# Design Methodologies for RF and Mixed-Signal Systems

by

Iason Vassiliou

Grad. (National Technical University of Athens) 1991
M.S. (University of California, Berkeley) 1995

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy

in

Electrical Engineering and Computer Science

in the

GRADUATE DIVISION
of the
UNIVERSITY of CALIFORNIA at BERKELEY

Committee in charge:

Professor Alberto Sangiovanni-Vincentelli, Chair
Professor Paul R. Gray,
Professor Shmuel Oren

Spring 1999

The dissertation of Iason Vassiliou is approved:

Chair Date

Date

Date

University of California at Berkeley

Spring 1999

**Design Methodologies for RF and Mixed-Signal Systems**



by

Iason Vassiliou

## Abstract


Design Methodologies for RF and Mixed-Signal Systems

by

Iason Vassiliou
Doctor of Philosophy in Electrical Engineering and Computer Science

University of California at Berkeley

Professor Alberto Sangiovanni-Vincentelli, Chair

As the number of transistors that can be packed on a silicon die increases rapidly following Moore's Law, it is the design and verification procedure that poses the real limit on the complexity of modern electronic systems. Faster and more complex CAD tools are not always enough; it is often a shift in the overall design methodology that can help exploit the capabilities of modern IC processes. In the area of mixed-signal and RF systems, little has been done compared to digital. The continuing growth of the wireless telecommunication market that includes wireless embedded systems, poses an imminent need for the development of analog-RF and mixed-signal design methodologies. So far analog and mixed-signal CAD tools and methodologies are mainly addressing the design of specific modules, such as operational amplifiers or analog-to-digital converters.

The methodology presented here, is based on the top-down, constraint-driven design paradigm. Key idea to that, is the *hierarchical propagation* of constraints based on *behavioral modeling* and *optimization.* At each level of the design hierarchy, performance constraints are mapped onto constraints on the parameters characterizing the blocks of the subsequent level of the hierarchy. At the highest level, behavioral simulation and optimization can be used to evaluate different architectures. Once an architecture has been chosen, the process is repeated until the layout is generated or a module meeting the constraints is found in the library.

Based on this paradigm, to fully develop a methodology for complex RF and mixed-signal systems, it is essential to demonstrate the design flow by examples, using the appropriate CAD tools. Furthermore, to address the specific needs of different design

problems, it is necessary to develop tools and algorithms that fit into the framework of the methodology.

To demonstrate the methodology for complex mixed-signal systems, we have designed, fabricated and tested a video driver system. New behavioral modeling, optimization and layout techniques have been developed or extended from existing ones, in order to provide a full set of tools supporting the design of a class of similar mixed-signal systems. The experimental results from working fabricated parts from the first run, validate our approach. To further expand the methodology to RF systems, a new, system-level simulator has been developed based on Volterra series. This simulation technique addresses the need for a generic, "black-box" characterization of RF modules, independent of implementation details. Detailed examples are presented to validate the approach. Finally, as a vehicle to link the methodology with the high-level design of complex communication systems, including protocol design, the initial system-level design of a voice-mail pager is presented.

Professor Alberto Sangiovanni-Vincentelli
Dissertation Committee Chair

to my parents

# Contents

# List of Figures

# List of Tables

## Acknowledgements

Several people deserve special thanks for their help in making this work possible. First of all, I would like to thank my research advisor, Professor Alberto Sangiovanni-Vincentelli, for his support and guidance. His pioneering ideas on re-defining the design process for systems provided the foundation for this work. His amazingly energetic personality was always a source of motivation. As a teacher he was inspirational and attending his classes was one of the most fun experiences in Berkeley.

During my studies, I have been lucky enough to work closely with Dean Paul Gray, my research co-advisor, who gave me valuable guidance in circuit and system design issues. He is the example of a perfect gentleman and a great teacher and researcher.

A special thanks goes to Professor Yiannis Tsividis, now at Columbia University, who motivated me to apply to Berkeley in the first place, while he was a professor at the National Technical University of Athens, Greece. His great teaching was a primary motivation for me to work in the area of circuits and systems.

I would also like to thank Professor Bob Brodersen and Professor Shmuel Oren for serving on my qualifying exam and dissertation committees.

There are many colleagues from the CAD group that I would like to thank for their valuable cooperation and contributions to this work. I am grateful to Henry Chang for the project idea and work on the Video Driver example and also for giving me the opportunity to work at Cadence twice. Edoardo Charbon patiently spent so much time helping in nearly everything. Alper Demir was a great collaborator and gave me many ideas and lots of help. Special thanks also goes to Ed Liu, Eric Felt, Robert Neff and Amit Mehrotra. I would also like to thanks all other members of the CAD group for creating a great working environment all these years.

I am also grateful to the students of the analog IC group for their help, especially Chris Rudell, Manolis Terrovitis, Todd Weingandt, Keith Onodera and Thomas Cho.

The people at Cadence where I worked for two summers and one semester deserve a special acknowledgment. A special thanks goes to Ken Kundert and Mike Meyer.

A big thanks goes to Maria Cristobalina Moreno for all her love and support. She was a source of fun and a great companion.

The Greek gang of Berkeley was a source of fun and relaxation with long coffee breaks at Nefeli and animated discussions: George Pappas, John Lygeros, Regina Soufli, Kostas Adam, Dimitris Psilos, Stelios Perissakis, Nikos Biziouras, Manolis Terrovitis, Yannis Roussos, Eria Rassiou, Pavlos Tzermias, Christoforos Kozyrakis, Stefanos Loukakos, George Chryssikos, Yiannis Kareliotis and many others.

My friends in Greece who did not forget me and shared with me great vacation time back home, also deserve a big thanks: Panagiotis Spanos, Sotiris and Panagiotis Mavraganis, Yiannis Tzavelakos, Iason Demiros, Lili Skoulariki, Petros Diamantis, Filippos Daragiannis and many others.

A special acknowledgment goes to my Italian friends, Roberto Manduchi, Paolo Pavan, Luca Carloni, Tiziano Villa and Paolo Miliozzi.

