrix}\Bigg|_{\underline{\xi}=\underline{S}(t)}\right\} \star \begin{bmatrix} s_1(f) \\ s_2(f) \\ .. \\ s_n(f) \end{bmatrix}
\end{aligned}
\tag{6.5}
$$

As mentioned in chapter 4, the nonlinearities are modelled using the *AMS_Polynomial* or the *AMS_MultivariatePolynomial* class. For both classes functions are implemented which analytically calculate the (partial) derivatives with respect to the arguments. The derivates are then used to determine the transfer function T with the large-scale spectral input signal X and the nonlinear *describing function* NL according to Fig. 6.2.

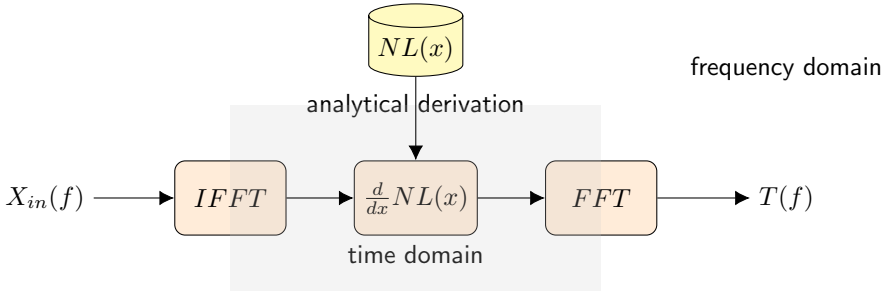


Figure 6.2: Calculation of the transfer function $T(f)$.

If the regarded block is a LTI system, the small-signal transfer function is independent of the operating point and is simply the transfer function characterising the linear circuit (eq. 6.6).

$$
s_{out}(f) = H(f) \cdot s(f) \tag{6.6}
$$

After discussing the derivation of the small-signal transfer functions for mildly nonlinear and linear circuits, a description of the most common RF transceiver blocks is possible.

Circuits like buffers, PFDs and FDs which are found in clock distribution or synthesizer circuits are usually linear in the phase regime even though they are strongly nonlinear in the current/voltage domain [55]. Exemplarily, the small-signal transfer function for a FD is derived in [98]. By means of analysing the propagation of spurs through a FD, a small-signal transfer function can be calculated. For calculation, a detour through the phase domain is done. In eq. 6.7 the input signal of a FD is analysed assuming a constant amplitude A_{in} and only a small phase jitter ϕ_{spur}. The signal can be approximatively decomposed into the sum of a large tone at the main frequency ω_{LO} and a signal part where the main tone is modulated by the phase jitter ϕ_{spur}. In the spectrum, this part can be found in close proximity to the main tone with a much smaller amplitude.

$$\begin{aligned} S_{in}(t) &= \Re\{A_{in}exp(j(\omega_{LO}t + \phi_{spur}(t)))\} \\ &= \Re\{(A_{in}exp(j\omega_{LO}t)) \cdot exp(j\phi_{spur}(t))\} \\ &\approx \Re\{(A_{in}exp(j\omega_{LO}t)) \cdot (1 + j\phi_{spur}(t))\} \\ &= \Re\{A_{in}exp(j\omega_{LO}t) + \underbrace{jA_{in}\phi_{spur}(t)exp(j\omega_{LO}t)}_{s_{spur,in}(t)}\} \end{aligned} \tag{6.7}$$

For the output signal, the equivalent calculation can be performed, but the phase is now divided by the divider factor N. It can be seen that the amplitude of the spurs is reduced by $\frac{1}{N}$ in comparison to the input spur but the offset frequency from the main tone is still the same since the argument of ϕ_{spur} is not scaled. This can be verified by inserting the term $A_{spur} \cdot cos(\Delta\omega t)$ for the phase noise ϕ_{spur}.

$$\begin{aligned} S_{out}(t) &= \Re\left\{A_{out}exp\left(j\left(\frac{\omega_{LO}}{N}t + \frac{\phi_{spur}(t)}{N}\right)\right)\right\} \\ &\approx \Re\left\{A_{out}exp\left(j\frac{\omega_{LO}}{N}t\right) + \underbrace{j\frac{A_{out}\phi_{spur}(t)}{N} \cdot exp\left(j\frac{\omega_{LO}}{N}t\right)}_{s_{spur,out}(t)}\right\} \end{aligned} \tag{6.8}$$

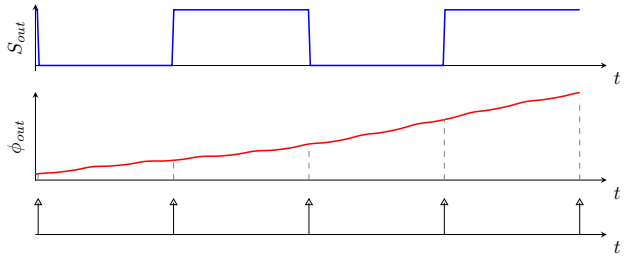


Figure 6.3: Sampling of the output phase.

$$
\begin{aligned}
s_{spur,in}(t) &= \Re\left\{ j\frac{A_{in}A_{spur}}{2} \cdot \left(exp\big(j(\omega_{LO} - \Delta\omega)t\big) + exp\big(j(\omega_{LO} + \Delta\omega)t\big) \right) \right\} \\
s_{spur,out}(t) &= Re\left\{ j\frac{A_{out}A_{spur}}{2N} \cdot \left(exp\big(j(\frac{\omega_{LO}}{N} - \Delta\omega)t\big) + exp\big(j(\frac{\omega_{LO}}{N} + \Delta\omega)t\big) \right) \right\}
\end{aligned}
\tag{6.9}
$$

The frequency shift of the input spurious tones to the output spurs can be derived from eq. 6.9. It is equal to $f_{shift} = (\omega_{LO} \pm \Delta\omega) - (\frac{\omega_{LO}}{N} \pm \Delta\omega) = \frac{(N-1)\omega_{LO}}{N}$.

Since a frequency divider deals with quasi rectangular signals, another effect of frequency dividers has to be considered. The output port switching of the FD is dominated by the main tone. Hence, for the output being either 1 or 0, the spurious tones do not affect the output signal. Thus, the spur influence can only be observed during transition periods of the signal. As a good approximation, it can be said that the phase noise is sampled by the edges of the output main tone [99] which is depicted in Fig. 6.3. Beneath the expected spurs, also replica of them arise at the output. In Fig. 6.4 both the sampling and the scaling can be observed. A clock signal with a frequency of 1.6 GHz passes a FD with the divider factor of 8. The input spurs at an offset frequency of 438 MHz are downconverted to the output spurs at 438 MHz offset with reduced amplitude. The 18 dB difference from −77 dBc to −95 dBc corresponds to the frequency division by 8 ($20 \cdot \log(8) = 18$ dB). Because of the sampling by the output signal edges, the spurs also get folded close to the main output tone with an offset frequency of 38 MHz.

Summing up, the small-signal transfer function for a FD can be reduced to a frequency shift by $\frac{(N-1)\cdot\omega_{LO}}{N}$, a scaling with $\frac{1}{N}$ and a sampling operation under the assumption that input and output amplitudes are constant. Similar approaches to calculate a

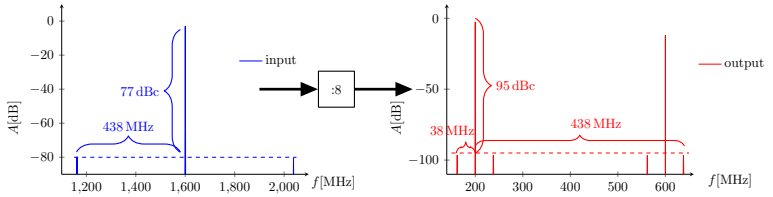


Figure 6.4: Simulated input and output spectra of a divide-by-8 FD.

small-signal transfer function are also applicable to other phase-domain circuits.

After covering the derivation for small-signal transfer functions for linear blocks in voltage/current and phase-domain and weakly nonlinear analog circuits, all important blocks in a RF transceiver system have been discussed.

6.2 System-level Noise Analysis

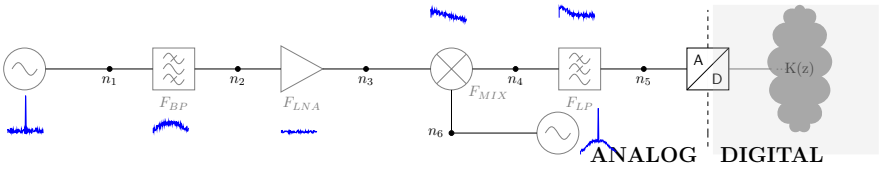
The idea of the system-level noise simulation is to analyse the influence of noise on one or more nodes in a system. The analysis method is split into two phases, which will be discussed in this section:

1. The simulation phase
2. The post-processing phase

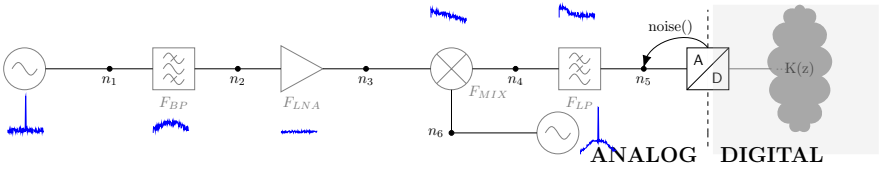
6.2.1 Simulation Phase

The noise analysis starts at a user-defined time point which must lie in the transient simulation interval. The algorithm starts at a node of interest and recursively steps backwards through the signal path of the DUT. On its way through the system, it collects all noise contributions from the circuits on the path which falls into the frequency band of interest while taking the small-signal transfer functions of the blocks into account.

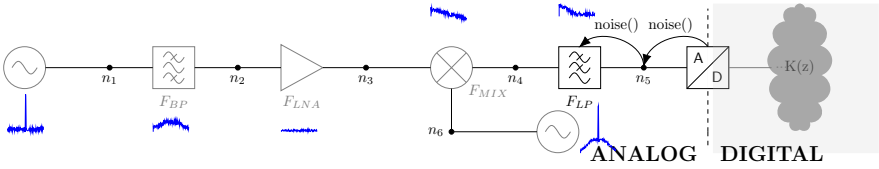
The idea is graphically illustrated using a generic receiver structure shown in Fig. 6.5. Here, the ADC's input noise is to be measured. Therefore, a *noise()* function is called, which collects the noise contributions from the ADC's input node n_5. In the next recursion step, the *noise()* function from node n_5 is called and collects the noise from blocks driving n_5, that is the lowpass filter. The filter itself is noisy, so the



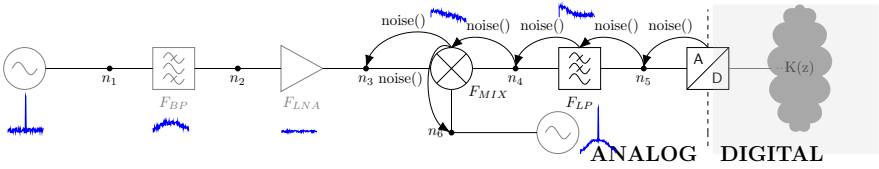
(a) Initial setup.



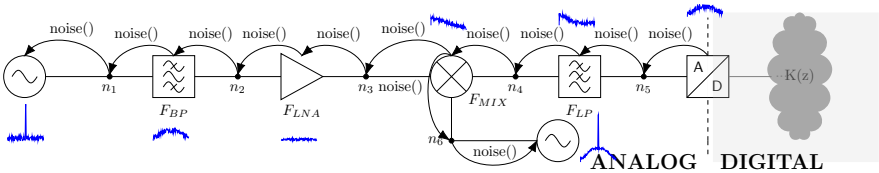
(b) Calling noise function for input node.



(c) Calling noise function of preceeding block from node.



(d) Blocks with multiple input signals.



(e) Resulting noise.

Figure 6.5: Idea of the noise analysis.

algorithm collects this contribution, calls the *noise()* function for the filter's driving node(s) and weights all noise components from preceding design blocks with the transfer function of the filter. If a node or a block has multiple inputs as the mixer

in Fig. 6.5d, the *noise()* function is called for each object driving the node or block. The recursion stops if a block with no inputs is found (e.g., a source) or if the noise transfer function from the inputs to the output is zero.

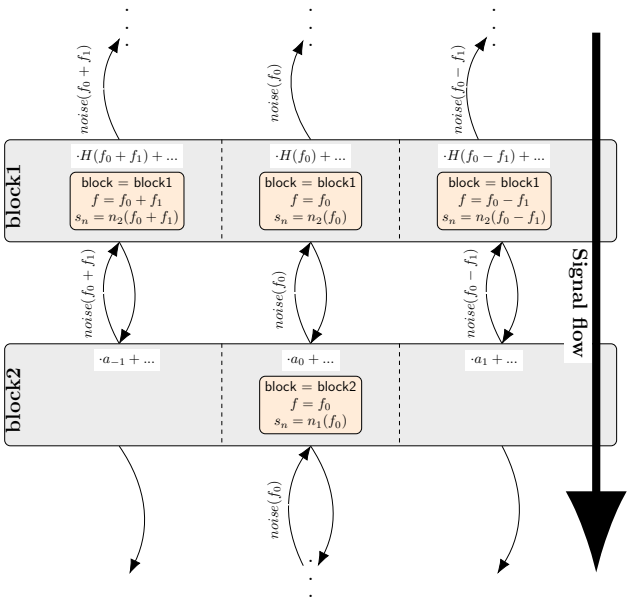


Figure 6.6: Cascade of a linear frequency dependent block (block1) and a nonlinear block (block2).

As derived in sec. 6.1, nonlinear blocks exited with a large-scale signal gather noise contributions from different frequency bands. Therefore, the noise collecting algorithm calls *noise* functions for each noise originating frequency band which is indicated in Fig. 6.6. In Fig. 6.6 a linear frequency dependent circuit is followed by a nonlinear block. The system is excited with a large-scale signal with the frequency f_1 and the noise at the output of the nonlinear block should be determined. The *noise* function is called at the output node of the nonlinear circuit and collects the internal noise at f_0. Because of the nonlinear distortion, the input noise of this block origins not only from the band-of-interest around f_0 but also from side bands shifted by multiples of the large-scale signal f_1. After the small-signal transfer function of this block is calculated and the factors a_i are determined, the noise collecting algorithm calls the *noise* functions for these frequency bands and weights the returned components with the corresponding a_i. The preceding linear circuit then adds its internal noise at the corresponding frequencies and calls then again the *noise*

functions for its input node at the frequencies $f_0 + i \cdot f_1$ and weights the returned noise with its transfer function evaluated at these frequencies. To limit the necessary computational effort, the coefficients of the noise transfer functions in nonlinear blocks are set to zero if they are below a certain threshold. This avoids the calculation of irrelevant noise components, speeds up the simulation and ensures more clarity in the result presentation.

To be able to separate the resulting noise by the contributors, only noise components of same origin block and same frequency band are added directly. A new signal structure called *AMS_SmallSignal* which is shown in Listing 6.1 was implemented for carrying the noise components. The *AMS_SmallSignal* signal type is composed of a vector of *AMS_SmallSignalComponent* objects which holds all different noise contributions (see. Listing 6.2). Furthermore, on the *AMS_SmallSignal* type operations like addition, a multiplication with a constant, a frequency shift or the processing with a linear transfer function are defined.

```
1 class AMS_SmallSignal : public AMS_Signal{
2 public:
3 //operations defined here
4 private:
5 std::vector<AMS_SmallSignalComponent> signalComponent;
6 [...]
7 };
```

Listing 6.1: AMS_SmallSignal class.

The *AMS_SmallSignalComponent* class possess the *origin* variable which stores the block where the noise component originates and the frequency band of the noise at its origin. In the evaluation phase, the noise signal can be decomposed based on this variable. Thus, blocks contributing most to the overall noise can be identified as well as the corresponding frequency band.

In the simulation setup, the system level noise analysis gets the starting time, the start and stop frequencies and a frequency step as parameters. At the starting time, a snapshot of the current system state is taken and the transfer functions are computed depending on the large scale signals. All signals are treated as periodic ones in their steady-state. The *noise* function is called with the starting frequency as argument, and the recursive algorithm starts. After collecting all noise contributions for the start frequency, they are saved in a *.csv* file, the frequency is increased by the defined step and the noise collecting functions continues to gather the noise for the new frequency. This procedure is repeated until the stop frequency is reached. After that, the normal transient analysis is continued.

```
class AMS_SmallSignalComponent{
public:
typedef struct {
AMS_Block* source;
double frequency;
} smallSignalSource;
[...]
private:
smallSignalSource origin;
dcmplx value;
[...]
};
```

Listing 6.2: AMS_SmallSignalComponent class.

Since all ports, nets and blocks have equivalent C++ objects (see Listing. 6.3), after the initial trigger from SV side, the complete analysis can be executed in C++. The implementation of the port respectively the net class contains pointers on attached nets and blocks, respectively ports. These pointers are used to step backwards through the design.

```
class AMS_Net : public AMS_Object{
public:
[...]
list<AMS_Port*> net_port_list;  //pointers to all ports attached
[...]
};

class AMS_Port : public AMS_Object {
public:
[...]
direction_type direction;   //direction of this port
AMS_Block* block;           //the block the port is associated with.
AMS_Net* net;               //the net that is attached to this port.
[..]
};
```

Listing 6.3: AMS_Net and AMS_Port classes.

As in the large-scale transient event-driven simulation, in the noise analysis the domain, the building block is modelled in, plays an important role. If a block is modelled in the phase-domain, it is automatically assumed that the noise is phase noise. The translator elements at the domain borders perform the noise conversion

similar to the calculation done in sec. 4.7.

6.2.2 Post-processing Phase

The post-processing phase has two tasks. It resolves digital feedback structures and calculates the influence of digital signal processing. Furthermore, it displays the simulated noise. For visualization, the noise contributions stored in the *.csv* file are read-in in the first step. Even IQ-systems can be visualized if two different noise files, one for the real part and one for the imaginary part, are available. Next, the noise contributions are sorted by their noise power and the overall noise is calculated from the different noise components. Due to the random nature of the noise, the noise processes from different blocks and different frequency bands cannot simply be added. Assuming that the noise contributions are uncorrelated, the absolute squares of the noise signals are summed up to get the resulting noise spectral density (eq. 6.10).

$$n_{res}(f) = \sum_{k} |s_{noise,k}(f)|^2 \tag{6.10}$$

For a user-friendly handling, a GUI was created which allows a fast and fail-safe evaluation of the simulation results. In the GUI the user can choose which noise contributions should be plotted. In Fig. 6.7 the results of an exemplary noise simulation are shown. The measurement point is at the output of the complex baseband filter. Noise contributions from the LNA from the frequency band around −2.68 GHz and noise from the supplies originating at the frequency band around −5.25 GHz are displayed. The shape of the noise is dominated by the frequency response of the baseband filter. The sum of the noise spectral densities of the plotted signals is calculated by the GUI to estimate the impact of the noise on the complete system.

The second task of the post-processing phase is to calculate the influence of the digital part of the mixed-signal system. After an analog signal is digitized, no further noise is added to that signal. Still, digital signal processing, e.g., filtering, can change the shape of the noise. Moreover, in mixed-signal feedbacks the noise is influenced by the transfer function of a closed mixed-signal loop. These effects need to be considered for a correct noise estimation.

An example on how the noise analysis deals with mixed-signal systems is illustrated by the ΔΣ-ADC shown in Fig. 6.8. The input signal S_{in} passes the loop filter before being digitized by a quantizer. The digital signal is fed back to the Digital-to-Analog

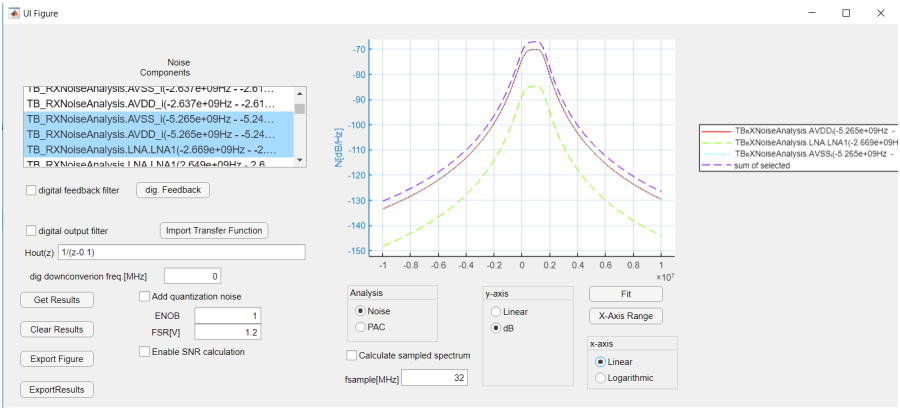


Figure 6.7: GUI for the noise analysis.

Converter (DAC). To cover a general case of a mixed-signal system, a digital filter $G(z)$ is put in the feedback path but usually in ΔΣ-ADCs this filter does not exist. The feedback signal is added to or subtracted from the input signal at node n_4. The output signal of the ADC is forwarded to a digital filter for downsampling and further signal processing.

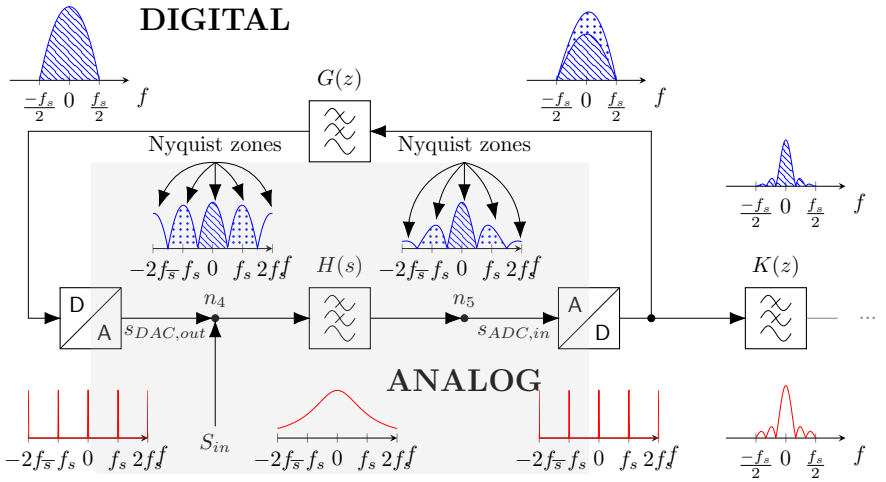


Figure 6.8: Noise in a mixed-signal system with feedback.

Due to the sampling, the parts in higher order Nyquist zones of the ADC's input

signal including the noise are folded in the first zone. In the DAC, it is the other way around and the signal gets copied from the main Nyquist zone to the other zones. The read-in noise signals can be considered as vector. The vector of each noise component can be reordered in multiple vectors according to eq. 6.11. The superscript of the vector entries illustrates in which Nyquist zone the component is located and the subscript corresponds to the frequency in the first Nyquist zone the component is folded to. The actual frequency of an entry ${}^{k}_{f_i}s$ is given by $f = f_i + k \cdot f_s$ with the sampling frequency f_s.

$$
\begin{aligned}
&[\underbrace{{}^{-n}_{f_0}s\ {}^{-n}_{f_1}s\ ..\ {}^{-n}_{f_i}s\ ..}_{\text{-nth Nyquist zone}}\ \underbrace{{}^{-n+1}_{f_0}s\ {}^{-n+1}_{f_1}s\ ..\ {}^{-n+1}_{f_i}s\ ..}_{\text{-nth+1 Nyquist zone}}\ ..\ \underbrace{{}^{1}_{f_0}s\ {}^{1}_{f_1}s\ ..\ {}^{1}_{f_i}s\ ..}_{\text{first Nyquist zone}}\ ..\ \underbrace{{}^{n}_{f_0}s\ {}^{n}_{f_1}s\ ..\ {}^{n}_{f_i}s\ ..}_{\text{nth Nyquist zone}}] \\
&\rightarrow \underbrace{\begin{bmatrix} {}^{-n}_{f_0}s \\ {}^{-n+1}_{f_0}s \\ .. \\ {}^{0}_{f_0}s \\ .. \\ {}^{n}_{f_0}s \end{bmatrix}}_{f_0 \frac{s}{}}, \underbrace{\begin{bmatrix} {}^{-n}_{f_1}s \\ {}^{-n+1}_{f_1}s \\ .. \\ {}^{0}_{f_1}s \\ .. \\ {}^{n}_{f_1}s \end{bmatrix}}_{f_1 \frac{s}{}}, .., \underbrace{\begin{bmatrix} {}^{-n}_{f_i}s \\ {}^{-n+1}_{f_i}s \\ .. \\ {}^{0}_{f_i}s \\ .. \\ {}^{n}_{f_i}s \end{bmatrix}}_{f_i \frac{s}{}}, ..
\end{aligned}
\tag{6.11}
$$

Using this signal representation, the sampling in the ADC and DAC can be described via matrix operations as shown in eq. 6.12 and eq. 6.13, respectively. The factors k and c represent the gain of the ADC and DAC.

$$
s_{ADC,out}(f) = \underbrace{\begin{pmatrix} k & .. & k & .. & k \end{pmatrix}}_{\boldsymbol{H_{ADC}}} \cdot \begin{pmatrix} s_{ADC,in}(f - nf_s) \\ .. \\ s_{ADC,in}(f) \\ .. \\ s_{ADC,in}(f + nf_s) \end{pmatrix}, \tag{6.12}
$$

$$\begin{pmatrix} s_{DAC,out}(f - nf_s) \\ .. \\ s_{DAC,out}(f) \\ .. \\ s_{DAC,out}(f + nf_s) \end{pmatrix} = \underbrace{\begin{pmatrix} c \\ .. \\ c \\ .. \\ c \end{pmatrix}}_{\boldsymbol{H_{DAC}}} \cdot s_{DAC,in}(f), \tag{6.13}$$

The digital input signal of the DAC input can be calculated according to eq. 6.14. Since the analog loopfilter is linear it does not cause a frequency conversion. Thus, the relation between in- and output can be represented as matrix vector multiplication where the matrix $\boldsymbol{H_{LF}}$ has only entries on its diagonal (see. eq. 6.15). The transfer function of the analog filter is equal to the simulated noise contributions coming from the DAC when setting its noise amplitude to 1. This is exploited in the post-processing phase to get the matrix entries $H(f + kf_s)$.

$$s_{DAC,in}(f) = s_{ADC,out}(f) \cdot G(z)\big|_{z=exp(j2\pi f)} \tag{6.14}$$

$$\begin{pmatrix} s_{LF,out}(f - nf_s) \\ .. \\ s_{LF,out}(f) \\ .. \\ s_{LF,out}(f + nf_s) \end{pmatrix} = \underbrace{\begin{pmatrix} H(f - nf_s) & 0 & .. & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & .. & H(f) & .. & 0 \\ .. & .. & .. & .. & .. \\ 0 & .. & 0 & .. & H(f + nf_s) \end{pmatrix}}_{\boldsymbol{H_{LF}}} \cdot \begin{pmatrix} s_{LF,in}(f - nf_s) \\ .. \\ s_{LF,in}(f) \\ .. \\ s_{LF,in}(f + nf_s) \end{pmatrix}. \tag{6.15}$$

Naming the signal which describes the ADC's input signal without considering the closed loop $\underline{s}_{ol}$ and the signal considering the loop $\underline{s}_{cl}$, a relationship can be found between those two signals which is formulated in eq. 6.16.

$$
\begin{aligned}
\underline{s}_{cl} &= \underline{s}_{ol} + \boldsymbol{H_{LF}} \cdot \underline{s}_{DAC} \quad \textit{and} \\
\underline{s}_{DAC} &= \boldsymbol{H_{DAC}} \cdot G(z)\big|_{z=e^{j2\pi f T_s}} \cdot \boldsymbol{H_{ADC}} \cdot \underline{s}_{cl} \\
\Rightarrow \underline{s}_{ol} &= \underbrace{\left[\mathbf{1} - \boldsymbol{H_{LF}} \cdot \boldsymbol{H_{DAC}} \cdot G(e^{j2\pi f T_s}) \cdot \boldsymbol{H_{ADC}}\right]}_{\boldsymbol{M}^{-1}} \cdot \underline{s}_{cl} \\
\Rightarrow \underline{s}_{cl} &= \boldsymbol{M} \cdot \underline{s}_{ol}.
\end{aligned}
\tag{6.16}
$$

The conversion matrix $\boldsymbol{M}$ is used to calculate the input referred noise at the ADC's input by using the cyclostationary noise conversion theory presented in [100] and [101]. There, it is stated that the Fourier-transformed autocorrelation matrices of the open-loop $\boldsymbol{N_{ol}}$ and closed-loop noise $\boldsymbol{N_{cl}}$ obey the relation described in eq. 6.17, where $\boldsymbol{M}^{\dagger}$ denotes the conjugate transpose of $\boldsymbol{M}$.

$$
\boldsymbol{N_{cl}} = \boldsymbol{M} \cdot \boldsymbol{N_{ol}} \cdot \boldsymbol{M}^{\dagger}. \tag{6.17}
$$

If the noise contributions of the different Nyquist zones are uncorrelated the matrix $\boldsymbol{N_{ol}}$ has only entries on the main diagonal which corresponds to the absolute square value simulated small-signals noise signals from a specific blocks and frequency band split into the Nyquist zones. The resulting closed loop spectral noise densities are located on $\boldsymbol{N_{cl}}$'s main diagonal. The other matrix entries describe the cross correlation between the noise contributions. Eq. 6.17 is individually evaluated for every noise contribution coming from the simulation phase and every simulated frequency step in the first Nyquist zone.

The user initially needs to define where the loop ic closed, e.g. at the DAC's output, and must eventually set the transfer function in the digital feedback path. Furthermore, the used sampling rate must be known to the program.

Since every analog-to-digital conversion process adds a certain amount of quantization noise, the GUI offers the possibility to take this into account. The quantization noise spectral density is calculated in eq. 6.18 based on the quantization step sizes and the sampling rate according to [102]. It is treated like the other noise contributions and thus the shaping by the feedback loop also effects the quantization noise.

$$
n_Q(f) = \begin{cases} \frac{\Delta^2}{12 f_s} & -\frac{f_s}{2} \leq f < \frac{f_s}{2} \\ 0 & \text{else} \end{cases} \tag{6.18}
$$

The step Δ can be derived from the Effective Number of Bits (ENOB) and the Fullscale Range (FSR) of the quantizer [103] (eq. 6.19).

$$\Delta = \frac{FSR}{2^{ENOB} - 1} \tag{6.19}$$

Additionally to resolving mixed-signal feedback loop, the implemented post-processing routine allows to consider the influence of digital signal processing on the noise. This is for example useful to calculate the noise level at the demodulator's input after digital filtering of the ADC's output signal. It is done after the feedback loops are resolved and the signals are already split in the Nyquist zones. The digital input noise is then calculated as shown in eq. 6.12. The resulting output noise power spectral densities are computed by multiplying the digital input noise with the absolute squared value of the transfer function $K(z)$ of the digital filter (eq. 6.20). The transfer function can either be explicitly specified by the user as rational function in z or the amplitude response from a simulation or measurement can be imported. Also, frequency conversion in the digital system, e.g., through a digital mixer, can be considered.

$$n_{dig,out}(f) = |K(z)|^2\big|_{z=exp(j2\pi f)} \cdot n_{dig,in}(f) \tag{6.20}$$

6.3 Proof of Concept

As accuracy check, a simple system consisting of the cascade of a nonlinear amplifier, a filter, a second nonlinear amplifier and a second filter were simulated [104] once using a HB in combination with a *hbnoise* analysis in SpectreRF and once the here proposed system-level noise analysis.

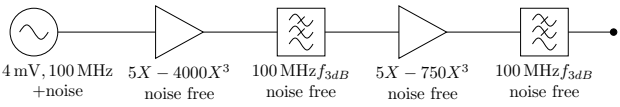


Figure 6.9: System used to compare the system-level noise simulation to a conventional noise analysis.

The first amplifier's characteristic function is $X_{out} = 5X_{in} + 4000X_{in}^3$ and the transfer function of the second amplifier is $X_{out} = 5X_{in} + 750X_{in}^3$. The filters are first order

lowpass LTI systems with $f_{3dB} = 100\,\text{MHz}$ and unity gain at $f = 0\,\text{Hz}$. The chain was exited with a sinusoidal signal $S_{in} = 4\,\text{mV} \cdot cos(2\pi 400\,\text{MHz}t)$. The input source also serves as only noise source in this system with an noise spectral density of 1 dBm/Hz. In Fig. 6.10 the noise contributions coming from different frequency bands are compared. For the sake of clarity, only the single sided spectrum is plotted. The results from the system-level noise analysis matches the SpectreRF results accurately and only due to numerical inaccuracies [104] the curves deviate by a maximum of 0.335 dB from each other.

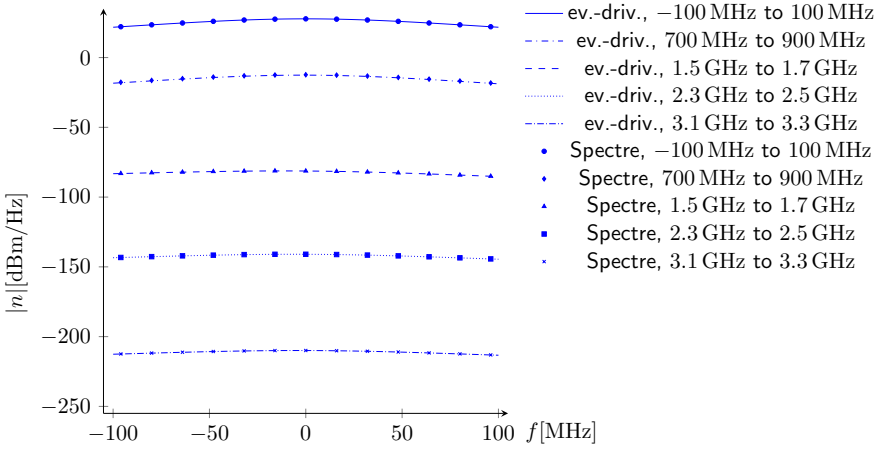


Figure 6.10: Comparison to a Spectre HB noise simulation.

6.4 Outlook: Extension to a System-level AC and sensitivity analysis

At the moment the noise analysis is not able to evaluate nested loops correctly since the innermost loop must be resolved first, followed by the second innermost, and so on. For the post-processing phase, it is therefore crucial to know in which loop the noise signal is originated. This could be solved by tagging the noise signal with a loop hierarchy counter variable which is increased every time the signal passes a loop break point. The post-processing phase could then evaluate the loops based on the counter variables.

Instead of simulating noise in a system, the proposed methodology can also be extended with only few changes to a system-level AC simulation. Similar to the

system-level noise analysis, the AC simulation is started at a given point of time. A snapshot of the large-scale signal is taken and the small-signal transfer functions of the nonlinear blocks are calculated. At blocks of interest, a system-level AC simulation can be started by calling the *ac* analysis algorithm which traverses the design recursively and collects the AC contributions from the sources. Instead of searching all components which will be mapped to a given output frequency, the AC function returns the complex phasor and its associated frequency. If frequency conversion occurs, the *ac()*-function creates new small-signal objects and changes their frequency and phasor accordingly. Phasors of *ac* components with the same frequency can be added. At the evaluation points, the small-signal phasors and their frequencies can be saved in a *.csv* file and handed over to the post-processing phase. As for the noise simulation, in this phase feedback loops and the influence of digital signal processing can be resolved.

Another application of a small-signal simulation could be a sensitivity analysis. The output signal of each block can be described either in frequency domain or time domain by a transfer function F. The transfer function describes the underlying circuit. If some circuit parameters vary due to temperature or mismatch, the transfer function and thus the output signal changes. In order to design robust systems, the influence of such deviations from the ideal value must be analysed and minimized. For small changes, the transfer function deviation can be described by a first order Taylor series. The output of system is described by a function F which depends on the variable vector $\underline{w}$.

$$ {}^{0}X_{out} = F(X_{in}, \underline{w}) \tag{6.21} $$

For small deviations $\Delta\underline{w}$, the output signal can be described as follows.

$$ X_{out} = {}^{0}X_{out} + \underbrace{\frac{\partial F(X_{in}, \underline{w})}{\partial w_1}\Delta w_1}_{\Delta x_1} + \underbrace{\frac{\partial F(X_{in}, \underline{w})}{\partial w_2}\Delta w_2}_{\Delta x_2} + \ldots \tag{6.22} $$

If not already in the frequency domain, the output signal components Δx_i can be translated using the FFT algorithm and can be treated as small-signal components. Similar to the noise analysis, a recursive implemented algorithm could fetch all Δx_i from the different blocks with respect to the small-signal transfer functions of the subsequent blocks. The sensitivity components could be stored in an object

containing the source block and the parameter Δw_i the component origins from. Thus, the influence of parameter variations on a given point in the design could be observed with this method.

CHAPTER 7

APPLICATION EXAMPLES

In this chapter, application examples of the proposed modelling and simulation approaches are given. The first example is an artificial one which should demonstrate the interaction of *XREAL* and spectral signals for modelling PSN. An LNA is supplied by a buck converter. The influence of the switching noise of the DC-DC converter on the output spectrum of the LNA is simulated using the proposed PSN models. The second example shows the use of the developed models and techniques in the verification process of the *AixRF* chip, an integrated wireless triple-band transceiver fabricated in a 130 nm CMOS process designed at the IAS.

7.1 Buck Converter

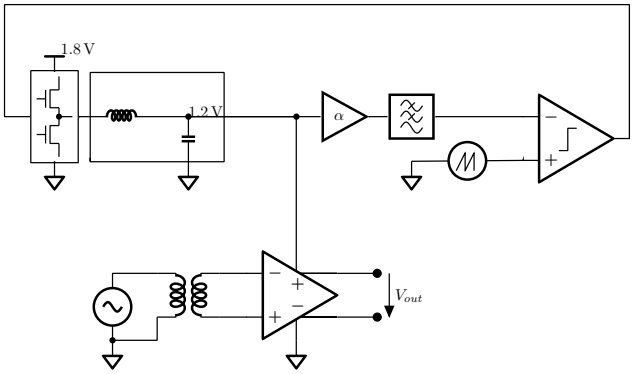


Figure 7.1: Test bench with buck converter and amplifier.

A small example which demonstrates the interaction between *XREAL* and spectral signals with regard to PSN is shown in Fig. 7.1. It is assumed that a nonlinear amplifier is supplied by a buck converter which steps down the voltage from 1.8 V to 1.2 V. The switching frequency is chosen to 10 MHz and the LC smoothing filter

consists of an inductor with 10 µH inductance and a series resistance of $R_{par} = 5\,\Omega$ and a capacitance of 100 nF. The input signal of the amplifier is a sinusoidal signal with a frequency of 2629 MHz and an amplitude of 50 mV.

It is advantageous to model the signals in the buck converter as *XREAL* signals for several reasons. First, the transient settling process is of interest which is more easily to depict with *XREAL* signals than with another kind of signal description. Moreover, the signal shapes in the buck converter are not arbitrary but are composed from a closed set of a priori known shapes which all can be described by *XREAL* signals. And last, the number of events is reduced in comparison to an equidistantly sampled signal description which results in a more performant simulation. On the other hand, the in- and output of the amplifier are modelled using spectral signals since it cannot be assumed that the input waveform shape can be decomposed in an *XREAL* signal and furthermore, the frequency conversion due to the nonlinearity is best described using spectral signals. The control and the clock signals are modelled using logic signals and thus, the task of the comparator model is to guess the crossings of the feedback voltage and the reference ramp voltage and create a logic output signal based on the crossing times.

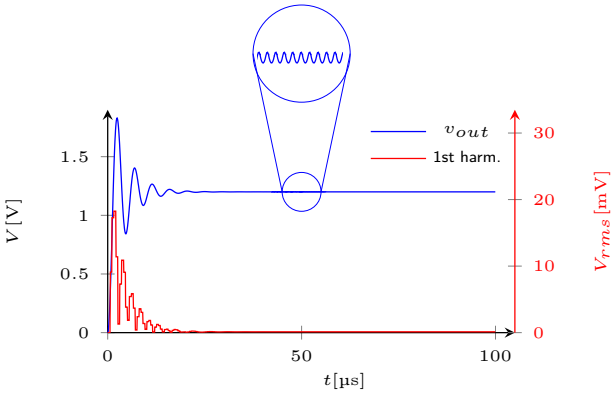


Figure 7.2: Output voltage of buck converter.

In Fig. 7.2 the buck converter's output voltage v_{out} and its spectral component at 10 MHz is plotted. It can be seen, that after the start-up, the output voltage settles at 1.2 V and has a small periodic ripple with a frequency of 10 MHz. At the beginning in the settling process when the activity is high the level of the spectral component at 10 MHz reaches an rms voltage of 20 mV but then decreases to approximately 125 µV.

Fig. 7.3 shows the spectral components at the output of the amplifier over time. The fundamental component at 2629 MHz (blue line) and the third harmonic (green line) are only weakly modulated by the settling process whereas the intermodulation products of the buck switching frequency and the amplifier's fundamental and third harmonic input tones at 2619 MHz and 7897 MHz are visibly influenced in the buck's start-up phase. The second order harmonic is largely suppressed since the amplifier is a differential one.

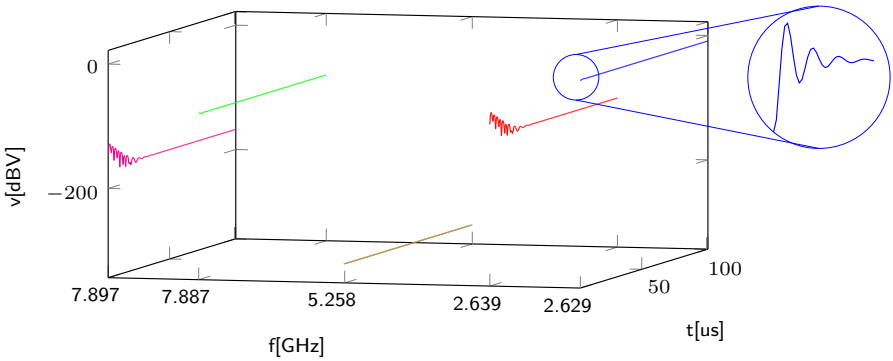


Figure 7.3: Spectral components at the output of the amplifier.

This small artificial testcase is a good example to show a few basic modelling and simulation techniques presented in this work. For each subsystem the most appropriate signal representation was chosen. By converting the *XREAL* signal into a spectral signal, the co-simulation of both systems is possible. The influence of the nonlinear amplifier is modelled using the methods presented in section 4.6. The spectral output components can be directly regarded during simulation because the spectral signal representation is used. When applying the methodology to a real-world example, intermodulation problems or other second order effects which might only appear on system-level are directly identified in the event-driven simulation.

7.2 AixRF

The *AixRF* chip is a fully integrated triple band wireless transceiver for robust and reliable communication. It was designed in a 130 nm CMOS process and covers the 433 MHz, 868 MHz-915 MHz and 2.4 GHz-2.5 GHz Industrial, Scientific and Medical (ISM) bands. The IC includes RF receiver and transmitter frontends for all three bands,

a ΔΣ-converter for digitizing the received signals, a fractional RF PLL serving as local oscillator, a quartz oscillator with clock management block and a power management block. Furthermore, the system comprises digital signal processing for modulating and demodulation, 2 kB of Static Random-Access Memory (SRAM) and a Serial Peripheral Interface (SPI) interface for communication with the chip. Additionally, the *AixRF* chip can monitor the 868 MHz or the 2.4 GHz band continuously with the help of an ultra-low power wake-up receiver for starting the rest of the system when detecting the wake-up signal.

The low-Intermediate Frequency (IF) receiver architecture of the *AixRF* IC mixes the incoming RF signals down to 1 MHz. The following ADC is a complex-valued, continuous-time ΔΣ-ADC with a sample rate of 96 MHz. To increase power efficiency, the Transmitter (TX) side was implemented as polar transmitters. The RF PLL can be modulated by two-point modulation directly by a digital word carrying the frequency of the modulation signal and the Power Amplifiers (PAs) are steered by the digital amplitude control signal.

All functional blocks in the transceiver can be controlled and adjusted by digital signals. Furthermore, some subsystems like the PLL have calibration loops to counteract mismatch and Process, Voltage and Temperature (PVT) variations. The complete system comprises about 20, 000 analog devices and 1 million digital transistors. The numbers of wires connecting the digital part with the analog part of the chip exceeds well the 1, 000, making it a true mixed-signal system. A system with such a high number of devices can only be analysed in a reasonable time and with reasonable resources using a sophisticated and performant event-driven simulation method like the one presented in this work.

The goal of the verification was on the one hand to ensure that the system works functionally, e.g., that registers can be written to, that every digital control signal is connected correctly and that no interface errors between the different functional blocks exist. On the other hand, the verification process should uncover performance issues of the RF systems, for example problems in the receiver chain due to unfitting baseband filter settings, unmatching signal levels or nonlinearity problems. Since functional verification is a well discussed field [14], this section focusses more on the behavioural simulation of the mixed-signal RF system.