I would also like to thank all the members of the administrative staff for their assistance and support and especially Flora Oviedo, Kia Cooper, Tim Castro, Brad Krebs, Ruth Gjerde, Mary Byrnes and Diane Chang.

Finally, I cannot forget my parents, Konstantina and Filippos and my brothers, Michalis and Alexandros, who gave me their love and support all those years.

This research was supported by SRC (DC-324-010) and the California MICRO program. This support is greatly acknowledged.

# Chapter 1

# Introduction

## 1.1 Design of Complex, Mixed-Signal and RF Systems

Driven by Moore's Law, modern integrated circuit processes have provided us with the capability of putting millions of transistors in one silicon die. To exploit this capability for building systems which are faster, smaller in size, cheaper and far more complex in functionality, increasingly sophisticated computer aided design (CAD) tools are being used. Nevertheless, this is not always enough. Often, a shift in the overall design methodology can have a much more significant impact in reducing the design cycle time and increasing the complexity of the systems that can be handled.

A *design methodology* is not just a collection of CAD tools. It is the overall procedure that the designer follows from specifications to a working design. This includes:

- The definition of the problem specifications.
- The decomposition procedure of the main problem to smaller sub-problems that are easier to handle.
- The definition of the CAD tools that are needed; this includes definition of desired input and outputs at the right level of abstraction.
- Tools that support the methodology, such as behavioral simulators, optimizers, circuit simulators and physical assembly tools.
- A well defined and understood flow that includes all the above.

In the digital design domain, this problem has been addressed in a quite successful way, even though deep sub-micron effects present new challenges. Systems are specified in a high-level description language and a hierarchical process is followed until layout is automatically generated. This process is being continuously expanded to include higher levels of abstraction, for example hardware-software co-design.

On the contrary, for wireless and other mixed signal systems, the bottleneck has been the analog/RF portion. Analog design is largely heuristic and intuitive, rather than systematic and structured. Contrary to digital signals, that can be efficiently described by the zero/one abstraction, analog components are mainly characterized by second order effects such as noise, distortion and mismatches. Unlike its digital counterpart, analog circuit design is not supported by fully automatic synthesis tools. This makes the attempt to define a generic methodology for analog circuits and systems a difficult task. Different systems are characterized by different performance metrics and a variety of CAD tools and methodologies is needed to address the specific nature of every design problem. For example, $\Sigma - \Delta$ modulator design requires different techniques and tools than RF design.

To address the need for an efficient design methodology for analog circuits, a "Top-Down, Constraint-Driven Design Methodology" [1] has been proposed. The methodology is a general framework, based on hierarchical propagation of constraints. Its main points are:

- Top-down hierarchical propagation of constraints starting from specifications, using behavioral simulation and optimization.
- Bottom-up hierarchical extraction and verification.
- Maximum support for automatic synthesis tools; complete automation is not possible for every class of analog circuits.
- Consideration of testability at all stages of the design.

Several tools have been developed as a part of an on-going team research in this area: automated layout tools for efficient layout generation of custom analog chips and modules, simulation techniques at all levels of the design hierarchy (e.g. transistor-level simulation and high level behavioral simulation), automatic test pattern generation for analog circuits.

So far, circuits of limited complexity have been addressed, such as D/A converters [2] and A/D converters [3]. However, with the expanding wireless communication market, there is an imminent need to address the increasingly important class of RF and mixed-signal systems with a complete methodology that includes the necessary supporting tools. The design of such systems is often an excessively long procedure. At the system level, heuristics and designers' expertise is used to generate specifications for individual blocks. Often, even though individual components work, the system fails due to inadequate system-level simulation. To compensate, RF and analog blocks are often over-specified, which makes the design task even harder. Due to the lack of efficient behavioral, system-level representations and simulation tools, the exploration of alternative design solutions is often inadequate. System-level simulation tools are often expansions of circuit simulators; the representation of the blocks is implementation-specific that limits the designer's choice and cause the simulation to be prohibitively inefficient. Therefore, a well-defined, more systematic design procedure with supporting tools is needed that will reduce design time and make more efficient use of hierarchy to explore design alternatives.

## 1.2 Goals of the Dissertation

The goal of this dissertation is to develop a rigorous methodology for complex RF and mixed-signal systems. Based on the methodological framework presented in [1] and a variety of layout, behavioral simulation and optimization tools [4] that support the design of circuits such as operational amplifiers, D/A and A/D converters, the first step is to design more complex systems. Thus, the design flow can be demonstrated and the tools and techniques necessary to handle the increasingly complex hierarchy can be identified or developed. In this work, the design and fabrication of a fully integrated video driver chip is undertaken. This design example serves as a means to develop new behavioral and optimization tools and techniques that can assist the hierarchical design of a large class of mixed-signal systems. Since in analog circuits second order effects such as noise and distortion are crucial to the performance of a system, those techniques should allow for the hierarchical enforcement of constraints on such effects, from specifications to layout.

To fully develop a design methodology for RF systems, new behavioral simulation techniques are needed. Those techniques should model all important second order effects and at the same time be implementation independent, so that they do not restrict the exploration of possible implementations. Consequently, the next goal of this dissertation is the development of such behavioral simulation techniques.

Finally, once the methodology is demonstrated for higher complexity systems and the necessary RF behavioral simulation tools are available, it is crucial to demonstrate the design flow for a complex wireless system that includes RF and mixed signal portions. The methodology for wireless systems should also include techniques to link communication systems engineering with IC system design. A complete system-level design methodology, should help in effectively partitioning functionality among software, digital and analog hardware. The system-level design of a voice-mail pager presented in this work, is intended to serve as a vehicle for investigating those issues.

## 1.3 Organization of the Dissertation

In chapter 2, the current state of methodologies and CAD for mixed-signal and RF design is surveyed. Specific emphasis is given to the "Top-Down, Constraint-Driven Design Methodology" described in [1], that provides a foundation for this work. The complete design flow for a video driver system is presented in chapter 3. The fundamental design concepts for phase-locked loops are reviewed and the tools and algorithms used are described. Experimental results from working fabricated parts are also presented. In chapter 4, a new behavioral modeling and simulation tool based on Volterra series is presented, with experimental results to verify the validity of the approach. Chapter 5 presents the high-level design flow for a voice-mail pager system. This design is intended to provide a vehicle to unify the methodology with communications system design and the POLIS hardware-software, co-design methodology. Finally, conclusions are presented in chapter 6.