The testbench for the AixRF transceiver is shown in Fig. 7.4. For the sake of clarity, all supply and bias sources and their connectivity were omitted. The DUT's RF inputs are fed with special signal sources, e.g., for testing a Gaussian Minimum Shift Keying (GMSK) receive case, a source supplying a spectral signal with a GMSK modulation is chosen. The transmitted data bits are created by a MATLAB® script and are read-in in the initialization step of the source. Furthermore, the output signals of the transmitter stages are probed by load models and the resulting waveforms are

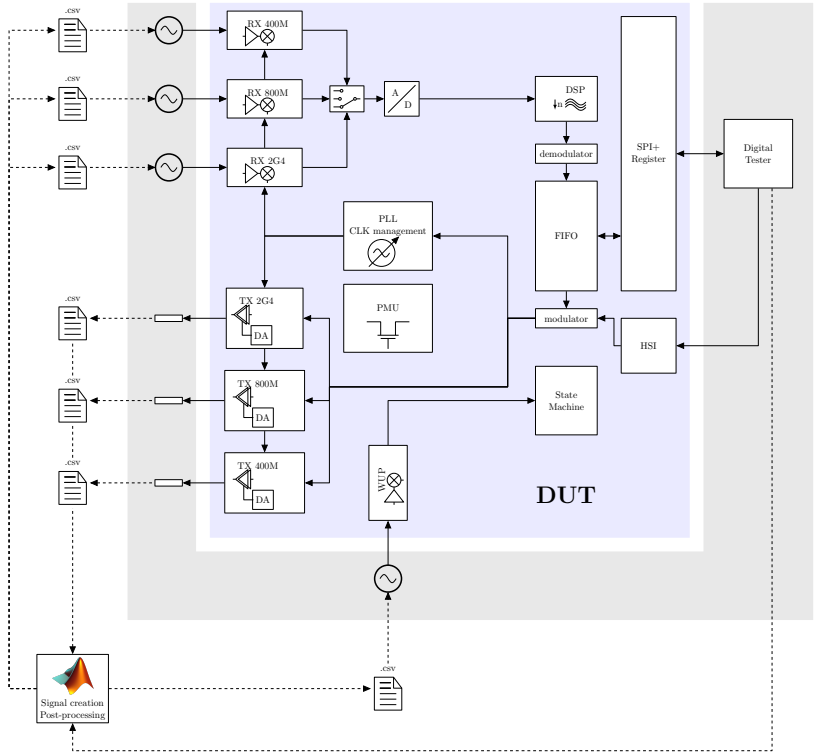


Figure 7.4: System level testbench of the AixRF.

saved in *.csv* files and can be post-processed by MATLAB® scripts or the signal analyser GUI (see. Fig. 7.5). The digital part of the chip is steered by a digital tester which communicates with the AixRF via the SPI protocol. Moreover, for high data rates it supports modulation data via a proprietary high-speed interface (HSI) to the chip. With this setup, the virtual IC is simulated as the real chip would be tested in the lab. It can be checked if all registers can be written to and read from, if the digitally assisted analog parts react as planned and if system-level algorithms work as expected.

The chosen modelling techniques are exemplarily explained using the analog front-tend of the 2.4 GHz path shown in Fig. 7.6. Both the RX as well as the TX architecture contain a sliding-IF mixer. The sliding-IF approach was chosen to reduce the needed VCO frequency which saves power. Furthermore, with this architecture it is possible

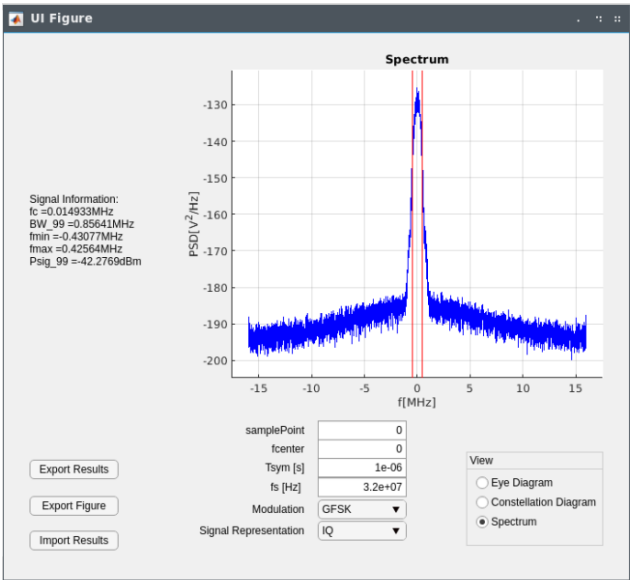


Figure 7.5: MATLAB® Signal Analyser GUI.

to use a single PLL and VCO for all three ISM bands. To achieve best simulation performance while maintaining a high accuracy, the modelling domains where partitioned as shown in Fig. 7.6. Since the VCO and the frequency dividers in the PLL and high-speed clock distribution paths have a constant amplitude, it is sufficient to pass only the phase information between the blocks. As explained in chapter 4 the advantage of this approach is that the description of these blocks simplifies by modelling them in the phase domain because they are designed to be linear in that regime while they are strongly nonlinear in the voltage domain. Also, all parts of the transmitter excluding the final PA stage are described in the phase domain for the same reason. The *XREAL* signal description is used to describe all phase-domain signals, since for those signals, the transient behaviour is of interest. Moreover, *XREAL* signals are superior in terms of speed and accuracy in comparison to other signal descriptions when dealing with linear systems describable by a confined set of basis waveforms.

The remaining RF parts dealing with the received signal and the output of the PA are modelled in the voltage domain using spectral signals. The signals in these parts do not necessarily have a constant envelope but can be arbitrarily shaped. Furthermore, the nonlinearities in the amplifiers and mixers need to be considered and thus, the

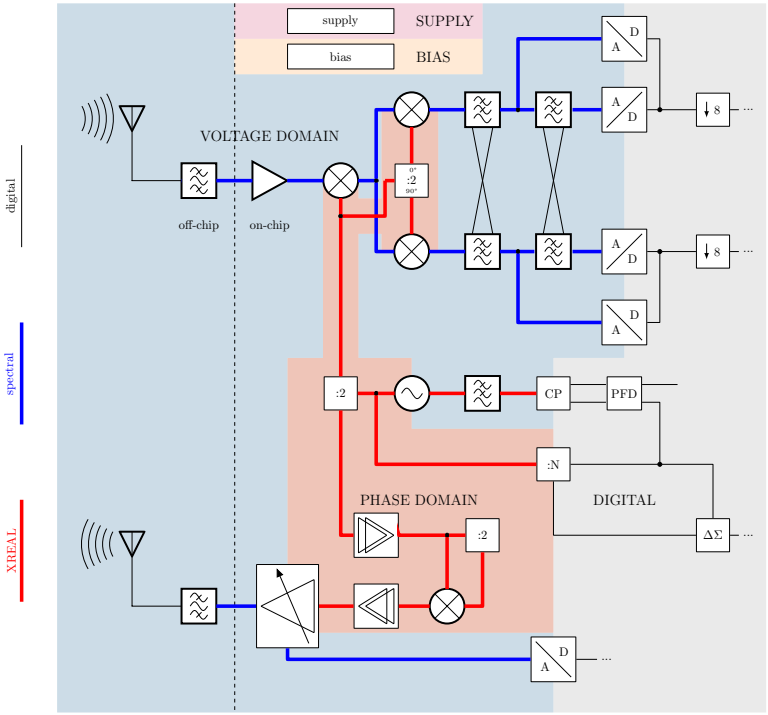


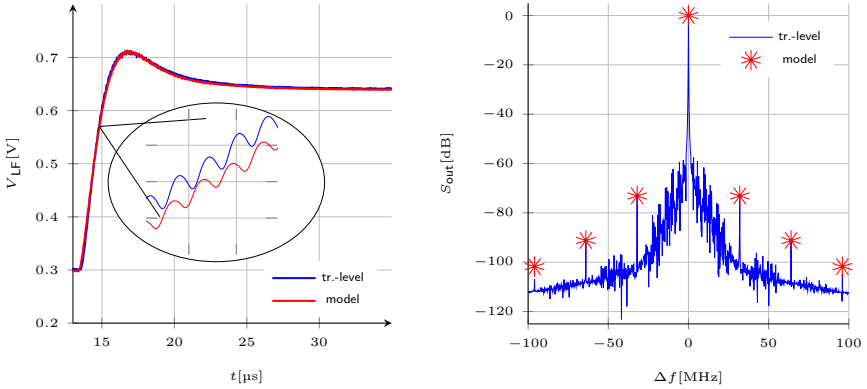
Figure 7.6: 2.4 GHz TX and RX path of the AixRF.

spectral signal description is the most beneficial one.

Since the PLL is a highly complex mixed-signal system which is responsible for the Local Oscillator (LO) generation and the phase modulation of the transmitter, a huge part of the verification was spent on checking its correctness. The simulation of integrated PLLs is extremely challenging. The VCO oscillates in the gigahertz range which imposes a small time step on the simulator, on the other hand the settling time can be up to a few microseconds. The large digital circuitry together with the huge number of events needed until reaching the steady-state results in an unacceptable computation time for SPICE-like simulators which complicates an iterative optimization on base of circuit simulations. Thus, the PLL is a perfect example to show the superiority of the RNM techniques presented in this work.

7.2.1 PLL Simulation Results

To check the accuracy and performance gain of the PLL modelling approach, the results of the event-driven simulations are compared against a transistor-level simulation. All event-driven simulation were executed on a desktop PC with an Intel i7-6700 processor clocked with 3.4 GHz and a RAM size of 16 GB. In Fig. 7.7a the loop filter voltage for a PLL settling scenario is shown. The targeted output frequency was chosen to 3.488 GHz which corresponds to a divider factor of 109. It can be seen that both simulations produce nearly identical results with only minor differences. The correlation between the two waveforms was calculated to 99.6%. The transistor-level simulation was executed using Cadence Spectre Accelerated Parallel Simulator (APS) on eight cores. For 50 µs of simulated time, it took one and a half day for the APS simulator but only 1.68 s for the event-driven simulation. Thus, the event-driven simulation is about the factor 66500 faster than the APS simulation. The spectral output signal can be calculated on-the-fly in the VCO model as described in section 3.6. The resulting spectral components are compared against the spectrum calculated from the APS simulation for steady-state conditions. It is shown in Fig. 7.7b and table 7.1 that the spurious tones match very accurately for both transistor-level and event-driven simulation.



(a) Time-domain.

(b) Frequency domain.

Figure 7.7: Comparison of model vs. schematic simulations.

The PLL model was published in [90]. In table 7.2 different PLL modelling approaches are compared to the approach presented here. The quotient of CPU time needed to simulate the systems with a conventional simulator and the time needed to simulate the proposed models is called speed-up and is used to compare the performance of

Table 7.1: Comparison of spur amplitudes.

f_{offset}	32 MHz	64 MHz	96 MHz
A_{model}[dB]	−73.13	−91.20	−101.79
$A_{tr.-level}$[dB]	−72.39	−91.54	−104.68
ΔA[dB]	−0.74	0.34	2.89

Table 7.2: Comparison with other modelling techniques.

	[60]	[105]	[62]	**This [90]**
Technique	PWL	PWC	mixed-domain	*XREAL*
Correlation	-	97.6%	-	99.6%
Sim. speed	1.33 µs/s	3 µs/s	16.9 µs/s	29.7 µs/s
Speed up	897.3[a]	5400.0[b]	-	66532.3[c]

Compared to: [a] SPICE [b] FAST SPICE [c] APS

the approaches. A PWL signal description approach was used in [60] whereas [105] used PWC signals. A generic SC PLL model using a time/frequency mixed-domain approach was presented in [62]. It can be seen that the *XREAL* approach with a simulation speed of 29.7 µs/s and a speed-up of 66532.3 is the most performant simulation while the accuracy still remains high. The approach was also compared against a model of the same system using baseband equivalent signal representations instead of *XREAL* signals. Furthermore, in the comparison model the VCO output is described in the voltage domain as fast oscillating digital signal and not in the phase domain. The simulation is slowed down about the factor 5 compared to the *XREAL* phase domain approach from [90] and therefore, demonstrates the superiority of the methodology used here.

In the *AixRF* RF PLL a calibration loop for coarse digital VCO tuning algorithm was implemented which is run at the start-up of the system and should guarantee that the oscillator's control voltage is always around $\frac{VDD}{2}$ independently of PVT influences and target frequency. After switching on the PLL, the VCO's digitally controlled capacitance is increased or decreased in binary weighted steps. A comparator compares the loop filter voltage to $\frac{V_{DD}}{2}$ and the algorithm decides on base of this feedback signal whether the capacitance needs to be increased or decreased. The algorithm stops if the Least Significant Bit (LSB) of the control word is set. Since the PLL does not react instantaneously on the capacitance change, the algorithm needs to wait for a given time until the comparison is evaluated.

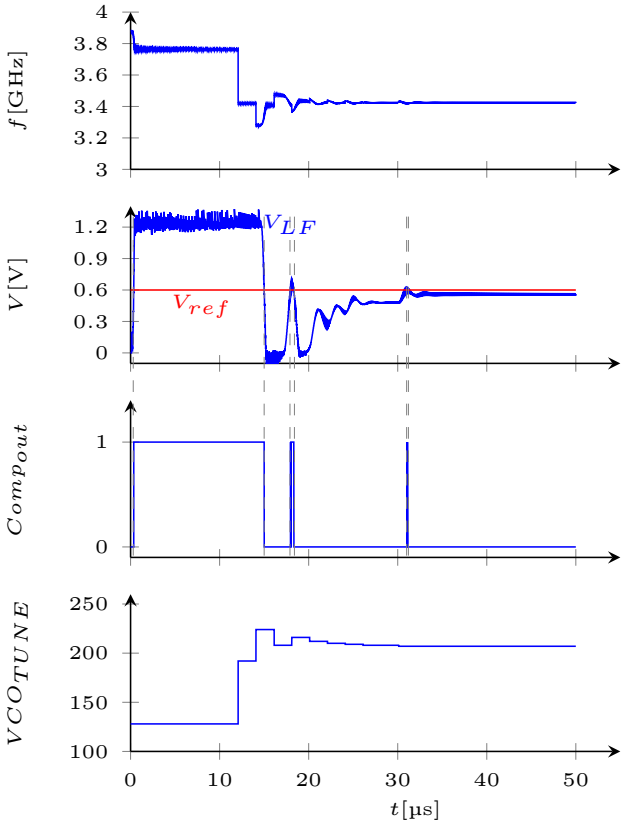


Figure 7.8: VCO Calibration.

In Fig. 7.8 the calibration process is shown. It can be seen that after turning on the PLL, the VCO is out of tune, the loop filter voltage is around VDD and the target frequency of 3.504 GHz cannot be reached. After the algorithm is started, the VCO tuning word is increased until the loop voltage drops below $\frac{VDD}{2}$ which means that too much capacitance is switched on. Now, capacitors are switched off until the comparator signal changes again to '1'. This procedure is repeated until all VCO control bits are set. By using a model-based simulation of the PLL it was possible to verify the functionality of the calibration algorithm. Furthermore, the algorithm parameters like the waiting times were optimized on base of the results. These steps would be inconvenient using a transistor-level simulation since it would take too much time.

7.2.2 Polar Transmitter Simulations

One high-level verification test-case is to check whether the RF transmitter can actually produce a distortion free and Bluetooth Low Energy (BTLE) compliant GMSK signal. Since the *AixRF* chip is designed for low power, it contains one polar transmitter for each supported ISM band. The polar architecture consumes less power than a conventional IQ transmitter architecture but usually the bandwidth is smaller and more effort must be put in the signal processing. The polar architecture makes use of the representation of a complex number in magnitude and phase. In the digital part of the system, the I and the Q component of the modulated signal is transformed into an amplitude and phase representation, e.g., with the CORDIC [106] algorithm. The phase is differentiated and the resulting frequency component is forwarded to the two-point modulator of the PLL. In the PLL, the modulation signal propagates on two different paths. Once, the VCO is modulated directly. The other modulation path leads via the ΔΣ-modulator, the Multi-Modulus Divider (MMD) and the other blocks of the PLL to the VCO. The PLL output signal which carries the phase information is then fed to the transmitter. The last stage of the PA is a digitally controlled amplifier whose gain is set by the amplitude of the modulation signal. In this amplifier phase and magnitude is combined, resulting in the wished transmitter signal.

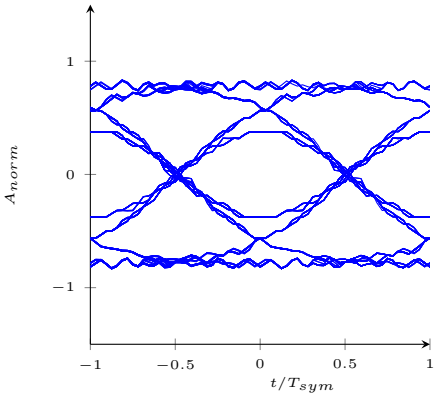
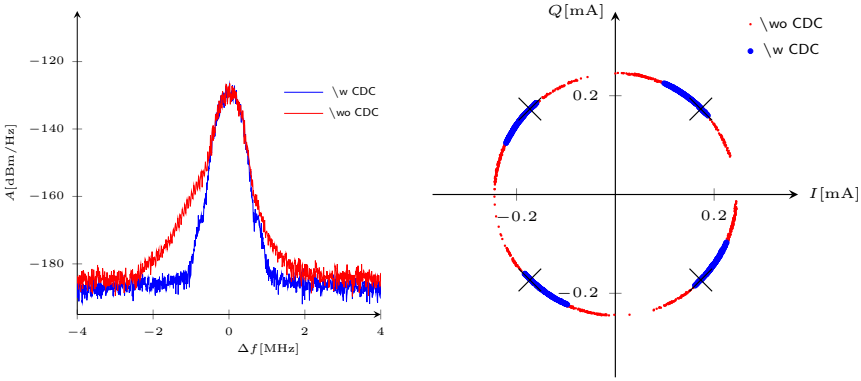


Figure 7.9: GMSK output signal from the CORDIC unit.

In Fig. 7.9 an eye diagram of a 1 MSym/s GMSK modulated signal at the output of the CORDIC unit is shown. It can be seen that the eye is widely opened which corresponds to a good signal quality. An event-driven simulation of the complete 2.4 GHz transmitter chain including the digital baseband processing, the RF PLL

and the power amplifier was done. The resulting spectrum of 795 symbols long simulation is displayed in red in Fig. 7.10a. Clearly, the output signal is distorted and violates the spectral mask specifications of the BTLE standard. The error could be tracked back to a Clock Domain Crossing (CDC) problem in the digital modulator of the PLL. In this block, the modulation signal crosses the domain from the digital clock to the clock coming from the MMD output of the PLL. Although both clocks are phase-locked in steady-state operation, when modulating the PLL the phase difference between the clocks varies over time. This leads to incorrect sampling points for some symbols of the modulation signal and causes the spectrum to spread. To counteract this effect, a dedicated block for a defined CDC was inserted. The resulting BTLE compliant spectrum is shown in blue in Fig. 7.10a. In Fig. 7.10b the constellation diagram for both cases is displayed. In case of using the CDC module, the symbols are confined around the ideal points while the signal points are scattered all around the unit circle if the module is missing.



(a) GMSK Spectra.

(b) Constellation diagram.

Figure 7.10: Transmitter output with and without CDC problems.

Since there are standards which demand a narrowband 1 MHz WLAN IEEE 802.11ah transmission, it was examined whether the *AixRF* transmitter is capable of supporting OFDM modulation. In Fig. 7.11 three different test cases are shown. The yellow curve shows the spectrum of the *AixRF* transmitter which was fed via a high-speed interface with polar digital baseband data. It can be seen that the plateau around the centre frequency is not flat but distorted and the edges of the spectrum are not steep but are slowly running out towards higher frequencies.

When changing the two-point modulation DAC and increasing its resolution from 8 bit to 10 bit, the resulting spectrum looks like the blue curve in Fig. 7.11. The

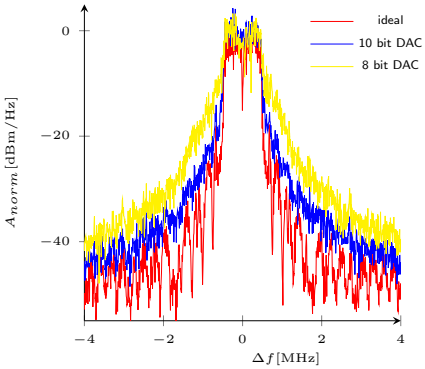


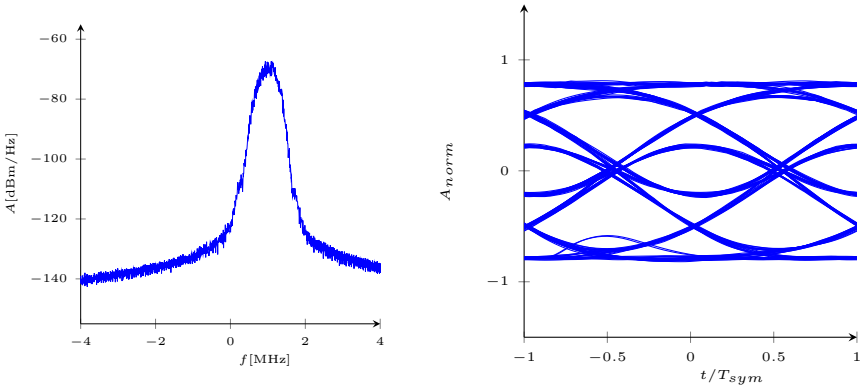
Figure 7.11: OFDM spectra.

steepness of the edges and the flatness of the plateau have increased. When neglecting the non-ideal phase transfer function coming from finite propagation delays of the preamplifier and the last PA stage and removing the anti-aliasing filter of the two-point modulation DAC, the output spectrum results in the red curve. The amplitude of the far-off spectral parts is further decreased in comparison to the spectrum considering the phase transfer function.

The OFDM simulation test cases are good examples to shown that the simulation framework is not only suitable for verification purpose but can also be used for an easy and fast design space exploration. Using a model-based approach a top-down design flow for complex SoCs is made possible.

7.2.3 Receiver Simulations

To check whether a communication between two *AixRF* chips is possible, a test case with two *AixRF* VPs, one serving as transmitter and the other as receiver, was simulated. Transmitting a 1 MSym/s GMSK signal, the spectrum is measured at the baseband Polyphase Filter (PPF) output (Fig. 7.12a). Due to the low-IF architecture the signal is centred around 1 MHz. In Fig. 7.12b the eye diagram at the RX frontend is shown. Although the eye is closed a little more than at the output of the CORDIC unit of the transmitter, the signal quality is still good enough to achieve a low Bit Error Rate (BER). After the transmission of 595 bit Pseudorandom Binary Sequence (PRBS) data, the BER after the digital modulator was measured to 0.005. The runtime for this test case was 93.6 s.



(a) Spectra.

(b) Eye diagram.

Figure 7.12: GMSK signal at the output of the baseband RX PPF.

In addition to the transmit-receive test case reciprocal mixing scenario is presented here which should demonstrate the usefulness of the spectral signal representation on the one hand and the seamless interaction between *XREAL* and spectral signals on the other hand. The input signal of the 2.4 GHz RX path is composed of two 1 MSym/s GMSK modulated RF carriers. The wanted receive signal has an amplitude of 50 µV and is centred around 2.629 GHz[3] while the blocker signal has an amplitude of 3 mV and an offset frequency of 28 MHz or 32 MHz, respectively.

In multiple BER simulations with each 640 bit the spur amplitude of the PLL output signal was gradually increased. This was achieved by artificially adding a small offset current on the CP's output. By this, the activity of the PLL is increased and the spur level rises. In Fig. 7.13 the BERs as function of the measured spur level for both cases are plotted.

Since the reference frequency of the PLL is 32 MHz, the resulting spurs are also 32 MHz away from the main tone and thus, spectral parts of the input signal with an offset frequency of 32 MHz from the wanted receive signal are convolved into the BB. This effect is called reciprocal mixing. As can be seen, the BER of the test case in which the blocker signal is exactly 32 MHz away from the wanted signal is constantly rising with increasing spur level while in the other case where the blocker is at an offset frequency of 28 MHz there is no trend in BER development

[3]The frequency was chosen to the out of band frequency 2.629 GHz because the maximum gain of the LNA was found in this region after schematic simulation. With parasitic layout effects, the centre frequency will shift towards 2.4 GHz in real operations.

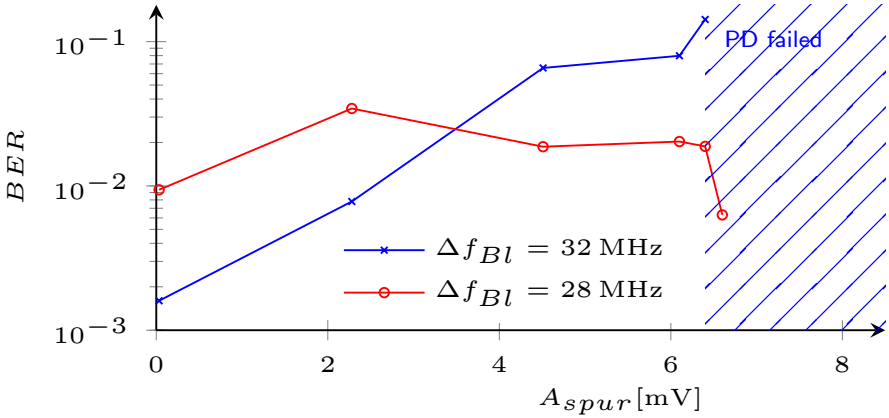


Figure 7.13: Effects of reciprocal mixing on BER.

recognizable although the blocker is closer to the wanted receive frequency. This can be explained by the fact that in the first case both the blocker signal and the wanted input signal are downconverted to the same IF and the demodulation process is disturbed.

7.2.4 Simulation Performance

To show the superiority of the developed RF event-driven simulation methodologies with regards to conventional SPICE or Spectre transistor-level simulations for large mixed-signal systems, the simulation speed for an RX, a PLL and a PA test case are compared in table 7.3. The first line of the table gives the CPU computing time for the transistor-level simulation on a matrix-solving simulator and the second line gives the time needed for the event-driven simulation. Hereby, the first number stands for the computing time and the second number gives the simulated time. The speed-up is the acceleration gained through the simulation on a digital simulator and the accuracy is a measure for the derivation of the event-driven simulated waveforms from the transistor-level simulation (1 means a perfect match). It can be seen that the speed-up ranges from 5000 to nearly 67000 while the accuracy of the event-driven approach remains high. By using event-driven models the simulation of complex system-level verification and system exploration test cases is finished within a matter of seconds or minutes instead of hours or days.

Table 7.3: Comparison of simulation speed.

	RX frontend	PLL	800 MHz PA[a]
Transistor level	33.7 ks/10 µs	112 ks/50 µs	13.6 ks/5 µs
Ev.-driven	670.3 s/1 ms	101 s/3 ms	193.3 s/1 ms
Speed-up	5028	66532	14071
Accuracy	0.982	0.996	-

[a] The transistor-level PA simulation was done using only the final PA stage without PLL or digital part while in the event-driven simulation the complete digital part, the PLL, the power and the clock management was included. Therefore, the significance of the speed up value is not so meaningful and the accuracy could not be calculated.

7.3 Receiver Noise Simulation

The influence of noise on the receiver chain was examined using the system-level noise analysis method [104] presented in chap. 6. Three test cases show different scenarios including the propagation of supply noise through the receiver. In the first test case, the input signal has an amplitude of 50 µV and a frequency of 2.628 GHz. The LO signal coming from the PLL is set to 2.627 GHz and is disturbed by a 4.5 mV spurious tone at 32 MHz offset frequency. A supply noise of 50 µV/$\sqrt{\text{Hz}}$ was assumed. The LNA output noise and the VCO noise were extracted from SpectreRF circuit simulations. In Fig. 7.14 - Fig. 7.16 the noise analysis results of the major noise components are plotted at different phases of the analysis and different measurement points in the DUT.

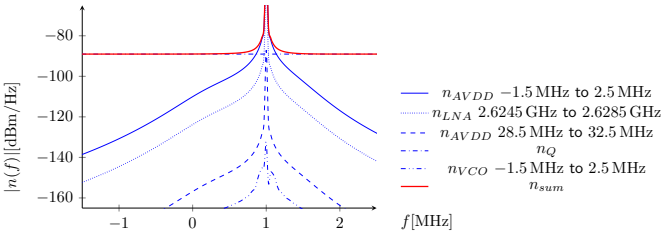


Figure 7.14: Noise at the input of the ADC without considering the feedback.

Fig. 7.14 shows the noise spectrum at the input of the comparators of the ΔΣ-ADC of the receiver frontend without considering the feedback structure via the DACs. Due to the pole of the polyphase feedback filter in the ADC, the noise around 1 MHz is increased tremendously. In addition to the LNA noise, the VCO noise and the quantization noise n_Q of the comparators, PSN is found at the output. The supply

noise n_{AVDD} is mixed upwards by the nonlinearity of the LNA to a frequency band around the receive channel. In the next step, it is downconverted by the LO to BB around 1 MHz. Furthermore, the PLL spur at 32 MHz offset frequency causes reciprocal mixing of the mixer input noise. Therefore, not only the PSN centred around 1 MHz is found at the receiver output but also supply noise around 32 MHz which is first upconverted in the LNA and then downconverted by the LO spur. Since the digital main clock has a frequency of 32 MHz and thus, the switching noise is increased in this band, it is crucial to shield the LNA from PSN from the digital part.

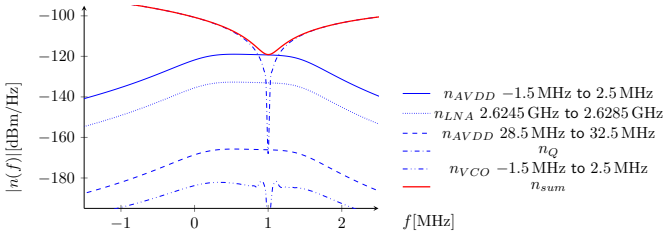


Figure 7.15: Noise at the input of the ADC considering the feedback.

In Fig. 7.15, the result from the post-processing phase of the noise analysis is shown. The same components as in Fig. 7.14 are regarded but now with closed feedback. It can be seen that the out-of-band noise is dominated by the shaped quantization noise which has highpass character while the other components are lowpass shaped. Because of the closed loop in the PLL, the low frequency VCO noise is also suppressed.

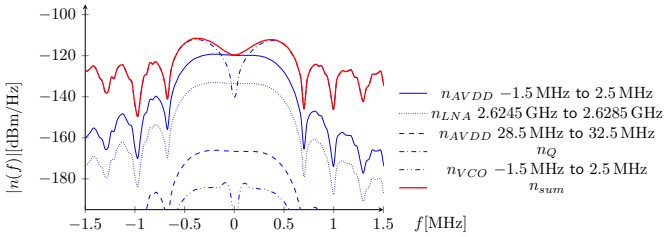


Figure 7.16: Noise at the input of the demodulator.

In Fig. 7.16, the noise at the input of the demodulator is shown. Through the downconversion by the digital mixer, the noise is shifted down by 1 MHz and the following digital filter dampens the noise outside of the receive channel. By integration over

frequency, the total noise and thus, an Signal-to-Noise Ratio (SNR) of 17.63 dB is calculated. This estimation can be used to calculate the BER of the receiver chain.

In two other scenarios, the influence of blockers at 2 MHz and 32 MHz offset frequency and amplitudes of 3 mV and 300 µV, respectively, are examined. In these test cases, the PLL spur amplitudes are negligibly low while the other simulation conditions stay the same as in the first example. As shown in Fig. 7.17 and Fig. 7.18, the supply noise originated in the frequency bands around 2 MHz and 32 MHz, resp., gets folded to BB at the comparator input of the ΔΣ-ADC. In addition to the LNA intrinsic noise, in the 2 MHz blocker case the nearby VCO noise plays a more important role than in the other examples because due to the high blocker level, more phase noise gets converted into the receive band.

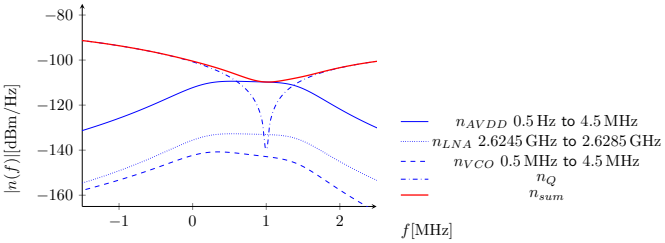


Figure 7.17: Influence of a blocker.

In Fig. 7.19 it can be seen that with rising blocker level power at 32 MHz, the spectral density of the supply noise components coming from the frequency band centred around 32 MHz are constantly increasing while the noise coming from the band around DC stays constant. For system design or system verification this analysis is useful to select the components' specifications or to find the critical points in the DUT where the system must be improved. The feedthrough from the supply to the output of the LNA is a major contributor to the output noise. To achieve a better SNR, one way would be to decrease the sensitivity of the amplifier with regards to PSN.

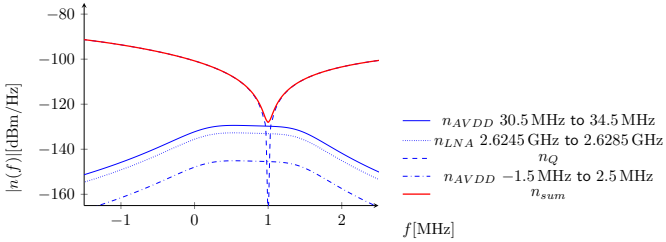


Figure 7.18: Second blocker scenario.

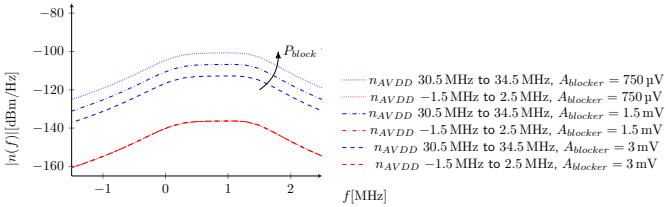


Figure 7.19: Influence of the blocker level on the output noise of the ADC.

CHAPTER 8

SNEAK PATH ANALYSIS

The degree of electrification in vehicles is rising constantly which makes the wire-harness design one of the most complex and important tasks in the design process. The total length of all wires in modern cars can be up to 3 km and the weight can exceed 80 kg [7]. With rising degree of electrification and Advanced Driver Assistance Systems (ADAS) systems, the complexity of the harness gives serious issues in the design and verification process. The need for fast and error free development of the electrical system is crucial for getting the vehicle to the market as fast as possible. Due to the enormous number of components and different configurations of the wire harness it is a nearly impossible task to design an error free circuit first time right. Therefore, analysis methods are needed to find and fix the design errors.

A special kind of design errors that are hard to detect, but can lead to disastrous behaviour of the system are sneak circuits.

> *Sneak circuits are unintended paths in a network that can cause undesirable actions. Sneak-Circuit Analysis (SCA) is the procedure for avoiding these paths or detecting them.* [107]

As the quote says, the main concern of a SCA is to identify circuits and conditions that cause undesired behaviour of the system without a component having failed. An infamous example is the failure of the firing circuit of the Mercury-Redstone rocket [108] which was launched in 1961. Using this circuit depicted in Fig. 8.1, the problems of sneak circuits and the need for an efficient SCA is pointed out.

The Redstone rocket had launched successfully 60 times until 21 November 1961. The firing circuit works as follows: The motor is ignited by the on-board fire switch. Once the switch is closed, a relay latches and continues to supply the engine with electricity. A second switch can abort the launching process which also latches a cut-off relay. In the ground control centre, an indicator bulb signals the actuation of the 'fire' switch. The external cut-off switch is intended to enable the operator to stop the launching process from the outside in case of an emergency. On the day of the incident, the 'fire' switch was activated, the 'fire' relay was latched and the engine started. The rocket lifted off only a few centimetres before the connectors to

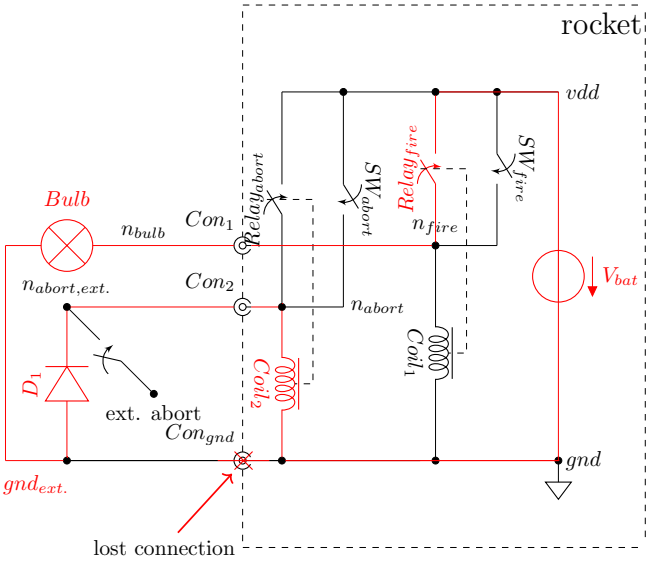


Figure 8.1: (Simplified) Mercury-Redstone firing circuit.

the external control broke. Unluckily, the first connection to break was the ground connector. A sneak path via the 'fire' relay, the indicator bulk and the diode to the 'cut-off' relay was established. The 'cut-off' relay closed and the launch process was aborted. Luckily, no one was harmed in this accident, but the failure postponed the start of the rocket until the error was found and fixed and the resulting costs were high. Of course, this behaviour was not intended and nobody thought about this special case in advance. With proper SCA this failure could have been avoided.

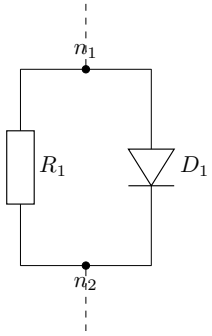
Standard sneak analysis procedures are costly from a time, money, and personnel perspective. The processing of design data available only during the latter portions of the development cycle are highly labour intensive and create difficulties in instituting a designing automated techniques for sneak analysis. Automated SCA methods have been developed by several contractors, but are either not completed or are proprietary. One brute force approach for SCA analysis of simulating all possible scenarios and checking the result for unwanted behaviour is presented in [109], but the number of simulations increases exponentially with the number of switching elements which limits the procedure to small designs. Other analysis methods use recognition of critical topologies together with a clue list of patterns which could lead to sneak circuits [108][110], but this requires a complete and good clue list.

8.1 SCA Operating Principle

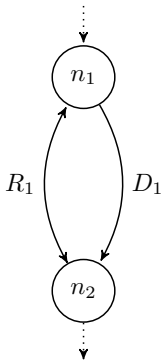
A SCA approach based on path search in graphs was developed in the course of the Ford Alliance Project between the electrical verification department located at Ford's development centre in Cologne and the IAS. Opposite to brute force simulation approaches, this approach can be regarded as formal verification method since it tries to prove the absence of unwanted current paths in the design.

The first step in the development phase of the method is to exactly define a sneak circuit and to make a few assumptions which applies the harness designs in cars and simplifies the analysis enormously. The assumptions and definitions made are listed below:

1. It is assumed that there is only one wanted path to a certain component. This path is characterized by the fact that it has the smallest resistance. Every other path is a possible sneak path.
2. Current is only flowing out and not in of the highest potential.
3. Current is only flowing in and not out of the lowest potential.



(a) Diode and resistor in parallel.



(b) Graph representation of the subcircuit. Since the resistor conducts currents in both directions, it is represented as bidirectional edge while the diode is represented as unidirectional edge.

Figure 8.2: Circuit elements in graph.

Every schematic can also be represented as a graph. The circuit nodes are the nodes in the graph and the components between the nodes equals the edges in the

graph. For a proper representation of the circuit, multiple edges between two nodes must be allowed because there might be devices in parallel. Furthermore, it can be differentiated between unidirectional elements like diodes and elements which allow bidirectional current flow like resistors, coils, etc. Therefore, directed edges are used for the circuit representation. A data structure which supports this is called a MultiDiGraph. An example of how to represent a circuit as a MultiDiGraph is shown in Fig. 8.2.

A diode and a resistor are placed in parallel between the nodes n_1 and n_2. The diode allows a current flow only from n_1 to n_2 which is represented by the directed edge D_1. The resistor allows a current flow in both directions which is indicated by the edge R_1 with arrows pointing in both directions.

Instead of storing only the information that there is an edge between two points, a weight of this edge can be provided as well. With regard to this application, it makes sense to use the resistance of the device as weight. In case of nonlinear elements like diodes an average resistance can be chosen. A switch can take a low or a high resistance value depending on whether it is closed or open.

The idea of the developed SCA algorithm is to find multiple paths from the positive supply pin to the negative supply pin via a certain component/load. If more than one path is found, the circuit contains possible sneak path because it was assumed that there is always only one path to a certain component. The Dijkstra algorithm [111] is used for finding the path with the lowest weight. As discussed before the component's resistance is used as edge weights. The SCA algorithm starts searching the shortest path from the positive supply pin to the component and from there to the negative supply pin. In the beginning, it is assumed that all switches are closed regardless of whether it is a sensible setup or not. After the algorithm found the shortest path via the component, it checks whether the found path contains a switch. If so, it is opened and the SCA routine tries to find a new shortest path. If no other path can be found, there is no sneak path. If a second path is found, it is stored as possible sneak path and the search for other sneak paths will be continued by testing all other combinations of opened and closed switches on the already found paths. By doing so, it is assured that every possible sneak path is found and not only the one with the lowest weight. Due to using a shortest path algorithm, the found possible sneaks are sorted by the path resistance which could give a hint on how severe or probable the sneak is. Moreover, the SCA is stopped if a certain path resistance value is exceeded because any additional path that is found has such a high resistance that it is uncritical. A conservative choice for the critical path weight is for example the resistance of an open switch. If the algorithm only finds paths which have higher resistances than this threshold, it is safe to assume that there are no more sneak paths to this special load in the DUT.

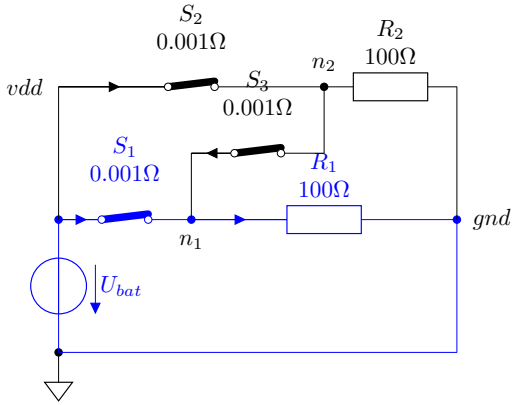
Since the Dijkstra algorithm has a runtime in the Landauer notation of $O(|edges| + |nodes| \cdot log(|nodes|))$ [112], the size of the design determines the runtime of the SCA. In addition to that, the speed of the algorithm depends on the number of critical loads that have to be checked and the number of switches which are found on the paths.

A short example on how the SCA algorithm works is presented using the circuit shown in Fig. 8.3a. All sneak paths to the critical load R_1 should be found. The algorithm starts with all switched closed and finds the shortest path from the positive supply pin (*vdd*) to the negative supply pin (*gnd*) via the load R_1 which includes the switch S_1. The algorithm regards this path marked in blue in Fig. 8.3b as wanted one which has a weight of 100.001 Ω. In the next step, the switch S_1 is opened and again a shortest path is searched. While S_1 is now high-ohmic (100 MΩ), a possible sneak path via S_2 and S_3 (marked in red) is found with a path weight of 100.002 Ω (see Fig. 8.3c and Fig. 8.3d). The path is saved and the switches on it are extracted. In the following step, the switch positions S_2 open and S_3 closed, S_3 open and S_2 closed and both S_3 and S_2 open are examined, but none of these combinations lead to a shortest path with the weight below the critical weight of an open switch resistance. Thus, the algorithm terminates and displays the one possible sneak path to the user which have to decide, whether this path is intended or not.

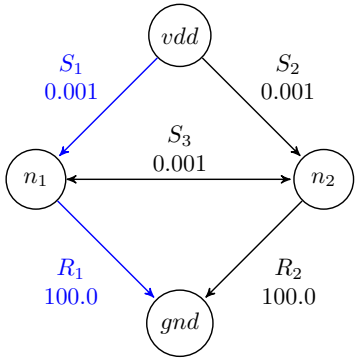
The back end of the SCA was implemented in Python using the *networkx* package [113] which supports representations of graphs and operations on them including the Dijkstra algorithm. Python was chosen for several reasons: First, it is open source which makes it free to use. Furthermore, it provides a large amount of packages which makes it well-suited for rapid prototyping. Moreover, the language is easy to learn, understand and debug. Last, the used package for path search is optimized for performance and the execution speed was sufficient for this application.

The application starts with reading in a netlist in the Synopsys Saber® MAST-format of the DUT. The netlist is parsed and converted in a MultiDiGraph representation. All circuit components are categorized either in *switchables* which are elements that can be switched on or off, loads, sources, unidirectional and bidirectional elements or wires. Furthermore, the user has the possibility to define more components out of the basic elements by writing a description in the MAST-netlist format. An exemplarily definition file of an NPN-transistor is given in Listing 8.1. Since the SCA does not aim for performance analysis, the transistor is coarsely described as diode between base and emitter and switch between collector and emitter.

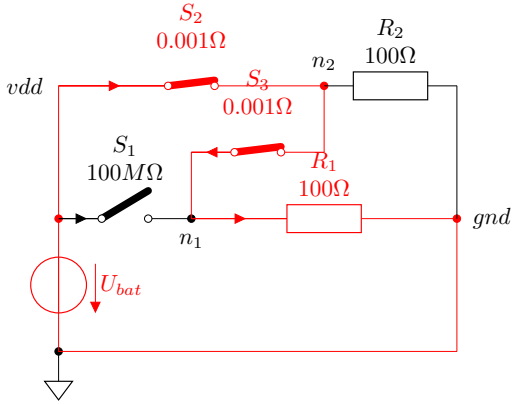
The developed parser is also able to understands hierarchical netlists by recursively calling itself. In the parsing step the design is automatically flattened. If components appear that are new to the parser, it will warn the user and will automatically create a MAST template in which all pins of the component are listed. The user then has



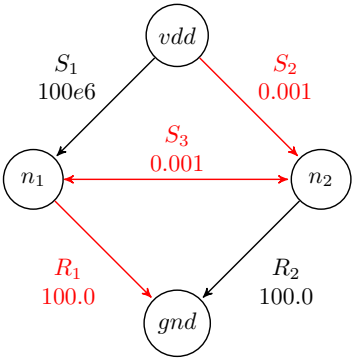
(a) Wanted path.



(b) Wanted path.



(c) Sneak path.



(d) Sneak path.