# Chapter 2

# Background

## 2.1 Mixed-Signal and RF Design

The design of analog/RF and mixed-signal systems is often an intuitive process, sometimes considered to be an “art”. A general design flow can be identified as shown in Figure 2.1. Starting from a set of specifications, which include performance and cost objectives, an architecture is generated. The architecture is selected from a set of topologies or is custom-made to meet the specifications. In general, it consists of a set of schematics containing interconnection and parametric information of circuit elements, e.g. transistors, resistors, capacitors and inductors.

Depending on the complexity of the system, architecture generation may be a hierarchical process. In mixed-signal and RF systems which contain digital subsystems, the generation of the analog and the digital architectures are generally decoupled. The partitioning of the system functionality to digital and analog hardware, is mostly being done heuristically and separate specifications are given to the two subsystems. For the analog/RF subsystem, which is often the most critical part of the design, the architecture generation process has a top-down and a bottom-up phase. Depending on how much emphasis is given to each design phase relative to the other, a design process can be characterized as “bottom-up” or “top-down”.

In the top-down phase, high-level functional simulation or heuristic rules-of-thumb may be used to select an architecture of macro-blocks. Such macro-blocks can be mixers, amplifiers and filters in an RF transceiver. Even though many tools are equipped with behavioral simulation or macromodeling capabilities, the parameters of the macro-blocks

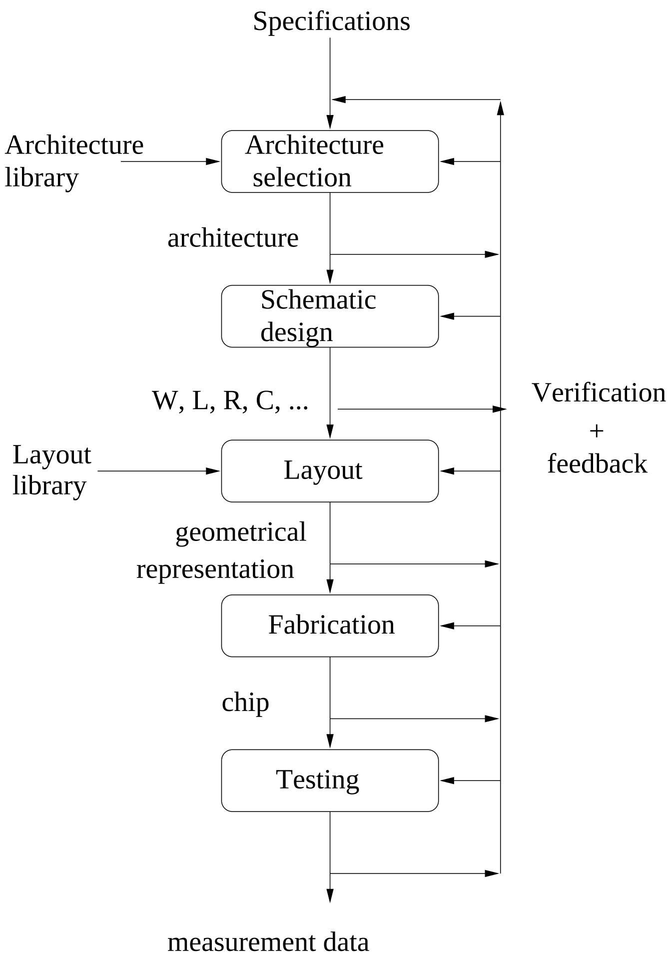



Figure 2.1: Typical IC Design Flow

are often selected by using rough analytical approximations and worst-case operating conditions.

The detailed architecture generation is mostly a “bottom-up” process. A variety of topologies and possibly new heuristics can be explored to identify a detailed sub-block architecture that can meet the high-level specifications. This is typically achieved by trial and error, using a mixture of analytical techniques and circuit simulation. Once a detailed schematic is available, layout is generated using custom or automated techniques. Typically, performance constraints are not strictly enforced during this phase, which causes iterations until the extracted layout meets the performance specifications. Verification of the overall system follows, using information extracted from the final system layout. Ideally, full circuit simulation has to be used. However, this is not always possible due to limitations of circuit simulators in handling very complex systems. Due to the same limitations, for systems containing digital subsystems, full simulation of the combined mixed-signal system is usually impossible. Even when it is possible, the efficiency of the simulation is limited, especially in RF systems, due to inadequate modeling of nonidealities such as distributed RC effects and substrate coupling. At the verification phase, design flaws are often discovered and the designer has to modify the individual blocks or even the overall system architecture. Once the verification is complete, the system is fabricated. Full testing of the prototype fabricated system may be the only way to account for effects such as parasitic coupling, noise, distortion and mismatches. The full functionality of both the analog/RF and digital part of the system is typically verified only at this late stage.

In general, design iterations can occur at any stage of the design flow. Using systematic, hierarchical techniques that take advantage of detailed behavioral modeling and simulation, can minimize costly iterations at the late stages. However, due to the inadequate behavioral modeling and constraint propagation techniques currently available, especially in the RF and mixed-signal design domain, the design is mostly “bottom-up”. As a result, design flaws are discovered late in the design process, which significantly increases the overall design time and time-to-market.

## 2.2 Tools and Methodologies for Mixed Signal and RF Systems

### 2.2.1 Circuit Simulation

Transistor-level simulation using SPICE [5] or any SPICE-based simulator, is the most common CAD technique used in analog circuit design. However, for RF and mixed signal circuits, SPICE-like simulators have significant limitations:

- Typically, in RF and other mixed signal circuits the frequency of different excitations may differ by orders of magnitude, causing the transient simulation to be prohibitively slow.
- The simulation of harmonic intermodulation distortion may also be prohibitively slow and inaccurate due to numerical noise.
- Noise simulation may be inaccurate, if strongly nonlinear elements are present.
- There is no accurate modeling of distributed elements.
- Modeling of coupling through substrate and package parasitics is typically inefficient.
- Accurate modeling of on-chip inductors is needed.