Figure 8.3: Example on how the SCA works.

```
template npn b,c,e
{
d.dbe p:b n:e    #base-emitter diode
sw.sce m:e p:c #collector-emitter, channel as switch
}
```

Listing 8.1: Description of NPN-transistor.

the possibility to describe the connectivity between the different pins and can repeat the analysis. After parsing, to simplify the graph and to speed up the analysis, nodes which are connected with wires are merged to one node. Exceptions can be defined by the user for critical wires which may break. These wires are then considered as *switchables* and will be treated the same way as conventional switches.

Once the netlist is converted into a graph, the positive and negative supply nodes of all possible loads are identified. In the next step, for each load the paths with minimum weight from the positive supply pin of the voltage source to the load and from there to the negative supply pin of the source is searched using the Dijkstra algorithm. The found paths are identified as 'wanted' paths. If this path contains no *switchables*, the load is always connected to the supply with low resistance and for this reason it makes no sense to continue searching for sneak paths. In the case that there are *switchables* on the path, the SCA is continued as explained above. The program stops when every load is examined and for every load the direct path plus possible sneak paths are found. A flowchart of the process is provided in Fig. 8.4 for better understanding.

The developed Python backend was integrated in the Electronic Computer Aided Design (ECAD) software E3.cable® from Zuken used for designing and documenting cable plans and harness layouts. In the E3® series the possibility to develop and run customized Visual Basic Script (VBS) exists. Furthermore, specialized VBS functions allow to interact with the designs in E3®. A script was written which creates a Saber® netlist out of a E3® wiring plan and then starts the SCA backend afterwards. The VBS script waits for the algorithm to finish and reads in the results. Direct and sneak paths are then highlighted in E3.cable®, allowing the user to get a direct visual feedback of the SCA.

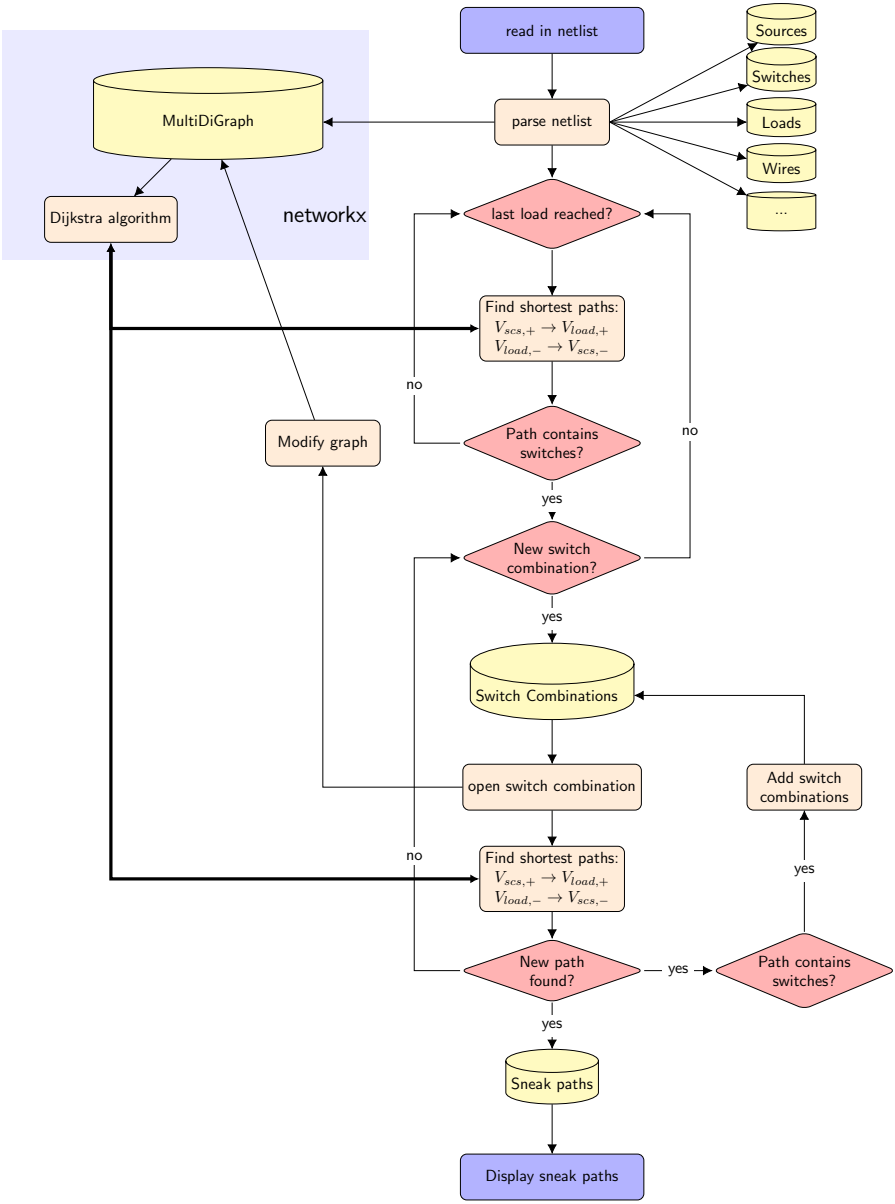


Figure 8.4: Flowchart of application.

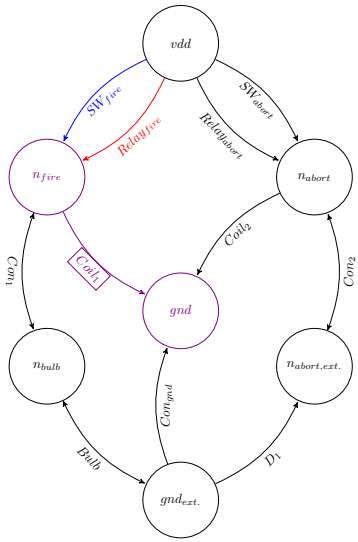
8.2 Application Examples

Exemplarily, the SCA results of two different designs are discussed. For testing the developed application, the Mercury-Redstone firing circuit as presented in Fig. 8.1 was chosen as DUT. It was built up in SaberRD® and the netlist was used as input for the algorithm. The coils of the fire relay and the abort relay and the light bulb were automatically identified as critical loads by the tool. Fig. 8.5a - Fig. 8.5d show four sneak paths found by the analysis represented as graph. The wanted paths are coloured in blue whereas the sneak paths are marked in red. Components that are both part of the wanted and the sneak path are painted in violet. Sneak path 1, 2 and 4 can be easily discarded because they are found due to parallel switches (fire switch || switch of fire relay and abort switch || switch of abort relay), which are, according to the definitions, possible sneak paths, but will not cause problems in real life. Sneak path no. 3 is the sneak path which caused the abortion of the launch process.

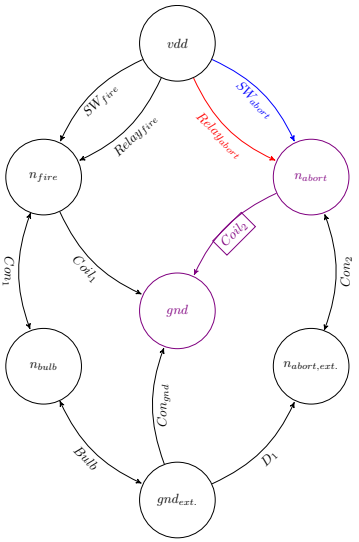
The second test case discussed here is an interior lighting circuit from a car, but due to confidentiality reasons it will not be revealed from which model. The circuitry is shown in Fig. 8.6. Several light bulbs can be switched on either by a switch or remote by a steering signal which switches on a transistor or by the combination of both. The inside of modules A_1 - A_6 is not revealed, only the connectivity between the pins is described by a MAST netlist (Listing 8.2). It is known that the bottom wire can break off. Thus, this wire is marked so that the application treats the wire as *switchable* and does not remove it in the optimization step.

The schematic of the circuit was drawn in Zuken E3® and the SCA was called from the E3® GUI. Sneak circuits were found when the common ground connection was separated from the main ground due to the loose cable W_{bottom}, including an already known one which causes X_2 to shine even if M_2 is switched off. The current path shown in Fig. 8.7 leads via transistor M_1, switch SW_3, light bulb X_3 and module A_1 to X_2.

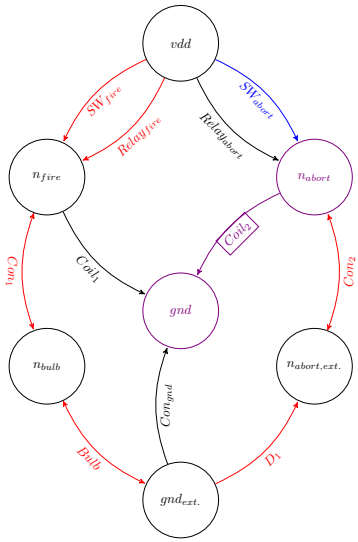
After the Python backend algorithm has been started from Zuken E3® and has finished finding the sneak paths, both the direct and the sneak paths are coloured; the wanted path in blue and the sneak path in violet. If more than one sneak path is found, the SCA frontend in E3® allows to switch through the different results by a keyboard combination. The checking for sneak paths and the visual feedback in the design environment gives the designer an easy to execute and comprehensible feedback.



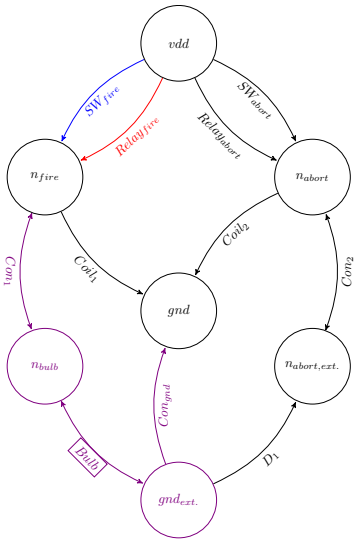
(a) Sneak path 1.



(b) Sneak path 2.



(c) Sneak path 3.



(d) Sneak path 4.

Figure 8.5: Sneak paths found in the firing circuit by the developed SCA.

Both analysis runs took about a second. Unfortunately, a larger circuit could not be provided to test the performance on a more complex system. It would be interesting to examine the runtime for different circuit sizes.

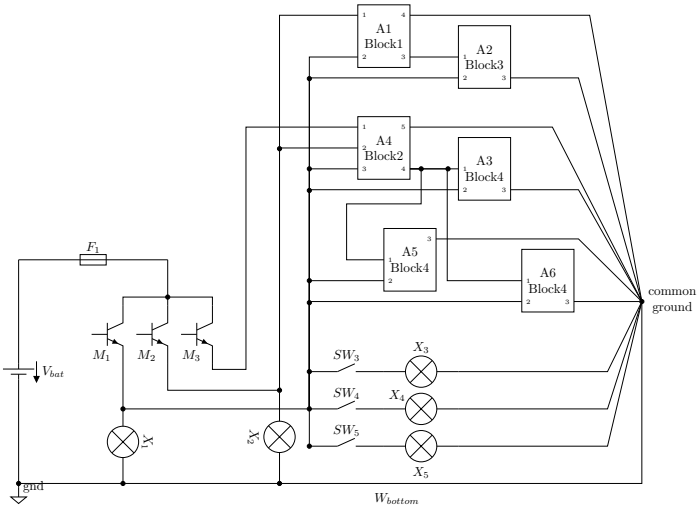


Figure 8.6: Indoor lighting circuit in a car.

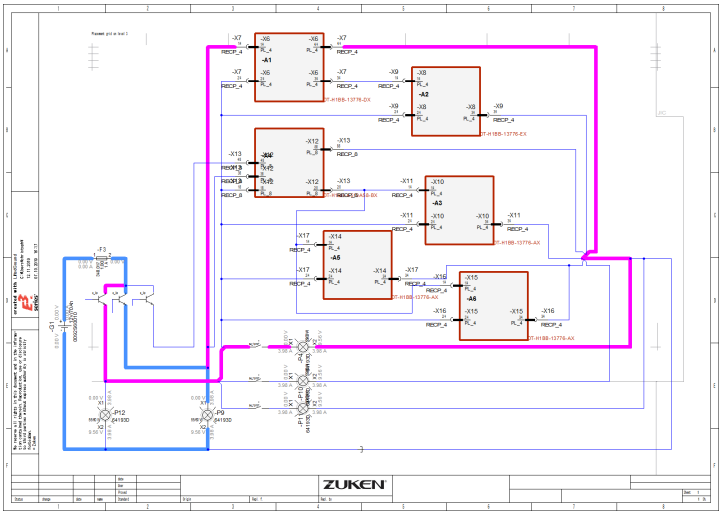


Figure 8.7: Example of found sneak circuit in a car.

```
template Block1 1,2,3,4
{
r.14 p:1 m:4 = rnom = 100.0
}

template Block2 1,2,3,4,5
{
r.r25 p:2 m:5 = rnom = 100.0
}

template Block3 1,2,3
{
}

template Block4 1,2,3
{
}
```

Listing 8.2: Connectivity of modules.

CHAPTER 9

WIRE HARNESS OPTIMIZATION

The increasing need for optimized, reliable and secure components of a car demands for highly efficient and fast development processes to be competitive under the time-to-market pressure. Manual design steps are still widely common but with increasing complexity of the systems, solely manual processes are too slow and error prone for today's automotive market. Certain repeating tasks like optimization of a given wire harness topology can be efficiently managed by a computer program. The advantage of the automation of an optimization step is that it accelerates the design process because during the time the computer runs the optimization, the engineer can take care of other tasks. Furthermore, a computer algorithm always executes every step that was programmed and does not forget to check relevant design criteria. This guarantees the functionality and safety of the components which is the top priority in the automotive market.

Typical optimization goals regarding wire harnesses are the minimization of power loss, weight and the bill of materials. Since it is hard to decide which of these goals is the most important one, it is sensible to search for Pareto optimal solutions and pick the one of the found solutions which suits best. Moreover, different Pareto optimal solutions can be compared against each other. In addition to the minimization problem, the target designs need to fulfil several constraints and requirements, e.g., the wire temperature must stay under the critical value, the voltage at the different components needs to be high enough, the fuses must response fast enough in case of a failure, etc.

The here presented optimization tool focusses only on the optimization of the different cable diameters and wire types. The length and the topology of the wire harness are already specified as well as the electrical components supplied by the harness. The tool uses circuit simulation to calculate the power loss in the design and to verify the constraints. Thus, it is a good example on how automated simulation can accelerate the design process and therefore it was included in this work.

An overview about several optimization algorithms used for the optimization of a power supply system can be found in [114]). Gradient based search algorithm cannot be used because they apply on continuous problems and the available cable

diameters and types can only be chosen from a discrete set. Due to the extremely large search space and the high number of dimensions, only stochastic, meta-heuristic algorithms are considered for solving the problem [114]. There are numerous of these algorithms like Tabu Search, Simulate Annealing, Evolutionary Algorithms and Swarm Intelligence which are more or less suited for this optimization problem. In case of the Tabu Search, it is expected that the progress of the algorithm is too undirected and no result can be achieved in acceptable time. Furthermore, Simulated Annealing has a comparatively slow convergence speed for a reliable finding of the optimum [114]. Thus, both algorithms are not further considered for the wire harness optimization. Evolutionary algorithms and Swarm intelligence both have their advantages and disadvantages. Exemplary, a Genetic Algorithm (GA) from the class of evolutionary algorithms and a Particle Swarm Optimization (PSO) from the class of Swarm Intelligence are compared. There are problems [115] in which the GA has a better performance than the PSO and papers [114] in which the PSO is preferred to the GA. The choice finally fell on the GA since the optimization tool is based on a software developed in a master thesis [116] in which the GA was used and because MATLAB® provides a multiobjective GA algorithm.

For better understanding, the GA algorithm is shortly described. Afterwards, the chapter deals with the development of the tool and how the software is partitioned. Finally, some obtained results are presented.

9.1 Genetic Algorithm

The GA was first discovered as a result of the research of John Holland around 1960 [117] and is loosely based on the theory of evolution of Charles Darwin. In this, the biological evolution of living beings is described as an adaptation process to changing environmental conditions. The basic idea is that within a population only the fittest individuals survive. The surviving individuals multiply among themselves, so that the genetic characteristics of the parent pair are recombined and transferred to their children. Within the recombination, however, errors can occur, so that individual genes are not solely determined by the mixing of the parent genes, but also by the coincidence of nature which is called mutation. Thus, the genes of a population change from one generation to the next. The GA transforms the theory of evolution into an abstract sequence of defined steps, which have to be adapted to the specific problem. In the following, a short explanation for every step will be given.

9.1.1 Definitions

When speaking about GA a few word definitions must be explained. Since the idea comes from the biological evolution theory, the wording is based on its termini. In GA, a *chromosome* is a set of *genes* which define a proposed solution to the optimization problem. All these solutions together form a population. The *chromosomes* can be represented as string of bits, integers, chars, etc. depending on the problem formulation.

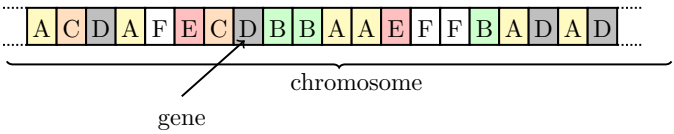


Figure 9.1: Chromosome and genes.

9.1.2 Initialization

In the first step, an initial population is generated. A population is a set of different possible solutions for the optimization problem. The number of individuals in the population is one parameter of the GA. Usually, the population contains several tens to thousands of individuals. Often, the initial individuals are generated randomly, which allows the entire range of possible solutions in the search space.

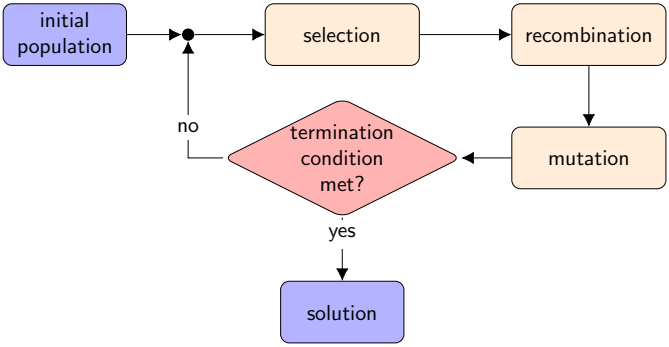


Figure 9.2: Genetic algorithm.

9.1.3 Selection

After each new generation creation, a part of this population is selected to produce a new generation. The quality of the solutions is measured by a fitness function and the fittest individuals are most likely to be selected. A few less fitting solutions are also chosen for breeding to ensure genetic diversity within the genetic pool. The fitness function might be a mathematical expression or the fitness can be determined through evaluating simulation results.

9.1.4 Recombination

The next step in the generation of a new population from the selected individuals is the recombination. For each new solution to be bred, *parent* solutions are selected for reproduction from the old generation. The *child* solutions share *genes* of the *parents*. For each new *child*, new parents are selected. The number of parents is not necessarily limited to two but can also be higher. The average fitness of these *children* will increase by this process because mostly only the individuals with the best *genes* have been chosen to reproduce.

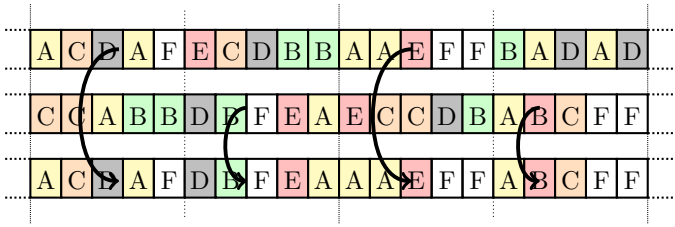


Figure 9.3: Recombination process.

9.1.5 Mutation

Mutation is used to maintain genetic diversity from one generation to the next and is analogous to the process of biological mutation. Mutation changes one or more gene values in an individual from its original state. In the GA the user can define a mutation probability for the genes to be altered. This parameter is a key value in the GA and is usually kept low since if it is set too high, the search will turn into a random search. The goal of mutation is to avoid local minima by preventing the individuals of a population from becoming too similar to each other.

9.1.6 Termination

The selection, recombination and mutation process is repeated until a termination condition has been reached. A terminating condition is reached if a solution is found that satisfies the criteria defined, a maximum number of generations reached or iterations no longer produce better results.

9.2 Optimization Tool

In this section, the tool requirements and details of the optimization process are discussed. Since the application should optimize already existing topologies, it needs to import them in a common format. The Saber® netlist format was chosen because it is widely used. Furthermore, the case may occur that not the complete system is to be optimized, but only some cables. For example, if a subsystem has already successfully been implemented in previous car models, it should be preserved as it is. In addition, the time required to find the optimum is reduced by restricting the search space. Therefore, the tool offers the possibility to select the wires which should be optimized and leaves all other untouched. Other requirements are user-friendliness, comfortable settings of the constraints and an easy-to-understand result presentation.

To reduce implementation effort, the application uses the *gamultiobj* algorithm from the MATLAB® optimization toolbox. It allows a minimization of multiple functions at the same time delivering the Pareto front by using a GA. For the Wire Harness Optimization (WHO), the costs, the weight and the loss in the cables needs to be minimized. With the results from the tool, the user can choose a design on the Pareto front and decide which trade-off between costs, power loss or weight is most beneficial regarding the complete system. Before the MATLAB® GA implementation is called, the tool needs to setup the simulation environment and the user can set the parameters for the algorithm. A flowchart describing the complete program overview is shown in fig. 9.4.

Since the main optimization application and the simulation software used to calculate the power loss and the electrical constraints are two different processes, a method for Inter-Process communication (IPC) is needed. It was chosen to share the necessary information via files, because of its simple setup. In the first step of the optimization process, the name of the DUT and the simulation settings are written to the files *TestbenchName.txt* and *'SimulationSettings.txt.* The information is later used in the SaberRD® simulator. Next, the selected wires which should be optimized are written to a file named *Wires2Change.txt.* In the following step, the GA options are defined

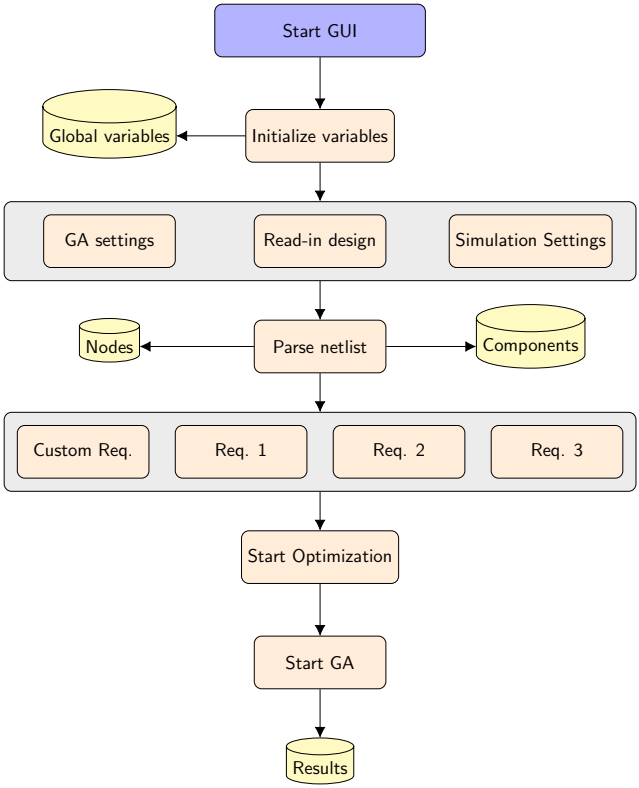


Figure 9.4: Overview of application.

and old files from previous optimization runs are deleted. After this, the SaberRD® simulator is started. Since the licence checking takes some time, the MATLAB® application waits before proceeding until the simulator has responded by creating the *SaberInitialized.txt* file. After finishing the initial setup, the MATLAB® GA implementation is called. This MATLAB® algorithm maps the population variable in the first step to a cable type and its diameter. Invalid combinations are found out and marked as not simulatable pairs which are later dismissed by the simulator. To avoid repeated checkout of Saber® licenses before every new simulation, a *tcl* script was written which puts the simulator in idle mode if there is currently no task instead of exiting and recalling Saber® for every new task. Since for an optimization process several thousand simulations may be started and the licence checkout process takes a few seconds, a huge amount of time is saved this way. The control of the simulator is described in detail in section 9.2.4.

After finishing, MATLAB® tells the SaberRD® to close by creating the *MatlabFinished.txt* file and displays the results. The MATLAB® GA algorithm supports the computation of fitness and nonlinear constraint functions in parallel. Therefore, all MATLAB® functions and all *tcl*-scripts on SaberRD® side supports a vectorized handover of the variables. A flowchart of the optimization process is shown in Fig. 9.5.

9.2.1 Price and Weight Calculation

The cost and the weight function of the wire harness is directly calculated in MATLAB® using the length, the type and the diameter of the cables. The exact equations are given in [116]. The functions accept vectors of wire type/diameter pairs as input parameter and deliver vectors of the calculated prices and weights.

9.2.2 Power Loss Calculation

The power loss function is a MATLAB® routine which starts a SaberRD® simulation to calculate the cable power loss of the DUT. In the first step, the changed wire parameters are written to a file. In the next step the *saberSimulate.txt* file with a '0' flag as body is created telling the simulator that the cable loss instead of a constraint should be run. Furthermore, the *SaberReady.txt* file from the previous run is deleted. The *tcl* script on simulator side notices the file creation and starts a transient simulation, reads-in the results and calculates the loss in the cables. The results are written in a file called *TotalCablePower.csv*. After the simulation is finished, the simulator creates the *SaberReady.txt* file, telling MATLAB® that the run has finished. The power loss function waits for this file, reads-in the results and uses them in the optimization step.

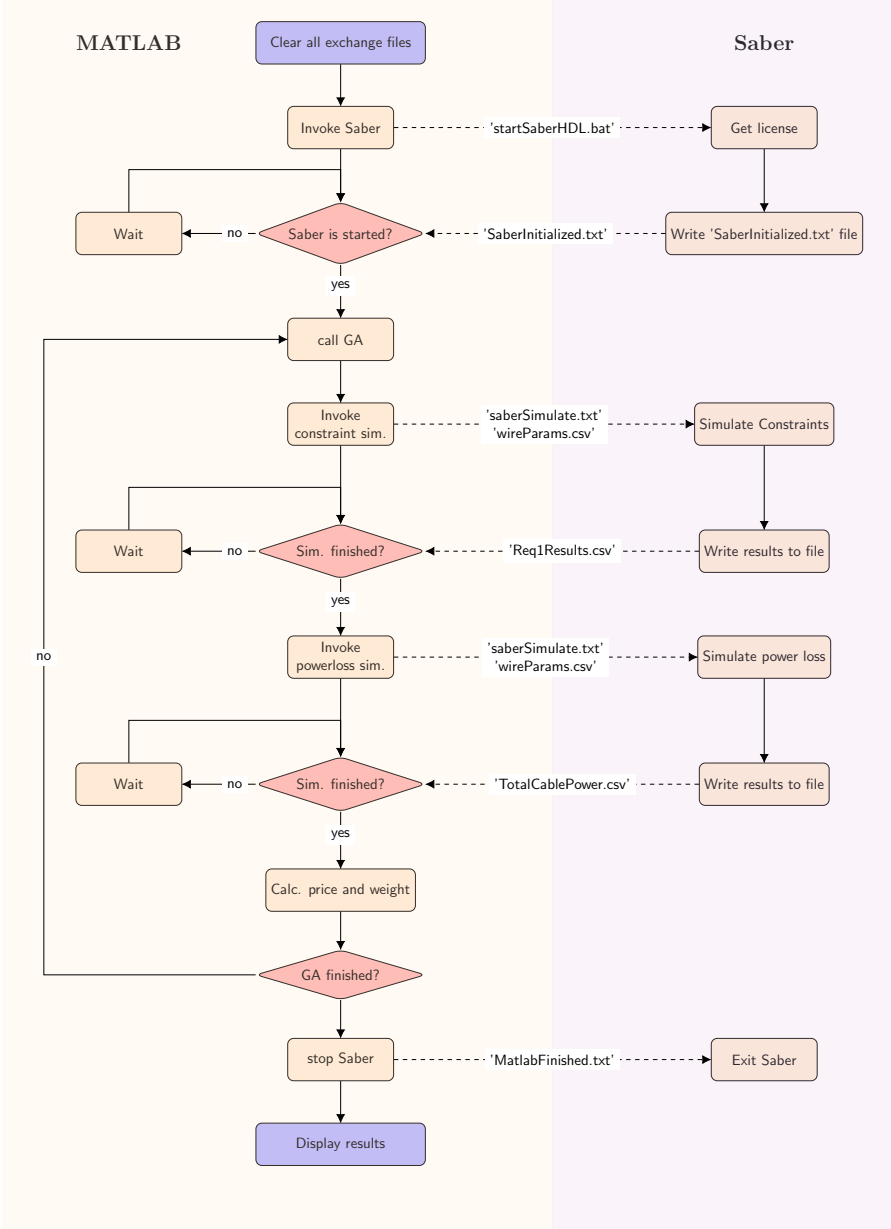


Figure 9.5: Overview of algorithm.

9.2.3 Constraints

The wire harness optimization is done under certain constraints such as the preservation of a minimum voltage level at the components. The constraint functions are set in the GA options. On MATLAB® side, the handling is similar to the power loss function. In the first step, the wire parameters to change are written to a file. Other settings e.g., the ambient temperature or the battery voltage are also handed over to the simulator. Now, the *saberSimulate.txt* file with the flag '1' is created telling the simulator that the requirements are simulated and the *SaberReady.txt* file from the previous run is deleted. The simulator starts the constraint simulations, which are defined in separate *tcl* files and reads-in the results. The results are written back in files. After the simulation is finished the simulator signals the MATLAB® application by the creation of the *SaberReady.txt* file that it has finished. After that the constraint function reads-in the results.

The three most used requirements are predefined and ready to use for the optimization process. But there is also a possibility to define custom requirements or constraints for a given DUT and use them in the WHO application. Custom requirements need to be defined as SaberRD® *Experiments*. *Experiments* are a predefined sequence of simulation and evaluation steps and are used to automate frequently performed simulation processes. An existing *Experiment* is stored in the *XML* format and can be loaded in the WHO GUI where it is converted into a *tcl* script. An example of the resulting *tcl* file for a custom requirement is shown in listing 9.1. The script runs a DC and transient fault analysis, measures the voltage at node n_{50} at 99 ms and checks whether the measured voltage is above 11.9 V.

9.2.4 SaberRD® Simulations

As mentioned above, the power loss calculations and the constraints are executed by the SaberRD® simulator. In the following section, the simulation process is described. SaberRD® is started in batch mode within the MATLAB® optimization process. After the licence checkout process the *tcl* script *TestSaberHDL.ai_scs* is executed. The script reads-in the files created on MATLAB® side and then creates the *SaberInitialized.txt* file telling MATLAB® that the SaberRD® simulator setup is finished. Since the request for a licence in the step taking the longest time, the SaberRD® session is started only once at the beginning of the optimization process and is not allowed to close until the WHO is finished. Therefore, the *tcl* script controlling the simulation process enters a loop and waits until it receives the *saberSimulate.txt* file or the *MatlabFinished.txt* file. If the *MatlabFinished.txt* file is found, the script exits and closes the simulator. In the other case the script reads-out the flag in the *saberSimulate.txt* file and either

```
proc customReq { runNumber } {
    set __customReqCheckVar 1
    saberhdl:fault -ffile TestWCA_fault_dir/Fault1.xml -body {
        dc -algstart dyn_ramp -dr_tsettle 20000 -algstepping no -dc_eppfile dc1
        tr -pfile tr1 -tend 100m -tstep 1n -siglist :...:*
        set tr1_file TestWCA.tr1.ai_awd
        set pfile_tr1 [pf:open $tr1_file]
        set wf1 [pf:read $pfile_tr1 :TestWCA:n_50]
        set meas1 [ Measure:At $wf1 99m ]
        set test1 [ expr { $meas1 > 11.9 } ]
        set __customReqCheckVar [ expr { $__customReqCheckVar && $test1 } ]
    }
    set customReqOutFileName Optimization/CustomRequirement/customReqOutFile
    append customReqOutFileName $runNumber ".csv"
    set customReqoutFile [open $customReqOutFileName w]
    puts $customReqoutFile $__customReqCheckVar
    close $customReqoutFile
};
```

Listing 9.1: Example of an automatically created *tcl* file for the custom requirement.

starts the power loss simulation or the constraint simulations. In the next step, the script reads-in the new parameters for the wires from a file and changes the wires accordingly. If an invalid wire combination is read-in, no simulation is started and a bad fitness value respectively a failed constraint is assumed. After finishing the simulations and writing the results to files, the simulator tells MATLAB® that the simulation results are available now by creating the file *SaberReady.txt*.

9.2.4.1 Simulation Error Handling

It might happen that the simulator is not capable to calculate an initial DC operating point or to finish the transient simulation. Therefore, each simulation command is embraced in an exception handling block. If the simulation fails, the script catches the error and can continue instead of abording due to the error. Furthermore, a special block (**sec_to_abort.sin**) is automatically inserted in the DUT which monitors the CPU time. If the timeout is exceeded, this block stops the simulation and throws an error. The script catches this error and continues with the next simulations. All faulty simulations are regarded as bad designs and therefore bad fitness values and failed requirement results are written back for those configurations. Furthermore, the application automatically logs some information about the optimization process and can also perform error handling in case something went wrong. The results of

the optimization process are printed to the *GA_Information.txt* file and in case of an error the tool writes an error log file *MatlabErrorLogFile.txt*.

```
Optimization terminated: maximum number of generations exceeded.
Optimization took 498.054879 seconds.
26 generations were simulated.
Number of function calls: 626.
```

Listing 9.2: Example of a log file.

```
2020-02-14 14:23:11,596 ERROR
Error during optimization - File: TotalCablePower.csv not found
```

Listing 9.3: Example of an error file.

9.2.5 Application Example

In Fig. 9.6, an example of a wire harness which should be optimized is shown. The battery voltage is set to 12 V and the design must fulfil two standard requirements. For each requirement, 2 fault cases needs to be regarded. In the initial design, the diameters of the cables were chosen large in order to be on the safe side regarding power loss and cable temperature. This results in a costly and heavy design.

The optimization of the design included the modification of the four wires (wire_v3_2, wire_v3_3, wire_v3_4, wire_v3_6) marked in red in Fig. 9.6. The DUT has to fulfil two standard requirements and one of the requirements demands for a simulation stop time of 1800 s and the coverage of two fault cases. The power loss was measured after a simulated time of 0.1 s and the maximum time step was chosen to 1 µs. The GA population size was set to 25 and in this test case 80 generations were examined. All in all, 6000 simulations were run. The computation time was measured to 39 min and 44 s on a laptop with i7-8650u core at 1.9 GHz and 19GB RAM. In comparison to a brute force approach which would have taken 113 million simulations, the GA approach needs only 0.0045 % of the simulations. Extrapolating this to the computational run time, a brute force optimization would need 611 days to test all possible combinations for the four selected wires.

Fig. 9.7 shows a screenshot of the GUI after the optimization runs has finished. In the left graph, the price and the power loss of the four wires is plotted for all simulated designs, but it is also possible to view weight vs. power loss or weight

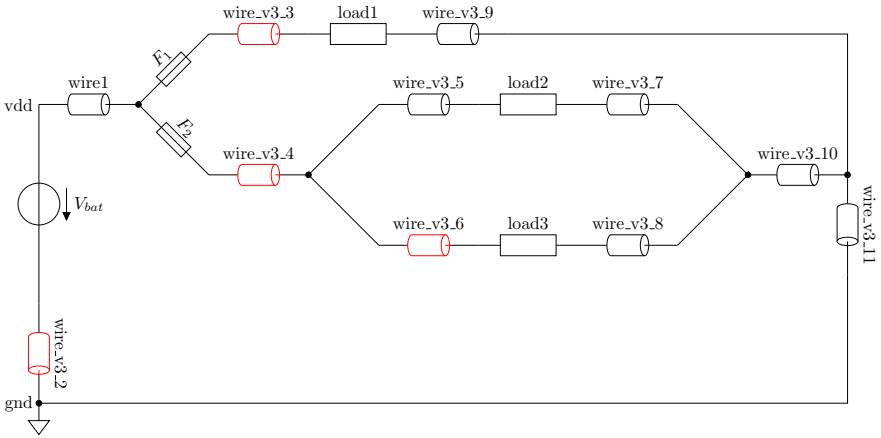


Figure 9.6: Example wire harness.

vs. price. All Pareto optimal solutions are marked by a red circle. The user has the possibility to select several designs by clicking on the points in the graph and view the associated wire parameters. It can now be chosen between optimal solutions depending on whether cost and weight or power loss is more important in the manufacturing process. In the right graph the cost development of the wires versus the GA generations is shown.

Table 9.1 compares the three marked Pareto optimal solutions with the initial one. It can be seen that even if the power loss of the initial design is the smallest one, the price and the weight are much higher than the values of the other solutions. Solutions no. 2 and no. 3 have only 20 mW respectively 70 mW more power loss but are more than 20 € and 40 € cheaper and the weight is remarkably lower. Design no. 3 sacrifices the low power loss for a low weight (below 1 kg) and low costs (4.12 €). Weight and cost of the wires have a high correlation due to the fact that the dominating factor for the price is the material. Thus, the more metal is needed, the heavier the cable gets and the more expensive it is. On the other hand, the thicker the wires get, the less power is lost in them. This anti-proportional behaviour is also visible in the graph, where all Pareto optimal points lie on a hyperbolic shaped curve.

Table 9.1: Different harness designs.

design	costs	weight	power loss
initial	68.20 €	10.16 kg	1.74 W
no. 1	44.32 €	8.54 kg	1.76 W
no. 2	23.78 €	4.54 kg	1.81 W
no. 3	4.12 €	0.87 kg	2.58 W

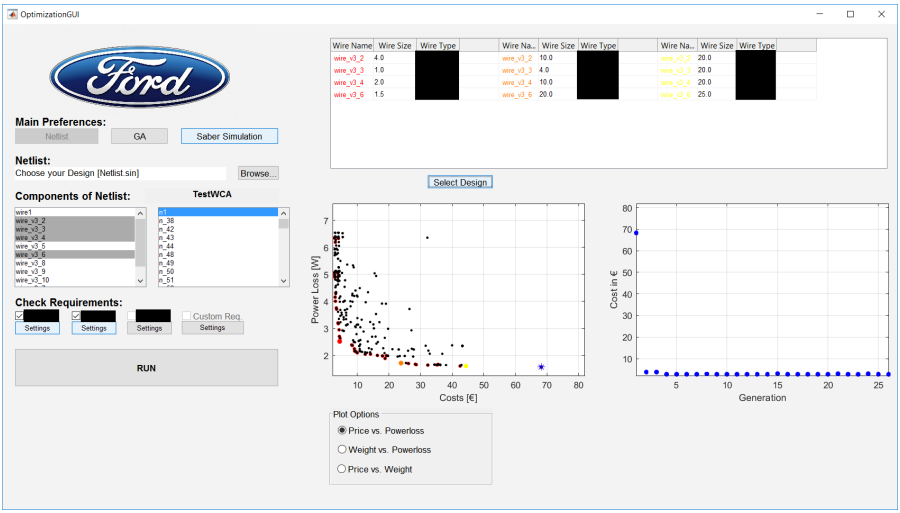


Figure 9.7: Interactive result presentation[4].

9.2.6 Outlook

Although it has been shown that the optimizer works, it is user-friendly and it has been optimized for speed, there are a few points that can still be changed in future versions. Using files for IPC is an easy and straightforward methodology but for further speed optimization, more sophisticated solutions like shared memory could be used. Moreover, a possibility to change the optimization engine from GA to another meta-heuristic algorithm could be implemented. It would be interesting to observe

[4]Due to confidentiality reasons the names of the requirements and the wire types were blackened.

which algorithm suits best for WHO. Last, the idea of the harness optimization could be transferred to the area of integrated circuits, where existing circuit topologies (current mirrors, Operational amplifiers (OPAMPs), VCOs, ...) could be optimized for performance and power consumption in a similar way wire harnesses are optimized for costs vs. power loss. A way to achieve this would be to redefine the cost function for another metric.

CHAPTER 10

CONCLUSION AND OUTLOOK

10.1 Conclusion

In this work analysis methods for two different application fields are discussed. The first part of the thesis deals with the simulation of complex integrated mixed-signal RF systems. The second section handles the analysis of and optimization of wire harnesses in the automotive field.

Since mixed-signal ICs are becoming more and more complex, traditional analog simulation tools like SPICE or Spectre are not suited anymore due to their superlinear runtime and memory requirements needed to solve the underlying equation systems. As a solution the analog part of the IC is modelled using RNM techniques and the complete system can be simulated using a digital, event-driven simulator whose runtime scales better with the system's size. Furthermore, events in the system are processed locally and only parts of the IC that are affected by signal changes needs to be recalculated. To overcome drawbacks regarding the modelling of analog circuits in conventional HDLs, an analog mixed-signal framework was developed which combines the advantages of SV as HDL like the possibility to express temporal processes and concurrencies, and C++ for signal processing. Due to the degrees of freedom in this approach, a way to speed-up simulations is the use of sophisticated signal descriptions. Depending on the circuit to model and its field of application, different ways to describe a signal are more or less beneficial. In the scope of this thesis, it was focussed on two different signal descriptions: Spectral signals and *XREAL* signals. Spectral signals are suitable for the description of modulated RF signals which are sparse in spectrum. Furthermore, they are beneficial when modelling nonlinearities and their effects like intermodulation or frequency conversion. The *XREAL* description on the other hand possesses the advantage that a recalculation only needs to be triggered if the signal shape changes. This special form of describing signals gives a performance advantage when simulating (quasi-) linear systems since filter outputs are calculated analytically without involving timestep integration. This spares many additional events and calculation steps which speeds-up simulation. The disadvantage of the *XREAL* signal description is its limitations on (quasi-) linear

systems and due to the *XREAL* composition out of a set of analytical basis functions they are not suitable for systems in which arbitrary waveforms can occur.

Closely related to the signal representation is the selection of the domain in which a system is modelled. Since some circuits like FDs or clock buffers are strongly nonlinear in the voltage or current regime but nearly linear in the phase or frequency domain it is beneficial to describe them in the this domain. Because in this regime the systems are linear, the *XREAL* description is used to get the maximum speed performance. At the domain borders, translator elements are needed which transforms the signal from one representation into the other. The most common transformations are discussed in sec. 3.6 and sec. 4.7.

In addition to the signal description and choice of a proper domain, modelling techniques and the process for creating event-driven models with focus on mixed-signal RF ICs is discussed in this work. The way to describe frequency-dependent LTI and nonlinear systems is outlined. Special attention is paid to the modelling of filters in combination with spectral as well as *XREAL* signals. Furthermore, a technique to describe supply coupling effects on analog circuits is discussed. An extra chapter was spent on the detailed explanation on parameter extraction step in which the models are filled with the correct values.

In the course of many verifications of integrated circuits it has been shown that the model development takes too much time, if always both the SV and the C++ code have to be written. Thus, an approach which uses a library of generic modelling cells which describes fundamental analog behaviour was developed. The modularization of the models has three advantages. First, the model development is much faster because only SV code must be written when creating a new model. With the use of a model generator, this process can be speed-up even more. Second, since the standard cells are well tested, the probability of making a mistake in the modelling step decreases. Last, the usage of a cells from a standard library helps if the functionality of the models is extended as for example it is done for the system-level small signal analysis described in chap. 6 because only the standard cells need to be touched and not every model ever created.

For linearised systems, an analysis in the frequency domain instead of a time domain simulation is beneficial since the complete system behaviour can be captured in one simulation with an appropriate number of frequency points. Therefore, the system-level small-signal analysis for noise simulations was developed in course of this work. The analysis makes use of the fact that electrical noise is usually so small that it does not influence the operation point of the circuit. Based upon the already existing RF simulations using spectral signals, small-signal transfer characteristics of the models can be calculated from the large-scale signal operation point. The developed algorithm collects recursively the different noise contributions from the system's

building blocks for given frequency points while considering the derived small-signal transfer functions. In a post-processing step, also the interaction between digital and analog part of the system is considered, which allows a noise analysis for a complete mixed-signal system.

In chap. 7 the proposed RF modelling and simulation techniques are verified using examples of mixed-signal circuits. It can be shown that the correlation between the event-driven simulation using macro models of the analog building blocks and a conventional simulation using Spectre is very high while the simulation speed is increased up to a factor of 66,000. Moreover, it could be shown using the example of a PLL that the usage of *XREAL* signals in combination with a sophisticated partitioning of the modelling domains accelerates the simulation by the factor of 5 in comparison to a conventional real number modelled PLL. Due to the enormous increase in speed, the simulation of complex test scenarios became possible like a calibration of the PLL, a BER simulation and the verification of the transmitter output signals. It could be shown that design errors like Clock Domain Crossing (CDC) which can be hard to detect simulating only the digital circuitry can be found using the here presented mixed-signal simulation approach. Moreover, block specifications can be validated and revised using a mixed-signal simulation which was shown by the OFDM transmission example. A noise analysis helps in finding the source of the highest noise contributor which can give a clue which circuit might be revised or better shielded.

Simulation is not always the silver bullet for the analysis of a system. When looking on complex wire harnesses, an important requirement is the absence of sneak circuits. Since each switch position needs to be considered, the number of simulations would increase exponential with the number of circuits. Thus, a SCA was developed in scope of an industry project which is based on path search in graphs. This analysis method can be counted to the formal verification techniques and circumvents the problem of calculating a high number of different scenarios.

Chapter 9 finally gives an example on how the automated simulation of electrical systems in combination with a GA can be used for the automated optimization of a given wire harness topology. This procedure saves valuable manpower which can be used for other tasks and furthermore avoids introduction of manual errors.

10.2 Outlook

In this thesis, several modelling and analysis techniques for different applications have been introduced. However, in the course of implementing and using these

techniques, some suggestions for improvement or for extending the developed tools and methods came up.

S_M-type systems are one way to describe nonlinear, frequency-dependent behaviour of electronic circuits but for some configurations the error between model and circuit simulation is rather large. It would be interesting to examine whether this error is introduced through the parameter extraction step or whether S_M-type systems are not suitable for describing all kind of circuits.