To overcome those shortcomings, various techniques have been used. Harmonic balance [6, 7] is the most common, used for calculating the steady-state response of a nonlinear circuit with harmonic excitations. It is based on substituting a nonlinear system of ordinary differential equations, with an nonlinear algebraic system. The main advantage is that simulation time does not depend on the value of the frequencies present in the system, but rather on the number of frequencies.

In [8, 9, 10], a mixed time-frequency domain method is proposed for systems excited by periodical nonlinearities. The periodical steady-state response is found and linear analysis around the periodical operating point can be used to find frequency translation characteristics of nonlinear modules such as mixers, including noise folding.

Multi-rate, envelope-following simulation techniques [11] use different time-scales to simulate circuits with frequency responses separated by orders of magnitude. Using different time variables, the transient response due to the slow component is computed and, at each time step, steady state analysis is used to find the response due to the fast component.

Since RF systems are extremely sensitive to noise and currently available noise analyses are often inadequate, there is a significant research activity for developing new modeling and simulation techniques. [12], presents a non-Monte Carlo simulation technique for nonlinear dynamic circuits. In [13], new simulation techniques taking into account the effect of cyclostationary noise generation are proposed. In [14], [15] new techniques are proposed for modeling phase noise in oscillators. Finally, in [16] the effects of noisy oscillators driving non-autonomous circuits are examined.

An extensive overview of simulation methods for RF circuits is presented in [17].

### 2.2.2 System-Level Simulation

Full circuit simulation cannot be used in system design, due to prohibitive simulation times. Furthermore, system simulation at the top-down phase, requires hiding implementation details to allow for flexibility in the selection of low-level blocks.

A typical approach for system simulation, mostly at the bottom-up verification phase, is the use of macromodels. Macromodels are extracted from circuit blocks and try to capture as much functionality as possible with less implementation details. Even though they can reduce overall simulation time trading off accuracy, they depend typically on implementation details and they cannot be efficiently used for top-down design. [18] provides an algorithm for macromodel optimization.

Many commercial simulators use analog hardware description languages, such as Verilog-A, to add macromodeling capabilities. They can be efficient in modeling first order functionality but they have significant shortcomings in the simulation of RF systems. Noise modeling and analysis is simplistic, often ignoring frequency translation effects and non-stationarity. Modeling of frequency-dependent nonlinearities may also be inaccurate. Filters are mapped onto pole-zero implementations which may lead to long simulation times for high-order filters. Furthermore, transient, time-domain simulation using widely spaced harmonic excitations, as it is typical in RF systems, may also lead to excessively long simulation times. Harmonic-balance simulation tools equipped with macromodeling capabilities may avoid this problem, but have limitations in simulating strongly nonlinear systems.

Customized system simulators have also been used for specific types of circuits.

SWITCAP [19] uses a discrete-time algorithm for frequency and time domain analysis of ideal switched capacitor circuits. MIDAS [20] is used for discrete-time, functional simulation of mixed-signal sampled systems.

Finally, time-domain macromodeling can be used in communication system simulators, such as PTOLEMY [21] . Usually, first order functional models are used to compute directly information such as bit error rate (BER) in a communication link. Even though they can be useful in co-simulating digital signal processing algorithms with RF circuits, such models do not capture all important second order effects and often depend on implementation details. Those models typically have the same shortcomings with analog hardware description language models used with time-domain transient simulation.

### 2.2.3 Synthesis and Design Methodologies

To accelerate the design of various analog circuit blocks, a significant amount of research has been devoted into developing automated or semi-automated synthesis environments and integrated methodologies with supportive CAD tools.

Automatic synthesis tools and integrated approaches have been mostly limited to generating layout from specifications for specific topologies of analog circuits. Such systems are:

- AIDE2 [22], where a circuit topology described in C-language, is mapped onto a fixed layout based on a library of subcircuits.
- IDAC/ILAC [23, 24] that uses an equation-based architecture selection mechanism and a layout synthesis procedure that allows enforcement of constraints.
- OPASYN [25], that also uses analytical models to select between predefined operational amplifier topologies, but includes more refined second order effects characterization. Optimization based on equations is used for transistor sizing.
- CADICS [26], that automatically generates cyclic A/D converters from specifications. It uses behavioral modeling in the synthesis process and performance-driven layout tools.
- [27] presents a tool for the automatic generation of current interpolative D/A converters. Integer programming is used for optimization in the selection of transistor sizes.

Those tools are optimized for generating specific-type analog circuits of limited complexity. Hybrid systems such as ISAID [28, 29, 30], use a mixed-approach that allows human interaction. Hierarchy is used via macromodeling for architecture selection. Qualitative reasoning is used to improve the initially synthesized circuit.

The ASTRX/OBLX system [31, 32] uses simulated annealing optimization to select circuit parameters. An inner/outer optimization loop technique is used to generate results that are tolerant to low-level parameter variations. Complex design examples such as pipelined A/D converters have been demonstrated.

A hierarchical approach for the design of more complex analog and mixed-signal circuits is the ARIADNE system [33, 34]. It is based on symbolic analysis and optimization. The use of analog high-level description languages is suggested in order to deal with hierarchy.

Most of the aforementioned analog CAD synthesis tools and design environments are limited to the design of small complexity circuit topologies. Hierarchy is not addressed in a robust way and the behavioral simulation used is mostly limited to first order effects. Finally, they are limited to the design of relatively low frequency blocks and the techniques used cannot handle RF frequency effects.

### 2.2.4 RF Hierarchical Design Methodologies

The explosive growth in the area of RF communication systems has also sparked research in methodologies and tools to address the specific needs of those systems. One of the areas where new, robust hierarchical techniques are needed, is linking communications systems engineering with high-level system design of RF circuits.

In [35], an optimization tool is presented for selecting parameters of RF transceivers based on first-order models. In [36], the methodology for the design of a single-chip CMOS transceiver is presented; system-level design is based on communication system simulators such as SPW [37]. A hierarchical technique for linking communication system simulation in SPW with circuit simulation in SPECTRERF, capturing effects such as distortion and I/Q mismatches, is described in [38]. In [39], MATLAB is used as an environment for mixed signal hierarchical modeling of a wireless communication system.

Most of the tools and methodological approaches mentioned, do not provide a robust behavioral modeling capability that accurately captures all non-idealities. A constraint propagation mechanism from specifications to layout, supported by tools at every level of the design hierarchy, is not clearly defined.

## 2.3 Top-Down, Constraint-Driven Design Methodology for Analog Circuits

### 2.3.1 Basic Concepts

To facilitate the design of analog circuits, a "Top-Down, Constraint-Driven Design Methodology" has been proposed in [1]. The methodology is not an automatic CAD tool that generates layout from specifications. It is rather intended to give the general guidelines for a systematic hierarchical process for analog design, supported by CAD tools. It also sets the framework for the development of appropriate CAD tools to support it.

The hierarchical, recursive process is depicted in Figure 2.2. The key idea of the methodology is the *hierarchical propagation* of constraints, based on *behavioral modeling* and *optimization*. At each level of the design hierarchy, performance constraints $P_i < C_i$ are mapped onto constraints on the parameters characterizing the blocks of the subsequent level of the hierarchy. At the highest level, behavioral simulation and optimization can be used to evaluate different architectures. Once an architecture has been chosen, the process is repeated until the layout is generated or a module meeting the constraints is found in the library. Throughout the hierarchy, performance degradation due to parasitics is also taken into account. By setting a budget to the maximum allowable parasitics at every level, the layout can be guaranteed to meet the specifications. At the "leaves" of the hierarchy, circuit simulation and optimization can be used to select transistor sizes and, finally, constrained-driven synthesis techniques are used to generate a layout that uses performance constraints to control parasitics. Automatic test pattern generation is also considered hierarchically.

Behavioral modeling and simulation allow for *early detection* of design faults and *efficient exploration* of the design space. Since models have to be estimated at high levels in the hierarchy, a *bottom-up verification* is also essential to fully characterize components, interconnects and parasitics. The performance of lower level blocks is extracted using circuit simulation and then, recursively, behavioral simulation is used to fully verify the system. Behavioral simulation and optimization are efficient means to perform constraint propagation but any other hierarchical method is acceptable. The overall goal is to guarantee

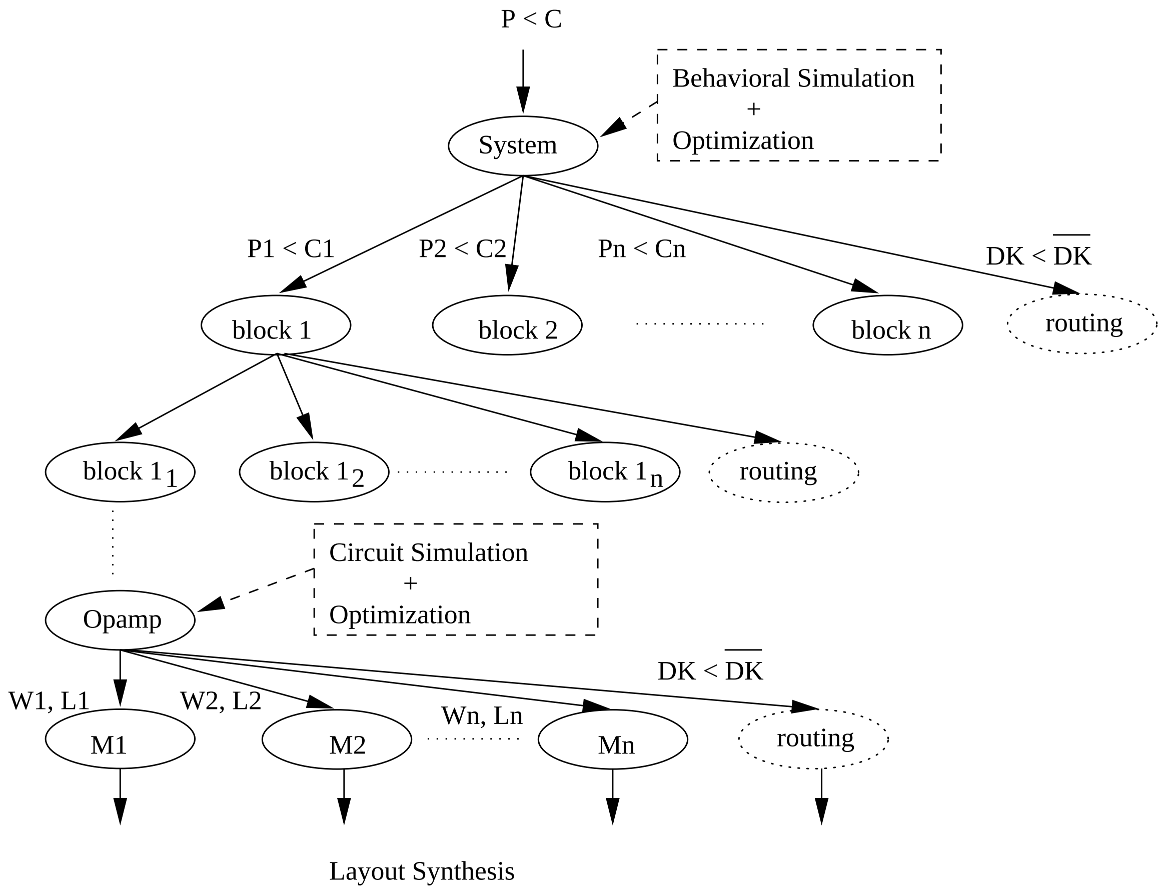



Figure 2.2: Top-Down Design Methodology Flow

that, once the constraints on the blocks of the lower levels of the hierarchy are met, the higher-level specifications are also met.