The idea of the system-level small signal analysis is for the moment only applied for a noise simulation, but the concept can be extended to a small-signal AC simulation or to a sensitivity analysis. Both could be useful to further analyse the analog circuits in context of a mixed-signal system. Calibration algorithm or mixed-signal feedback loops could be further examined with regards to small-signal behaviour and PVT related component variations.

Both the SCA and the WHO tool could for reasons of time only be tested on a few small to medium scale examples. It would be interesting to study their performance when applied to large wire harnesses. Furthermore, the idea of the WHO tool could be transferred to the field of integrated circuits and be used to optimize existing circuit for a specific used case. Moreover, it would be interesting to see whether the GA is the best choice as optimizer or if other algorithms perform better for the harness optimization problem.

BIBLIOGRAPHY

[1] "Three-Electrode Circuit Element Utilizing Semiconductive Materials (U. S. Patent No. 2,524,035)". In: *IEEE Solid-State Circuits Newsletter* 12.2 (2007), pp. 32–33. DOI: 10.1109/n-ssc.2007.4785576.

[2] Robert N. Noyce. "Semiconductor Device-and-Lead Structure, Reprint of U.S. Patent 2,981,877 (Issued April 25, 1961. Filed July 30, 1959)". In: *IEEE Solid-State Circuits Newsletter* 12.2 (2007), pp. 34–40. DOI: 10.1109/n-ssc.2007.4785577.

[3] A.W. Burks. "Electronic Computing Circuits of the ENIAC". In: *Proceedings of the IRE* 35.8 (1947), pp. 756–767. DOI: 10.1109/jrproc.1947.234265.

[4] Gordon E. Moore. "Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff." In: *IEEE Solid-State Circuits Society Newsletter* 11.3 (2006), pp. 33–35. DOI: 10.1109/n-ssc.2006.4785860.

[5] Karl Rupp. *40 Years of Microprocessor Trend Data*. 2020. URL: https://ourworldindata.org/search?q=moore's+law.

[6] Franco Maloberti and Anthony C. Davies, eds. *Short History of Circuits and Systems*. River Publishers, Apr. 2016. 344 pp. URL: https://www.ebook.de/de/product/33655745/short_history_of_circuits_and_systems.html.

[7] Maja Diebig. "Entwicklung einer Methodik zur simulationsbasierten Dimensionierung von Kfz-Bordnetzen". de. In: (2016). DOI: 10.17877/DE290R-17350.

[8] S.W. Rienstra. *Modelling and Perturbation Methods*. Eindhoven University of Technology, 2002.

[9] Michael Stamper. "Reducing Automotive Wire Harness Design Time and Cost Using Simulation". In: *SAE Technical Paper Series*. SAE International, 2016. DOI: 10.4271/2016-01-0106.

[10] D. Price. "Pentium FDIV flaw-lessons learned". In: *IEEE Micro* 15.2 (1995), pp. 86–88. DOI: 10.1109/40.372360.

[11] J. Teich. "Hardware/Software Codesign: The Past, the Present, and Predicting the Future". In: *Proceedings of the IEEE* 100.Special Centennial Issue (2012), pp. 1411–1430. DOI: 10.1109/jproc.2011.2182009.

[12] A. Molina and Oswaldo Cadenas. "Functional verification: Approaches and challenges". In: *Latin American applied research Pesquisa aplicada latino americana* (2007).

[13] Charles E. Stroud, Laung-Terng (L.-T.) Wang, and Yao-Wen Chang. "Introduction". In: *Electronic Design Automation*. Elsevier, 2009, pp. 1–38. DOI: 10.1016/b978-0-12-374364-0.50008-4.

[14] Jesse E. Chen. "A modeling methodology for verifying functionality of a wireless chip". In: *2009 IEEE Behavioral Modeling and Simulation Workshop*. IEEE, 2009. DOI: 10.1109/bmas.2009.5338880.

[15] Maja Diebig and Stephan Frei. "Optimizing Multi-Voltage Automotive Power Supply Systems using Electro Thermal Simulation". In: *EEHE Bamberg, Germany, 2014*. 2014.

[16] Chih-Ming Hung and Khurram Muhammad. "RF/analog and digital faceoff - friends or enemies in an RF SoC". In: *Proceedings of 2010 International Symposium on VLSI Technology, System and Application*. IEEE, 2010. DOI: 10.1109/vtsa.2010.5488968.

[17] A. Mercha, W. Jeamsaksiri, J. Ramos, D. Linten, S. Jenei, P. Wambacq, and S. Decoutere. "Impact of scaling on analog/RF CMOS performance". In: *Proceedings. 7th International Conference on Solid-State and Integrated Circuits Technology, 2004*. IEEE. DOI: 10.1109/icsict.2004.1434974.

[18] S D Amico, M De Matteis, G Cocciolo, V Chironi, P Delizia, and A Baschirotto. *Low Power Analog Design in Scaled Technologies*. en. 2009. DOI: 10.5170/CERN-2009-006.103.

[19] Stefan Heinen and Ralf Wunderlich. "High dynamic range RF frontends from multiband multistandard to Cognitive Radio". In: *2011 Semiconductor Conference Dresden*. IEEE, 2011. DOI: 10.1109/scd.2011.6068724.

[20] R. Bagheri, A. Mirzaei, M.E. Heidari, S. Chehrazi, Minjae Lee, M. Mikhemar, W.K. Tang, and A.A. Abidi. "Software-defined radio receiver: dream to reality". In: *IEEE Communications Magazine* 44.8 (2006), pp. 111–118. DOI: 10.1109/mcom.2006.1678118.

[21] Jaijeet Roychowdhury and Alper Demir. "Estimating noise in RF systems". In: *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design - ICCAD '98*. ACM Press, 1998. DOI: 10.1145/288548.288612.

[22] Raoul Pettai. *Noise in receiving systems*. New York: Wiley, 1984. ISBN: 9780471892359.

[23] K. Bult. "Broadband communication circuits in pure digital deep sub-micron CMOS". In: *1999 IEEE International Solid-State Circuits Conference. Digest of Technical Papers. ISSCC. First Edition (Cat. No.99CH36278)*. IEEE. DOI: 10.1109/isscc.1999.759110.

[24] M. van Heijningen, M. Badaroglu, S. Donnay, G.G.E. Gielen, and H.J. De Man. "Substrate noise generation in complex digital systems: efficient modeling and simulation methodology and experimental verification". In: *IEEE Journal of Solid-State Circuits* 37.8 (2002), pp. 1065–1072. DOI: 10.1109/jssc.2002.800927.

[25] Stefan Joeres. *Systemsimulationen zur funktionalen Verifikation von HF- und Mixed-Signal-Schaltungen*. Saarbruecken: Suedwestdeutscher Verlag fuer Hochschulschriften, 2009. ISBN: 9783838102252.

[26] R.G. Meyer and A.K. Wong. "Blocking and desensitization in RF amplifiers". In: *IEEE Journal of Solid-State Circuits* 30.8 (1995), pp. 944–946. DOI: 10.1109/4.400438.

[27] Gerhard Hofmann and Georg Scharfenberg. "Applicability of the ISO 26262 for functional safety for motorbikes". In: *2014 International Conference on Applied Electronics*. IEEE, 2014. DOI: 10.1109/ae.2014.7011679.

[28] Thomas R. Egel. "Wire Harness Simulation and Analysis Techniques". In: *SAE Technical Paper Series*. SAE International, 2000. DOI: 10.4271/2000-01-1293.

[29] E. Segev, S. Goldshlager, H. Miller, O. Shua, O. Sher, and S. Greenberg. "Evaluating and comparing simulation verification vs. formal verification approach on block level design". In: *Proceedings of the 2004 11th IEEE International Conference on Electronics, Circuits and Systems, 2004. ICECS 2004.* IEEE. DOI: 10.1109/icecs.2004.1399731.

[30] William K. Lam. *Hardware Design Verification: Simulation and Formal Method-Based Approaches*. PRENTICE HALL COMPUTER, Nov. 1, 2008. 585 pp. ISBN: 0137010923. URL: https://www.ebook.de/de/product/8431337/william_k_lam_hardware_design_verification_simulation_and_formal_method_based_approaches.html.

[31] Radek Pelanek. "Fighting State Space Explosion: Review and Evaluation". In: *Formal Methods for Industrial Critical Systems*. Springer Berlin Heidelberg, 2009, pp. 37–52. DOI: 10.1007/978-3-642-03240-0_7.

[32] Yifan Wang. *A hierarchical modeling and virtual prototyping methodology for functional verification of RF mixed-signal SoCs*. Muenchen: Verl. Dr. Hut, 2014. ISBN: 9783843918374.

[33] Sebastian Steinhorst. "Formal verification methodologies for nonlinear analog circuits". PhD thesis. Johann Wolfgang Goethe-Universitaet, 2011.

[34] Sebastian Steinhorst and Lars Hedrich. "Model Checking of Analog Systems using an Analog Specification Language". In: *2008 Design, Automation and Test in Europe*. IEEE, 2008. DOI: 10.1109/date.2008.4484700.

[35] John M. Dunn. "Where Did EM Simulation Tools Go?: A Comparison of How EM Tools Were Used in Circuit Simulators 25 Years Ago to Today". In: *IEEE Microwave Magazine* 15.1 (2014), pp. 65–69. DOI: 10.1109/mmm.2013.2288712.

[36] Janick Bergeron. *Writing testbenches : functional verification of HDL models*. Boston: Kluwer Academic, 2000. ISBN: 0306476878.

[37] Ira Miller Dan Fitzpatrick. *Analog Behavioral Modeling with the Verilog-A Language*. SPRINGER NATURE, Oct. 1, 1997. 211 pp. ISBN: 0792380444. URL: https://www.ebook.de/de/product/3814963/dan_fitzpatrick_ira_miller_analog_behavioral_modeling_with_the_verilog_a_language.html.

[38] C. Warwick. "In a Nutshell: How SPICE works". In: *IEEE EMC Soc. Newslett., no. 22* (2009).

[39] Ken Kundert. *The Designer's Guide to Spice and Spectre*. Kluwer Academic Publishers, 2003. DOI: 10.1007/b101824.

[40] Desmond J. Higham David F. Griffiths. *Numerical Methods for Ordinary Differential Equations*. Springer-Verlag GmbH, Nov. 11, 2010. ISBN: 9780857291486. URL: https://www.ebook.de/de/product/19207453/david_f_griffiths_desmond_j_higham_numerical_methods_for_ordinary_differential_equations.html.

[41] Tjalling J. Ypma. "Historical Development of the Newton–Raphson Method". In: *SIAM Review* 37.4 (1995), pp. 531–551. DOI: 10.1137/1037125.

[42] Nuno Borges Carvalho Jose Carlos Pedro. *Intermodulation Distortion in Microwave and Wireless Circuits*. ARTECH HOUSE INC, Apr. 1, 2003. 450 pp. ISBN: 1580533566. URL: https://www.ebook.de/de/product/3823383/jose_carlos_pedro_nuno_borges_carvalho_intermodulation_distortion_in_microwave_and_wireless_circuits.html.

[43] Kundert. "Simulation methods for RF integrated circuits". In: *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD) ICCAD-97*. IEEE, 1997. DOI: 10.1109/iccad.1997.643622.

[44] Josef Stoer. *Einfuehrung in die Numerische Mathematik I*. Berlin, Heidelberg: Springer Berlin Heidelberg Imprint Springer, 1979. ISBN: 9783662068625.

[45] C.-J. Richard Shi. "Mixed-signal system-on-chip verification using a recursively-verifying-modeling (RVM) methodology". In: *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*. IEEE, 2010. DOI: 10.1109/iscas.2010.5537313.

[46] Piloty. "The Conlan Project: Concepts, Implementations, and Applications". In: *Computer* 18.2 (1985), pp. 81–92. DOI: 10.1109/mc.1985.1662801.

[47] E.L. Acuna, J.P. Dervenis, A.J. Pagones, F.L. Yang, and R.A. Saleh. "Simulation techniques for mixed analog/digital circuits". In: *IEEE Journal of Solid-State Circuits* 25.2 (1990), pp. 353–363. DOI: 10.1109/4.52156.

[48] M. Zwolinski, C. Garagate, Z. Mrcarica, T.J. Kazmierski, and A.D. Brown. "Anatomy of a simulation backplane". In: *IEE Proceedings - Computers and Digital Techniques* 142.6 (1995), p. 377. DOI: 10.1049/ip-cdt:19952128.

[49] P. Frey and D. O Riordan. "Verilog-AMS: Mixed-signal simulation and cross domain connect modules". In: *Proceedings 2000 IEEE/ACM International Workshop on Behavioral Modeling and Simulation*. IEEE Comput. Soc, 2000. DOI: 10.1109/bmas.2000.888372.

[50] Ahmed Hussein Osman Ron Vogelsong and Moustafa Mohamed. "Practical RNM with SystemVerilog". In: *CDNLive 2015*. 2015.

[51] Jonathan B. David. "Radio receiver mixer model for event-driven simulators to support functional verification of RF-SOC wireless links". In: *2010 IEEE International Behavioral Modeling and Simulation Workshop*. IEEE, 2010. DOI: 10.1109/bmas.2010.6156596.

[52] *IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language*. DOI: 10.1109/ieeestd.2013.6469140.

[53] *IEEE Standard VHDL Language Reference Manual*. DOI: 10.1109/ieeestd.2009.4772740.

[54] Roland Priemer. *Introductory Signal Processing*. WORLD SCIENTIFIC, 1990. DOI: 10.1142/0864.

[55] Jaeha Kim, Kevin D. Jones, and Mark A. Horowitz. "Variable domain transformation for linear PAC analysis of mixed-signal systems". In: *2007 IEEE/ACM International Conference on Computer-Aided Design*. IEEE, 2007. DOI: 10.1109/iccad.2007.4397376.

[56] C.E. Shannon. "Communication in the Presence of Noise". In: *Proceedings of the IRE* 37.1 (1949), pp. 10–21. DOI: 10.1109/jrproc.1949.232969.

[57] Anders Jakobsson, Adriana Serban, and Shaofang Gong. "Implementation of Quantized-State System Models for a PLL Loop Filter Using Verilog-AMS". In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 62.3 (2015), pp. 680–688. DOI: 10.1109/tcsi.2014.2377411.

[58] Gustavo Migoni, Ernesto Kofman, and François Cellier. "Quantization-based new integration methods for stiff ordinary differential equations". In: *SIMULATION* 88.4 (2011), pp. 387–407. DOI: 10.1177/0037549711403645.

[59] E. Kofman. "Relative Error Control in Quantization Based Integration". In: *Latin American Applied Research* 39.3 (2009), pp. 231–238.

[60] Sabrina Liao and Mark Horowitz. "A Verilog piecewise-linear analog behavior model for mixed-signal validation". In: *Proceedings of the IEEE 2013 Custom Integrated Circuits Conference*. IEEE, 2013. DOI: 10.1109/cicc.2013.6658461.

[61] Ernesto Kofman. "A third order discrete event method for continuous system simulation". In: *Latin American applied research* 36 (Jan. 2005).

[62] Zhimiao Chen, Zhixing Liu, Lei Liao, Ralf Wunderlich, and Stefan Heinen. "A mixed-domain modeling method for RF systems". In: *2015 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2015. DOI: 10.1109/cicc.2015.7338433.

[63] Fabian Speicher, Jonas Meier, Christoph Beyerstedt, Ralf Wunderlich, and Stefan Heinen. "Advanced Modeling Methodology for Expedient RF SoC Verification and Performance Estimation". In: *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2018. DOI: 10.1109/smacd.2018.8434866.

[64] M. Frigo and S.G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005), pp. 216–231. DOI: 10.1109/jproc.2004.840301.

[65] J. Jang, M. Park, D. Lee, and J. Kim. "True event-driven simulation of analog/mixed-signal behaviors in SystemVerilog: A decision-feedback equalizing (DFE) receiver example". In: *Proceedings of the IEEE 2012 Custom Integrated Circuits Conference*. 2012, pp. 1–4. DOI: 10.1109/CICC.2012.6330558.

[66] *IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language*. DOI: 10.1109/ieeestd.2018.8299595.

[67] Bjarne Stroustrup. "Evolving a language in and for the real world". In: *Proceedings of the third ACM SIGPLAN conference on History of programming languages*. ACM, 2007. DOI: 10.1145/1238844.1238848.

[68] Zhimiao Chen. "SystemC-basierte Verhaltensmodellierung und Virtual Prototypong von RF-SoCs". en. PhD thesis. 2017, p. 2017. DOI: 10.18154/RWTH-2017-09469.

[69] Byong Chan Lim and Mark Horowitz. "An Analog Model Template Library: Simplifying Chip-Level, Mixed-Signal Design Verification". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.1 (2019), pp. 193–204. DOI: 10.1109/tvlsi.2018.2873387.

[70] J. Katzenelson and L. Seitelman. "An Iterative Method for Solution of Networks of Nonlinear Monotone Resistors". In: *IEEE Transactions on Circuit Theory* 13.3 (1966), pp. 317–323. DOI: 10.1109/tct.1966.1082594.

[71] M. Rewienski and J. White. "A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 22.2 (2003), pp. 155–170. DOI: 10.1109/tcad.2002.806601.

[72] J.R. Phillips. "Automated extraction of nonlinear circuit macromodels". In: *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference (Cat. No.00CH37044)*. IEEE. DOI: 10.1109/cicc.2000.852706.

[73] W. Pankiewicz. "Algorithms: Algorithm 337: calculation of a polynomial and its derivative values by Horner scheme". In: *Communications of the ACM* 11.9 (1968), p. 633. DOI: 10.1145/364063.364089.

[74] Rob A. Rutenbar, Georges G. E. Gielen, and Jaijeet Roychowdhury. "Hierarchical Modeling, Optimization, and Synthesis for System-Level Analog and RF Designs". In: *Proceedings of the IEEE* 95.3 (2007), pp. 640–669. DOI: 10.1109/jproc.2006.889371.

[75] Christoph Beyerstedt, Fabian Speicher, Jonas Meier, Ralf Wunderlich, and Stefan Heinen. "Baseband Equivalent Modelling Approach for Analog Linear Transfer Functions in Event-driven Simulations". In: *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2019. DOI: 10.1109/smacd.2019.8795301.

[76] Alain Y. Kibangou and GÉrard Favier. "Identification of Parallel-Cascade Wiener Systems Using Joint Diagonalization of Third-Order Volterra Kernel Slices". In: *IEEE Signal Processing Letters* 16.3 (2009), pp. 188–191. DOI: 10.1109/lsp.2008.2011706.

[77] W. Silva. "Identification of Nonlinear Aeroelastic Systems Based on the Volterra Theory: Progress and Opportunities". In: *Nonlinear Dynamics* 39.1-2 (2005), pp. 25–62. DOI: 10.1007/s11071-005-1907-z.

[78] M. Schoukens, K. Tiels, M. Ishteva, and J. Schoukens. "Identification of parallel Wiener-Hammerstein systems with a decoupled static nonlinearity". In: *IFAC Proceedings Volumes* 47.3 (2014), pp. 505–510. DOI: 10.3182/20140824-6-za-1003.00496.

[79] Michael J. Korenberg. "Parallel cascade identification and kernel estimation for nonlinear systems". In: *Annals of Biomedical Engineering* 19.4 (1991), pp. 429–455. DOI: 10.1007/bf02584319.

[80] Jan Hauth. "Grey-box modelling for nonlinear systems". PhD thesis. Techn. Univ. Kaiserslautern, 2008. URL: https://kluedo.ub.uni-kl.de/frontdoor/deliver/index/docId/2045/file/diss.pdf.

[81] S. Baumgartner and W. Rugh. "Complete identification of a class of nonlinear systems from steady-state frequency response". In: *IEEE Transactions on Circuits and Systems* 22.9 (1975), pp. 753–759. DOI: 10.1109/tcs.1975.1084123.

[82] Patrik Larsson. “di/dt Noise in CMOS Integrated Circuits.” In: *Analog Integrated Circuits and Signal Processing* 14.1/2 (1997), pp. 113–129. DOI: 10.1023/a:1008255029409.

[83] J. Briaire and S. Krisch. “Principles of substrate crosstalk generation in CMOS circuits”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 19.6 (2000), pp. 645–653. DOI: 10.1109/43.848086.

[84] M. Pfost and H.-M. Rein. “Modeling and measurement of substrate coupling in Si-bipolar IC's up to 40 GHz”. In: *IEEE Journal of Solid-State Circuits* 33.4 (1998), pp. 582–591. DOI: 10.1109/4.663563.

[85] B.E. Owens, S. Adluri, P. Birrer, R. Shreeve, S.K. Arunachalam, K. Mayaram, and T.S. Fiez. “Simulation and measurement of supply and substrate noise in mixed-signal ICs”. In: *IEEE Journal of Solid-State Circuits* 40.2 (2005), pp. 382–391. DOI: 10.1109/jssc.2004.841039.

[86] Bumhee Bae, Yujeong Shim, Kyoungchoul Koo, Jonghyun Cho, Jun So Pak, and Joungho Kim. “Modeling and Measurement of Power Supply Noise Effects on an Analog-to-Digital Converter Based on a Chip-PCB Hierarchical Power Distribution Network Analysis”. In: *IEEE Transactions on Electromagnetic Compatibility* 55.6 (2013), pp. 1260–1270. DOI: 10.1109/temc.2013.2250506.

[87] J. Meier, C. Beyerstedt, F. Speicher, F. Menke, R. Wunderlich, and S. Heinen. “Event - Driven Modeling of Cross-Coupling in Phase-Locked Loops Through Supply Paths”. In: *2019 Kleinheubach Conference*. 2019.

[88] Jonas Meier, Fabian Speicher, Christoph Beyerstedt, Tobias Saalfeld, Gregor Boronowsky, Ralf Wunderlich, and Stefan Heinen. “Modeling Power Supply Noise Effects for System-Level Simulation of $\Delta\Sigma$ -ADCs”. In: *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, 2019. DOI: 10.1109/smacd.2019.8795288.

[89] M. Xu, D.K. Su, D.K. Shaeffer, T.H. Lee, and B.A. Wooley. “Measuring and modeling the effects of substrate noise on the LNA for a CMOS GPS receiver”. In: *Proceedings of the IEEE 2000 Custom Integrated Circuits Conference (Cat. No.00CH37044)*. IEEE. DOI: 10.1109/cicc.2000.852683.

[90] Christoph Beyerstedt, Jonas Meier, Fabian Speicher, Markus Scholl, Daniel Blase, Ralf Wunderlich, and Stefan Heinen. “A Fast and Accurate True Event-driven Phase Locked Loop Model”. In: *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2020. DOI: 10.1109/icecs49266.2020.9294942.

[91] B. Gustavsen and A. Semlyen. "Rational approximation of frequency domain responses by vector fitting". In: *IEEE Transactions on Power Delivery* 14.3 (1999), pp. 1052–1061. DOI: 10.1109/61.772353.

[92] Ahmet Arda Ozdemir and Suat Gumussoy. "Transfer Function Estimation in System Identification Toolbox via Vector Fitting". In: *IFAC-PapersOnLine* 50.1 (2017), pp. 6232–6237. DOI: 10.1016/j.ifacol.2017.08.1026.

[93] Kenneth Levenberg. "A method for the solution of certain non-linear problems in least squares". In: *Quarterly of Applied Mathematics* 2.2 (1944), pp. 164–168. DOI: 10.1090/qam/10666.

[94] D. C. Sorensen. "Newton's Method with a Model Trust Region Modification". In: *SIAM Journal on Numerical Analysis* 19.2 (1982), pp. 409–426. DOI: 10.1137/0719026.

[95] E. Wysocki and W. Rugh. "Further results on the identification problem for the class of nonlinear systems SM". In: *IEEE Transactions on Circuits and Systems* 23.11 (1976), pp. 664–670. DOI: 10.1109/tcs.1976.1084151.

[96] Dirk Deschrijver and Tom Dhaene. "A Note on the Multiplicity of Poles in the Vector Fitting Macromodeling Method". In: *IEEE Transactions on Microwave Theory and Techniques* 55.4 (2007), pp. 736–741. DOI: 10.1109/tmtt.2007.893651.

[97] Jaeha Kim, Jihong Ren, and M. A. Horowitz. "Stochastic steady-state and AC analyses of mixed-signal systems". In: *2009 46th ACM/IEEE Design Automation Conference*. 2009, pp. 376–381.

[98] Christoph Beyerstedt, Jonas Meier, Fabian Speicher, Ralf Wunderlich, and Stefan Heinen. "Analysis of the frequency conversion of spurious tones in frequency dividers and development of an event-driven model for system simulations". In: *Advances in Radio Science* 17 (2019), pp. 101–107. DOI: 10.5194/ars-17-101-2019.

[99] Melina Apostolidou, Peter G.M. Baltus, and Cicero S. Vaucher. "Phase noise in frequency divider circuits". In: *2008 IEEE International Symposium on Circuits and Systems*. IEEE, 2008. DOI: 10.1109/iscas.2008.4541973.

[100] J. Roychowdhury, D. Long, and P. Feldmann. "Cyclostationary noise analysis of large RF circuits with multitone excitations". In: *IEEE Journal of Solid-State Circuits* 33.3 (1998), pp. 324–336. DOI: 10.1109/4.661198.

[101] D.N. Held and A.R. Kerr. "Conversion Loss and Noise of Microwave and Millimeterwave Mixers: Part 1–Theory". In: *IEEE Transactions on Microwave Theory and Techniques* 26.2 (1978), pp. 49–55. DOI: 10.1109/tmtt.1978.1129312.

[102] Richard Lyons. *Understanding Digital Signal Processing*. Pearson Education (US), Nov. 1, 2010. 992 pp. ISBN: 0137027419. URL: https://www.ebook.de/de/product/11024296/richard_lyons_understanding_digital_signal_processing.html.

[103] Friedel Gerfers Maurits Ortmanns. *Continuous-Time Sigma-Delta A/D Conversion*. Springer-Verlag GmbH, Dec. 8, 2005. ISBN: 3540284060. URL: https://www.ebook.de/de/product/4007491/maurits_ortmanns_friedel_gerfers_continuous_time_sigma_delta_a_d_conversion.html.

[104] Christoph Beyerstedt, Jonas Meier, Fabian Speicher, Ralf Wunderlich, and Stefan Heinen. "An Event-Driven System-Level Noise Analysis Methodology for RF Systems". In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021. DOI: 10.23919/date51398.2021.9474077.

[105] Amr Lotfy, Syed Feruz Syed Farooq, Qi S Wang, Soner Yaldiz, Praveen Mosalikanti, and Nasser Kurd. "A system-verilog behavioral model for PLLs for pre-silicon validation and top-down design methodology". In: *2015 IEEE Custom Integrated Circuits Conference (CICC)*. IEEE, 2015. DOI: 10.1109/cicc.2015.7338432.

[106] Jack E. Volder. "The CORDIC Trigonometric Computing Technique". In: *IRE Transactions on Electronic Computers* EC-8.3 (1959), pp. 330–334. DOI: 10.1109/tec.1959.5222693.

[107] Jeff Miller. *Sneak Circuit Analysis for the Common Man*. Rome Air Development Center, Air Force Systems Command, 1989. URL: https://books.google.de/books?id=IxaDNwAACAAJ.

[108] E.L. DePalma. "RADC SCAT automated sneak circuit analysis tool". In: *IEEE Conference on Aerospace and Electronics*. IEEE, 1990. DOI: 10.1109/naecon.1990.112910.

[109] C.J. Price and N. Hughes. "Effective automated sneak circuit analysis". In: *Annual Reliability and Maintainability Symposium. 2002 Proceedings (Cat. No.02CH37318)*. IEEE, 2002. DOI: 10.1109/rams.2002.981667.

[110] Mengni Zhu and Zheng Zhou. "System reliability and Sneak Circuit Analysis". In: *2014 10th International Conference on Reliability, Maintainability and Safety (ICRMS)*. IEEE, 2014. DOI: 10.1109/icrms.2014.7107205.

[111] E. W. Dijkstra. "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1.1 (1959), pp. 269–271. DOI: 10.1007/bf01386390.

[112] M. Barbehenn. "A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices". In: *IEEE Transactions on Computers* 47.2 (1998), p. 263. DOI: 10.1109/12.663776.

[113] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. "Exploring Network Structure, Dynamics, and Function using NetworkX". In: *Proceedings of the 7th Python in Science Conference*. Ed. by Gaël Varoquaux, Travis Vaught, and Jarrod Millman. 2008, pp. 11 –15.

[114] Florian Ruf. "Auslegung und Topologieoptimierung von spannungsstabilen Energiebordnetzen". PhD thesis. Technische Universitaet Muenchen, 2015. URL: http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:bvb:91-diss-20151202-1220402-1-6.

[115] Noe Elisa, Longzhi Yang, Fei Chao, and Nitin Naik. "A Comparative Study of Genetic Algorithm and Particle Swarm optimisation for Dendritic Cell Algorithm". In: *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2020. DOI: 10.1109/cec48606.2020.9185497.

[116] Artur Hottmann. "Integration numerischer Optimierungsmethoden in den Entwicklungsprozess von Kfzbordnetzen." MA thesis. 2017.

[117] Savio D Immanuel and Udit Kr. Chakraborty. "Genetic Algorithm: An Approach on Optimization". In: *2019 International Conference on Communication and Electronics Systems (ICCES)*. IEEE, 2019. DOI: 10.1109/icces45898.2019.9002372.

[118] Feng-Cheng Chang. "Recursive formulas for the partial fraction expansion of a rational function with multiple poles". In: *Proceedings of the IEEE* 61.8 (1973), pp. 1139–1140. DOI: 10.1109/proc.1973.9216.

APPENDIX A

APPENDIX

A.1 Proof of Closeness for physically realistic linear operations on *XREAL* signal

To show that linear operations appearing in analog systems will always map signals that can be represented in the *XREAL* form on *XREAL* signals, it will be looked on three different classes of operations:

1. Time shifts
2. Linear scaling
3. Transformations described by linear differential equation systems

The proof for the first two points is straightforward. For a time shift Δt, the starting points $t_{0,i}$ of every parameter set set_i has to be replaced by $t_{0,i} + \Delta t$ (see eq. A.1), whereas for a linear scaling with the factor c, the amplitudes b_i must be weighted with c (see eq. A.1). Since the resulting signals can still be represented as *XREAL* signals, the proof for 1. and 2. is done.

$$\begin{aligned} x(t) &= \sum_{i=0}^{K} \frac{b_i}{(m_i-1)!} \cdot (t-t_{0,i})^{m_i-1} \cdot \exp(a_i(t-t_{0,i})) \cdot u(t-t_{0,i}) \\ \rightarrow y(t) &= \sum_{i=0}^{K} \frac{b_i}{(m_i-1)!} \cdot (t-(t_{0,i}+\Delta t))^{m_i-1} \cdot \exp(a_i(t-(t_{0,i}+\Delta t))) \\ &\quad \cdot u(t-(t_{0,i}+\Delta t)) \end{aligned} \tag{A.1}$$

$$
\begin{aligned}
x(t) &= \sum_{i=0}^{K} \frac{b_i}{(m_i-1)!} \cdot (t-t_{0,i})^{m_i-1} \cdot \exp(a_i(t-t_{0,i})) \cdot u(t-t_{0,i}) \\
\rightarrow y(t) &= \sum_{i=0}^{K} \frac{c \cdot b_i}{(m_i-1)!} \cdot (t-t_{0,i})^{m_i-1} \cdot \exp(a_i(t-t_{0,i})) \cdot u(t-t_{0,i})
\end{aligned}
\tag{A.2}
$$

The proof of 3. is a little bit lengthier. For the sake of simplicity, *XREAL* input signals with only one parameter sets are regarded which is no restriction for generality, since more complex signals can be decomposed into the sum of *XREAL* signals with one parameter set. The operations regarded will have the form of linear transfer function which can be represented in the s-domain (see eq. A.3). In physically realistic systems, the number of poles is higher than the number of zeros, therefore $H(s)$ can be decomposed by partial fractional decomposition [118].

$$
H(s) = \frac{\sum_{l=1}^{L} c_l \cdot s^l}{\sum_{k=1}^{K} d_k \cdot s^k} = \sum_j \frac{g_j}{(s-p_j)_j^n} \tag{A.3}
$$

Again, for linearity reasons it can just be regarded just a single term $\frac{g}{(s-p)^n}$. As eq. 4.14 from chapter 4 shows, the output $Y(s)$ from such a system can be split up to the response to the input $Y_{res}(s)$ and the response to the initial conditions $Y_{init}(s)$ of the system. The response to an input as regarded here is

$$
Y_{res}(s) = \frac{b}{(s-a)^m} \cdot \frac{g}{(s-p)^n}, \tag{A.4}
$$

which can again be decomposed into a sum of partial fractions of the form $\frac{c}{(s-d)^l}$, since the numerator degree (1) is smaller than the denominator degree $(m+n)$. The response to the initial conditions is in the form

$$
Y_{init}(s) = \sum_i \alpha_i \cdot \frac{s^{m_i-\beta_i}}{(s-p)_i^m} \quad \beta_i \in \mathbb{N} \tag{A.5}
$$

Since the numerator degrees $m_i - \beta_i$ are smaller than the denominator degrees m_i, the expressions can be decomposed into expressions of the form $\frac{c}{(s-d)^l}$. Now it was shown that every part of the output $Y(s)$ can be represented with a sum of *XREAL* compatible parameter sets, which means, that also the complete output can be represented like this. So, the proof is provided, that linear operations appearing in analog systems will always map signals that can be represented in the *XREAL* form on *XREAL* signals.

A.2 Example of semi-automatically generated model

```
////module LNA_CG_TO created on 09-Nov-2018 13:49:37 by cbeyerstedt//////

`ifdef SIMULATION
import ams_spectral_dpi_pkg::*;
import ams_spectral_sv_user_pkg::*;
`endif

module LNA_CG_TO (SPC_TRIM,OUTPUTC_TRIM,CSC_BIAS_5USRC,LNA_RFOUT,RF_INX,RF_IN,
                  LNA_BIAS_25USRC,AVSS,AVDD,LNA_RFOUTX);

`ifndef SIMULATION
typedef real signal_type;
typedef real signal_net;
`endif

//parameters
string instance_name = $sformatf("%m");
parameter string digital_domain = "no_domain";
parameter real digital_level = 1.2;
parameter real v_cm_out = 0.0;
parameter string supply_domain_AVDD = "AVDD";
parameter real supply_level_AVDD = 1.2;
parameter string supply_domain_AVSS = "GND";
parameter real supply_level_AVSS = 0.0;
parameter real CSC_BIAS_5USRC_value = 1e-6;
parameter real CSC_BIAS_5USRC_tolerance = 1.0;
parameter string CSC_BIAS_5USRC_type = "current";
parameter real LNA_BIAS_25USRC_value = 1e-6;
parameter real LNA_BIAS_25USRC_tolerance = 1.0;
parameter string LNA_BIAS_25USRC_type = "current";
parameter string noiseFile = ↩
    ↪ "C++/cellviews/parameter_files/LNA_CG_TO/LNA_2G4_NF.csv";

//ports
input signal_net AVDD; //type:supply
input signal_net AVSS; //type:supply
input signal_net CSC_BIAS_5USRC; //type:ibias
input signal_net LNA_BIAS_25USRC; //type:ibias
output signal_net LNA_RFOUT; //type:spectral
output signal_net LNA_RFOUTX; //type:logic
input signal_net OUTPUTC_TRIM[4:0]; //type:spectral
input signal_net RF_IN; //type:logic
input signal_net RF_INX; //type:AVDD
input signal_net SPC_TRIM[3:0]; //type:LNA_RFOUTX

//internal signals
```

```
signal_net net_spectral_out1[1:0];
alias net_spectral_out1[0] = LNA_RFOUT;
alias net_spectral_out1[1] = LNA_RFOUTX;

signal_net net_spectral_in1[1:0];
alias net_spectral_in1[0] = RF_IN;
alias net_spectral_in1[1] = RF_INX;

signal_net net_supply [1:0];
alias net_supply [0] = AVDD;
alias net_supply [1] = AVSS;
signal_net net_logic [8:0];
alias net_logic [0] = OUTPUTC_TRIM[0];
alias net_logic [1] = OUTPUTC_TRIM[1];
alias net_logic [2] = OUTPUTC_TRIM[2];
alias net_logic [3] = OUTPUTC_TRIM[3];
alias net_logic [4] = OUTPUTC_TRIM[4];
alias net_logic [5] = SPC_TRIM[0];
alias net_logic [6] = SPC_TRIM[1];
alias net_logic [7] = SPC_TRIM[2];
alias net_logic [8] = SPC_TRIM[3];

signal_net net_bias [1:0];
alias net_bias [0] = CSC_BIAS_5USRC;
alias net_bias [1] = LNA_BIAS_25USRC;

signal_net net_intern1[0:0];
signal_net net_intern2[0:0];
signal_net net_intern3[0:0];

//dummy signals;
signal_net supply_dummy[-1:0];
signal_net logic_dummy[-1:0];
signal_net bias_dummy[-1:0];
assign supply_dummy = '{0,0};
assign logic_dummy = '{0,0};
assign bias_dummy = '{0,0};

//implementation

AMS_ControlUnit_Spectral #(.control_function("default_on"), ↩
    ↪ .controlType("SE2DIFF"),
.digital_domain_in(digital_domain), .digital_level_in(digital_level),
.num_spec_in(1), .num_spec_out(2), .num_logic(0), .num_bias(0), .num_supply(0),
.supply_domains('{"no_domain"}), .supply_levels('{0.0}), .ref_values('{10e-6}), ↩
    ↪ .ref_tolerances('{1.0}), .ref_types('{"no_type"}), .v_cm(v_cm_out)
) ControlUnit_out1(.in(net_intern3), .out(net_spectral_out1), ↩
    ↪ .supply(supply_dummy), .logic_in(logic_dummy), .bias_in(bias_dummy));

AMS_ControlUnit_Spectral #(.control_function("enable_high"), ↩
    ↪ .controlType("DIFF2SE"),
```

```
.digital_domain_in(digital_domain), .digital_level_in(digital_level),
.num_spec_in(2), .num_spec_out(1), .num_logic(9), .num_bias(2), .num_supply(2),
.supply_domains('{supply_domain_AVDD,supply_domain_AVSS}), .supply_levels('{supply_level_AVDD,supply_level_AVSS}), .ref_values('{CSC_BIAS_5USRC_value,LNA_BIAS_25USRC_value}), .ref_tolerances('{CSC_BIAS_5USRC_tolerance,LNA_BIAS_25USRC_tolerance}), .ref_types('{CSC_BIAS_5USRC_type,LNA_BIAS_25USRC_type}), .v_cm(v_cm_out)
) ControlUnit_in1(.in(net_spectral_in1), .out(net_intern1), .supply(net_supply), .logic_in(net_logic), .bias_in(net_bias));


//////////////////////custom implementation start//////////////////////////////
//filter
AMS_SpectralFilterReference #(.coefficient_file("LNA_CG_TO/LNA_2G4_TF.csv"),
.As('{0.0,1.26e4}), .Bs('{1.58e12,1.26e4,1.0}), .approx_degree(0))
Filter1(.in(net_intern1[0]), .out(net_intern2[0]));
//LNA
LNA_Spectral_SE #(.noiseFile(noiseFile), .nonlin_order(3),
.nonlin_coeffs('{0.0000000000,5.2469717645,0.0,5.6183091123}))
LNA1 (.in(net_intern2[0]), .out(net_intern3[0]));

//////////////////////custom implementation end////////////////////////////////
endmodule
```

Listing A.1: Example of generated LNA model.

CURRICULUM VITAE

Name	Christoph Beyerstedt
Academic Degree	Master of Science (M.Sc.)
Date of Birth	February 19, 1990
Place of Birth	Aachen, Germany
Nationality	German

Professional Experience

since 05/2021	Software Engineer Rohde & Schwarz GmbH & Co. KG Meckenheim, Germany
09/2016-02/2021	Research Assistant and Ph.D. student Chair of Integrated Analog Circuits and RF systems RWTH Aachen University
01/2016-06/2016	Internship Intel Mobile Communications Duisburg, Germany
10/2012-06/2015	Student Assistant Chair of Integrated Analog Circuits and RF systems RWTH Aachen University
05/2010-07/2010	Internship Radiometer Physics GmbH Meckenheim, Germany

Education

04/2014-07/2016	RWTH Aachen University Electrical Engineering, Information Technology and Computer Engineering RWTH Aachen University Master of Science
10/2010-01/2014	RWTH Aachen University Electrical Engineering, Information Technology and Computer Engineering RWTH Aachen University Bachelor of Science
08/2000-07/2009	Städtisches Gymnasium Rheinbach Abitur Rheinbach, Germany

MY PUBLICATIONS

Peer-reviewed Journal Papers

[1] C. Beyerstedt, J. Meier, F. Speicher, R. Wunderlich, and S. Heinen. Analysis of the frequency conversion of spurious tones in frequency dividers and development of an event-driven model for system simulations. *Advances in Radio Science*, 17:101–107, 2019. doi: 10.5194/ars-17-101-2019. URL https://ars.copernicus.org/articles/17/101/2019/.

Peer-reviewed Conference Papers

[2] C. Beyerstedt, J. Meier, F. Speicher, R. Wunderlich, and S. Heinen. An event-driven system-level noise analysis methodology for rf systems. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 380–385, 2021.

[3] Christoph Beyerstedt, Jonas Meier, Fabian Speicher, Markus Scholl, Daniel Blase, Ralf Wunderlich, and Stefan Heinen. A fast and accurate true event-driven phase locked loop model. In *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, nov 2020.

[4] Markus Scholl, Christoph Beyerstedt, Ralf Wunderlich, and Stefan Heinen. Triple band wireless transceiver demonstrating reliable and low latency communication. In *2020 27th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, nov 2020.

[5] C. Beyerstedt, F. Speicher, J. Meier, R. Wunderlich, and S. Heinen. Baseband equivalent modelling approach for analog linear transfer functions in event-driven simulations. In *2019 16th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, pages 281–284, 2019.

[6] J. Meier, F. Speicher, C. Beyerstedt, T. Saalfeld, G. Boronowsky, R. Wunderlich, and S. Heinen. Modeling power supply noise effects for system-level simulation of $\Delta\Sigma$ -adcs. In *2019 16th International Conference on Synthesis, Modeling,*

Analysis and Simulation Methods and Applications to Circuit Design (SMACD), pages 265–268, 2019.

[7] Markus Scholl, Tobias Saalfeld, Christoph Beyerstedt, Fabian Speicher, Jonas Meier, Michael Hanhart, Leo Rolff, Vahid Bonehi, Moritz Schrey, Ralf Wunderlich, and Stefan Heinen. A 32 MHz crystal oscillator with fast start-up using dithered injection and negative resistance boost. In *ESSCIRC 2019 - IEEE 45th European Solid State Circuits Conference (ESSCIRC)*. IEEE, sep 2019.

[8] J. Meier, C. Beyerstedt, F. Speicher, F. Menke, R. Wunderlich, and S. Heinen. Event - driven modeling of cross-coupling in phase-locked loops through supply paths. In *2019 Kleinheubach Conference*, pages 1–4, 2019.

[9] Tobias Saalfeld, Markus Scholl, Christoph Beyerstedt, Ralf Wunderlich, and Stefan Heinen. A tracking quantizer for continuous time quadrature bandpass sigma-delta modulators. In *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, dec 2018.

[10] Christoph Beyerstedt, Vahid Bonehi, Tobias Saalfeld, Markus Scholl, Ralf Wunderlich, and Stefan Heinen. Analysis and Design of a Passive Sliding IF Mixer With a Novel Built-in Gainstep Mechanism for an Integrated 2.4 GHz RF-Receiver. In *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, pages 313–316, dec 2018. doi: 10.1109/ICECS.2018.8618016.

[11] M. Scholl, T. Saalfeld, J. H. Mueller, Y. Zhang, V. Bonehi, C. Beyerstedt, F. Speicher, M. Schrey, R. Wunderlich, and S. Heinen. A multistandard, triple band wireless transceiver in a 130 nm cmos technology with integrated pas for iot applications. In *2018 IEEE Radio and Wireless Symposium (RWS)*, pages 88–90, 2018.

[12] Fabian Speicher, Jonas Meier, Christoph Beyerstedt, Ralf Wunderlich, and Stefan Heinen. Advanced modeling methodology for expedient RF SoC verification and performance estimation. In *2018 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*. IEEE, jul 2018.

[13] F. Speicher, C. Beyerstedt, M. Scholl, T. Saalfeld, V. Bonehi, M. Schrey, R. Wunderlich, and S. Heinen. Methodology for improved event-driven system-level simulation of an rf transceiver subsystem for wireless socs. In *2018 13th International Conference on Design Technology of Integrated Systems In Nanoscale Era (DTIS)*, pages 1–4, 2018.

[14] Moritz Schrey, Markus Scholl, Tobias Saalfeld, Jan Henning Mueller, Vahid Bonehi, Christoph Beyerstedt, Fabian Speicher, and Stefan Heinen. An o-QPSK modem using an FSK RF front end for IEEE 802.15.4 operation providing maximum-likelihood data. In *2018 International Conference on Computing, Networking and Communications (ICNC)*. IEEE, mar 2018.

[15] S.Vahid M. Bonehi, Christoph Beyerstedt, Zhimiao Chen, Lei Liao, Ralf Wunderlich, and Stefan Heinen. Gain and noise optimization of a passive sliding IF architecture. In *2015 IEEE Radio Frequency Integrated Circuits Symposium (RFIC)*. IEEE, may 2015.

Talks (non peer-reviewed)

[16] Christoph Beyerstedt. Analysis of the frequency conversion of spurious tones in frequency dividers and development of an event-driven model for system simulations. *Kleinheubacher Tagung, September 2018.*

Monographs

[17] Christoph Beyerstedt. Modeling and verification of an integrated rf transceiver in systemverilog. Master's thesis, RWTH Aachen University, 2015.

[18] Christoph Beyerstedt. Design and investigation of a passive sliding-if mixer for direct conversion receivers. Bachelor's thesis, RWTH Aachen University, 2014.