If optimization is used to perform high-level mapping, a well defined measure of "optimality" is needed. Since blocks are described only with behavioral parameters and no specific implementation is assumed, typical measures such as power or area cannot be used. The goal of the overall design methodology is to accelerate the design cycle and reduce the time-to-market. The concept of a "flexibility function" [1] is the high-level, heuristic measure of optimality that reflects this goal. The *flexibility* of a high-level design parameter is a measure of what degree of difficulty the designer attributes to achieving a certain value for that design parameter. Thus, by selecting to maximize the overall flexibility of the design, given by a weighted sum of the individual flexibilities, the optimization goal becomes the maximization of the "easiness" of the design, leading to a faster time-to-market. Figure 2.3 shows an example of a hyperbolic flexibility function. A very low value of the parameter is very hard to achieve (e.g. noise) so a very low flexibility is attributed while higher values of the parameter are easier to achieve.

Using the notion of flexibility, an optimization problem can be defined to map high-level specifications to lower-level building block constraints:

$$\begin{aligned} max \quad \sum_{i=1}^{n} flex_i(x_i) & \\ s.t. \quad P_1(x_1, x_2, \ldots x_n) &\leq C_1 \\ &\vdots \\ P_k(x_1, x_2, \ldots x_n) &\leq C_k \end{aligned} \tag{2.1}$$

Constraints are typically checked using behavioral or circuit simulation. At the lowest leaves of the hierarchy, when transistor sizes are selected, power or area can be used as optimization goals. A variety of optimization algorithms and behavioral or circuit simulators may be needed to apply the methodology to different classes of analog circuits.

Design examples have shown that using the hierarchical methodology with efficient CAD tools, can have significant advantages:

- acceleration of overall design cycle,
- high probability of working chips from the first fabrication iteration,
- faster exploration of the overall design space.

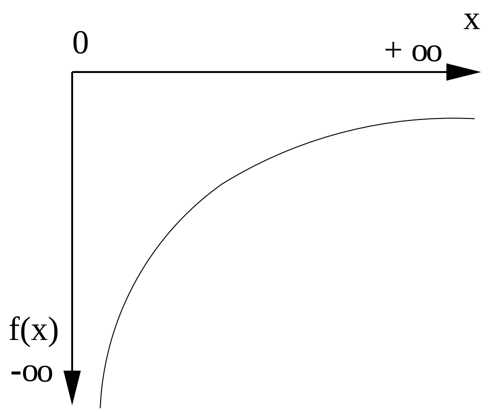



Figure 2.3: Hyperbolic Flexibility Function

### 2.3.2 Tools Supporting the Methodology

To support the application of the methodology to a variety of analog circuits, a number of CAD tools have been developed: constraint-driven analog layout tools, statistical behavioral simulators for data converters, mixed-signal event-driven behavioral simulators, specialized noise simulators and automatic test pattern generators.

#### 2.3.2.1 Constraint-Driven Layout

In [40], the foundation of a constraint-driven layout methodology is presented. Linear sensitivity analysis is used to generate layout parasitic constraints using quadratic programming. Based on this approach, [41], [42] present an extensive suite of tools including stack generation, placement, routing, compaction and extraction.

The layout generation itself is hierarchical. The basic steps are the following:

- A set of high-level specifications, such as transistor sizes and performance constraints, are translated into parasitics' constraints, using a-priori estimated bounds on the parasitics. The constraint generation tool, PARCAR, uses optimization to maximize the *flexibility* of the parasitics [40]. Higher parasitic values are attributed higher flexibilities since they are easier to achieve, thus overall flexibility maximization leads to ease of implementation. Circuit sensitivities are used to evaluate performance degradation due to parasitics.

- Physical size, symmetry and parasitics' constraints are used by the module generation and placement tool (PUPPY-A) to generate placed transistor stacks [43], [44]. Again sensitivities are used to evaluate performance degradations and simulated annealing optimization is used to achieve an optimal placement of stacks.

- Routing is performed by ROAD [45], a maze router based on the A* algorithm that uses sensitivities to generate weights for the cost function driving the area router. Symmetry constraints are also taken into account. A tool for RF and microwave routing that takes into account capacitive and inductive parasitics, interconnect discontinuities and substrate losses, is presented in [46].

- Compaction is performed by the one-dimensional symbolic compactor SPARCS-A, described in [47]. The tool enforces analog-specific constraints combining a Constraint-Graph algorithm and a linear program.

- Finally, extraction techniques are implemented in ESTPAR, described in [42]

A complete layout methodology for mixed-signal circuits including substrate coupling effects is described in [48]. Substrate coupling simulation techniques [49] are used to generate performance sensitivities with respect to a specific placement.

#### 2.3.2.2 Simulation and Testing

Behavioral simulation is essential for the application of the methodology. Unlike digital, analog circuits are characterized by a variety of second order effects, such as noise and mismatches. To capture the effects characterizing different classes of analog circuits, a variety of behavioral modeling and simulation techniques is needed. Behavioral models have to capture all the important effects and, at the same time, be independent of implementation details. In the context of the methodology, work in behavioral simulation has been concentrated into developing techniques for specific sub-classes of analog circuits, such as data converters and phase-locked loops.

In [50], [51], a behavioral representation for Nyquist-rate data converters is presented, that includes nonlinear effects and statistical variations. The representation is based on the calculation of the variance-covariance matrix of the high-level statistical parameters. In [52], a non-Monte Carlo behavioral noise simulation technique for sampled data systems is described.

For the behavioral simulation of phase-locked loops, a mixed-signal, event-driven simulator has been developed [53]. A mixed representation of the PLL building blocks is used, consisting of state equations to characterize continuous variables and state transition tables to characterize digital circuits. An iterative solution algorithm is used to minimize numerical noise. Estimation of timing jitter is done using Monte Carlo simulation.

Behavioral models are used in [54] for the design of optimal tests for D/A converters. In [55], a hierarchical methodology for statistical characterization of mixed-signals systems based on behavioral modeling is presented. The statistical distributions of higher-level parameters are directly calculated from lower-level technology parameters, using behavioral modeling, principal component analysis and response surface methodology.

Efficient transistor-level noise simulation is essential for the correct characterization of analog and RF circuits. In [56, 12], a non-Monte Carlo circuit-level noise simulator is presented. It is based on solving nonlinear, stochastic differential equations in the time domain.

### 2.3.3 Design Examples

Design examples are essential for demonstrating the design flow and illustrating the benefits of any methodology. Thus, great effort has been devoted into designing and fabricating real-world, industrial strength examples. In [2], the design flow for a 10-bit, current-source interpolative D/A converter is presented, supported by experimental results from fabricated chips. The design flow is based on the behavioral simulation techniques discussed in [51] and optimization. The automatic analog layout techniques previously discussed, are also used. One of the great benefits demonstrated, is the fast re-design of a specific type of analog circuit, once the design flow is established. In [2], a great acceleration of the design cycle was achieved for a second design and fabrication iteration of the D/A converter with different specifications. In [3], the design flow of a yet more complicated system, such as a $\Sigma - \Delta$ A/D converter, is presented. Again, behavioral simulation (based on MIDAS [20]) and optimization techniques are used. Experimental results from fabricated chips validate the approach.

## 2.4 Summary

The design of analog RF and mixed-signal systems, presents formidable challenges to the designer. The design is time-consuming and efficient techniques for the acceleration of the overall design cycle are needed. Current CAD tools and methodologies, do not fully address those challenges. The methodology presented in [1], provides a framework that can be expanded with the development of appropriate tools and techniques, to develop an efficient methodology to help accelerate the design cycle of those systems.

# Chapter 3

# Top-Down Design of a Video Driver System

## 3.1 Introduction

The primary goal of this work is to develop a hierarchical design methodology for complex analog/RF, mixed-signal systems, based on the top-down, constraint-driven design paradigm [1], [4]. To achieve this goal, it is crucial to undertake the design of complex, practical commercial systems and prove that increased complexity can be handled with a well-defined, hierarchical design flow supported by the necessary CAD tools. The design of the video driver system presented in this chapter, serves this objective. The system is similar to a Ramdac [57] with an on-chip programmable Phase-Locked Loop (PLL) clock generator and is far more complex than the circuits previously demonstrated using the top-down, constraint-driven design methodology [2], [3].

Unlike digital systems, which can always be described by a zero/one abstraction, the functionality of analog and mixed-signal systems is described by different performance metrics, depending on the system. Thus, each expansion of the methodology to higher complexity systems, requires identifying or developing the appropriate behavioral simulation, optimization, layout and verification tools. To define a design flow for this system, new behavioral modeling, optimization and layout techniques have been developed or extended from existing ones. This set of tools and techniques can support the design of a class of similar mixed-signal systems.

In this chapter, the full hierarchical design flow is presented, from system-level specifications to working fabricated parts. Illustrating the design flow, implies defining a systematic constraint propagation mechanism. At the high-level synthesis phase, emphasis is given at the PLL behavioral models and simulation techniques. The setup of the PLL optimization problem that performs the constraint mapping, together with the optimization algorithm, are described in detail. The novelty of the design approach includes a jitter constraint which is set at the system level and mapped hierarchically onto circuit level constraints. Typically, simulation and usually at the circuit level only, if any, is used to verify the performance at the bottom of the hierarchy, thus causing expensive design iterations. Following the high-level PLL design, the voltage-controlled oscillator (VCO) synthesis phase is depicted, with focus on the optimization approach that takes into account layout parasitics. The layout constraints generated at the circuit level are enforced during the VCO layout synthesis phase. Detailed extraction of the sub-blocks and behavioral system-level simulation is used for the verification of PLL performance. Experimental results from the first fabrication run, show that the chip meets the specifications.

## 3.2 System Description

Video driver systems such as RAMDAC [57] or similar [58, 59, 60, 61] have become indispensable modules in the PC and workstations graphics. The function of a RAMDAC is to convert a set of colors, $2^n$, into another set of colors, $2^m$, using a RAM look-up table and convert the output word into analog signal for the display. Typically $n = 8$, $m = 24$, so 256 out of 16 million possible colors can be viewed at the same time. An external oscillator can be used to provide the clocks required. However, integration of a programmable frequency synthesizer on the same die with the RAMDAC [62], further reduces the cost and gives the flexibility of using the same chip in different systems.

The video driver system we chose to implement includes three basic subsystems: first is the frequency synthesizer PLL, second are three D/A converters and third a digital interface file register for loading the D/A converters and programming the frequency synthesizer. This system is similar to a RAMDAC except that the SRAM lookup table is not implemented. It resembles commercial systems such as CHRONDAC [62] which include a programmable clock generator. A general block diagram of the system is shown in Figure 3.1. In this system, the PLL functions as a programmable clock generator, synthesizing

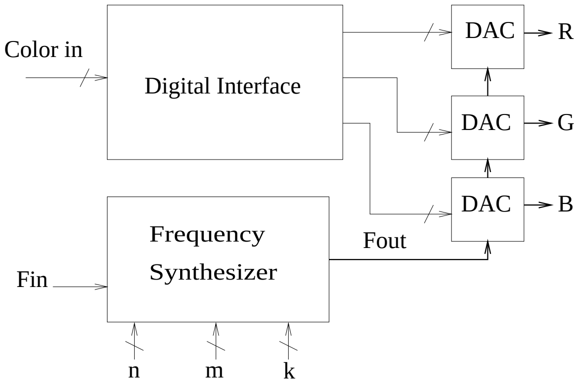



Figure 3.1: Display Driver System Diagram

the various dot clocks required for different screen resolutions. The D/A converters generate the output currents required for the red, green, and blue signals to the monitor.

The system is shown in detail in Figure 3.2. A parallel addressing interface is being used to load the 3 double eight-bit buffers containing the digital words corresponding to the red, green, blue signals for the display. The "blank" pulse forces the three outputs to zero. To test the limits of the performance of the display, each pixel can alternate between two different colors at the highest speed of the clock. The digital words used to control the programmable dividers are also loaded through the same addressing scheme. The clock generator produces the necessary signals for the vertical and horizontal sweep of the display. To complete the system, an external video memory and a "sync" pulse generator may be used.

### 3.2.1 System Specifications

Crucial to the performance of the system are the speed and resolution of the D/A converters as well as the frequency range and timing jitter of the frequency synthesizer. The specifications for the system are summarized in Table 3.1 and are intended to be typical to state-of-the-art for PC and workstation graphics [1]. Typical commercial systems [62] cover low-end PCs to high resolution workstations, giving a frequency range of $25MHz$ to $135MHz$. The timing jitter specification of 1% of the duty cycle is also typical [62, 61]. A number of discrete frequencies needs to be generated that correspond to different display standards. An $1\mu m$ Mosis HP process and a 5 volt supply are set as implementation

[1] for 1993 when the system was conceived

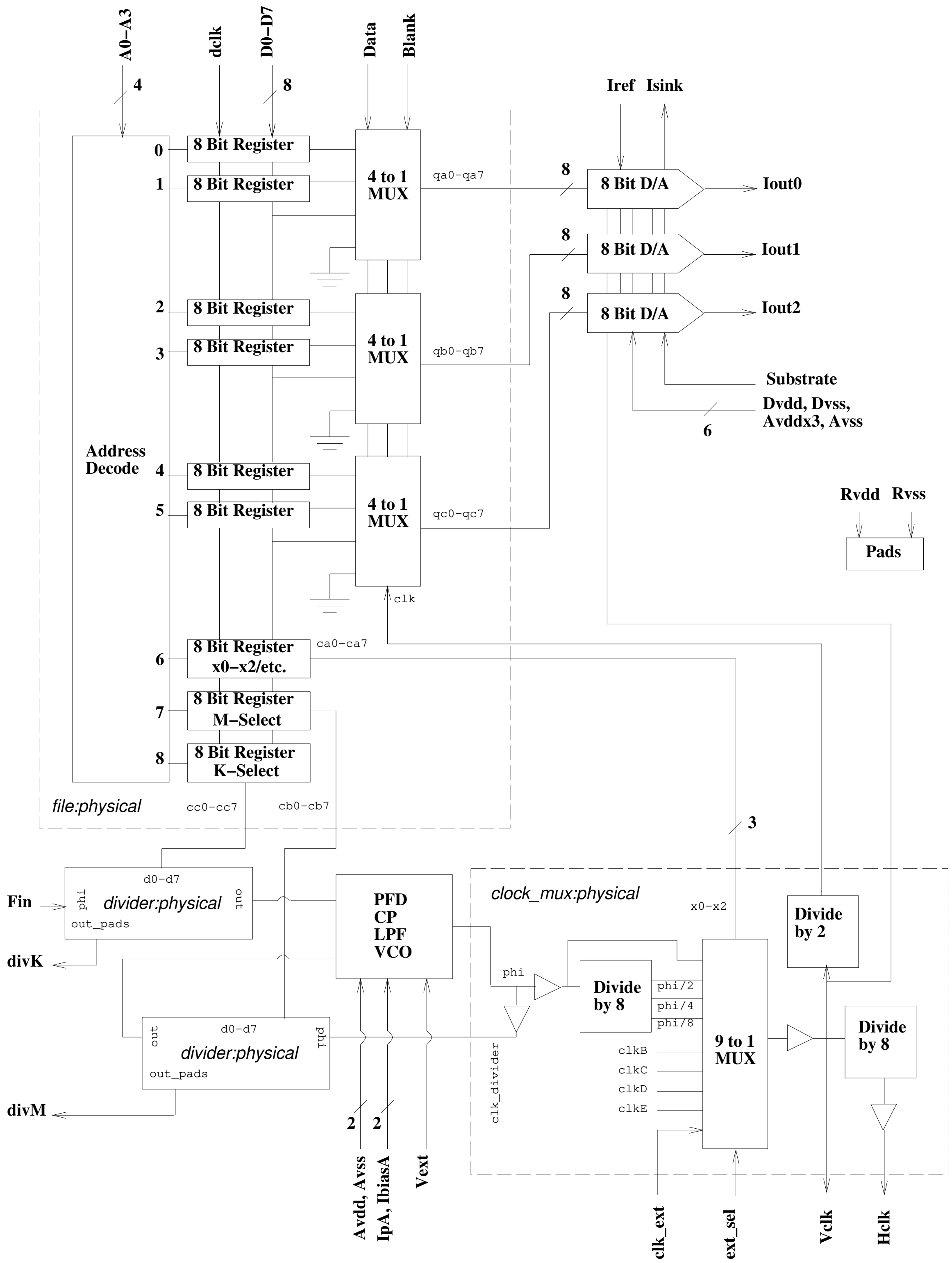



Figure 3.2: Display Driver System Detailed Functional Diagram

| *Type* | *Specification* | *Value* |
|---|---|---|
| Performance | Output Frequencies | 25 to 135 MHz |
| | Timing jitter | $\leq 1\%$ |
| | Video signal INL | $\leq 1$ LSB |
| | Video signal DNL | $\leq 0.5$ LSB |
| | DAC resolution | 8 bits |
| Operation | Supply voltage | 5 V |
| Technology | Spice models | HP CMOS34 |
| | Design rules | SCMOS |

Table 3.1: Video Driver System Specifications

constraints.

## 3.3 Overview of High-Level Synthesis Path

The design hierarchy for the system is highlighted in Figure 3.3. The first level of hierarchy divides the video driver into the following basic system components: three D/A converters, a PLL, and a digital subsystem that contains a data and control path to handle the incoming pixel data, as well as registers to control the frequency of the PLL output. The constraints of Table 3.1 can be trivially decomposed into D/A and frequency synthesizer constraints, since there is no interaction between D/A and PLL performance, except the loading from the parasitic capacitances. The D/A synthesis hierarchy stops after the D/A constraints are given, since an automatic module generator [63] is used. For the digital interface standard cells are used, as well as automatically synthesized cells using SIS [64].

The fully hierarchical method used for the PLL design is described in detail from high-level design to layout, in sections 3.4, 3.5, 3.6. Behavioral modeling and optimization techniques are used, to map high-level performance constraints, onto constraints for PLL building block parameters. At the bottom of the hierarchy, optimization and circuit simulation are used to select parameters for the voltage-controlled oscillator (VCO), which is the most critical PLL building block. At every level of the hierarchical design, constraints are imposed onto routing parasitics, in order to account for the performance degradation after the final layout is done. In section 3.7 the bottom-up hierarchical verification procedure is presented and finally, experimental results are shown in section 3.8.

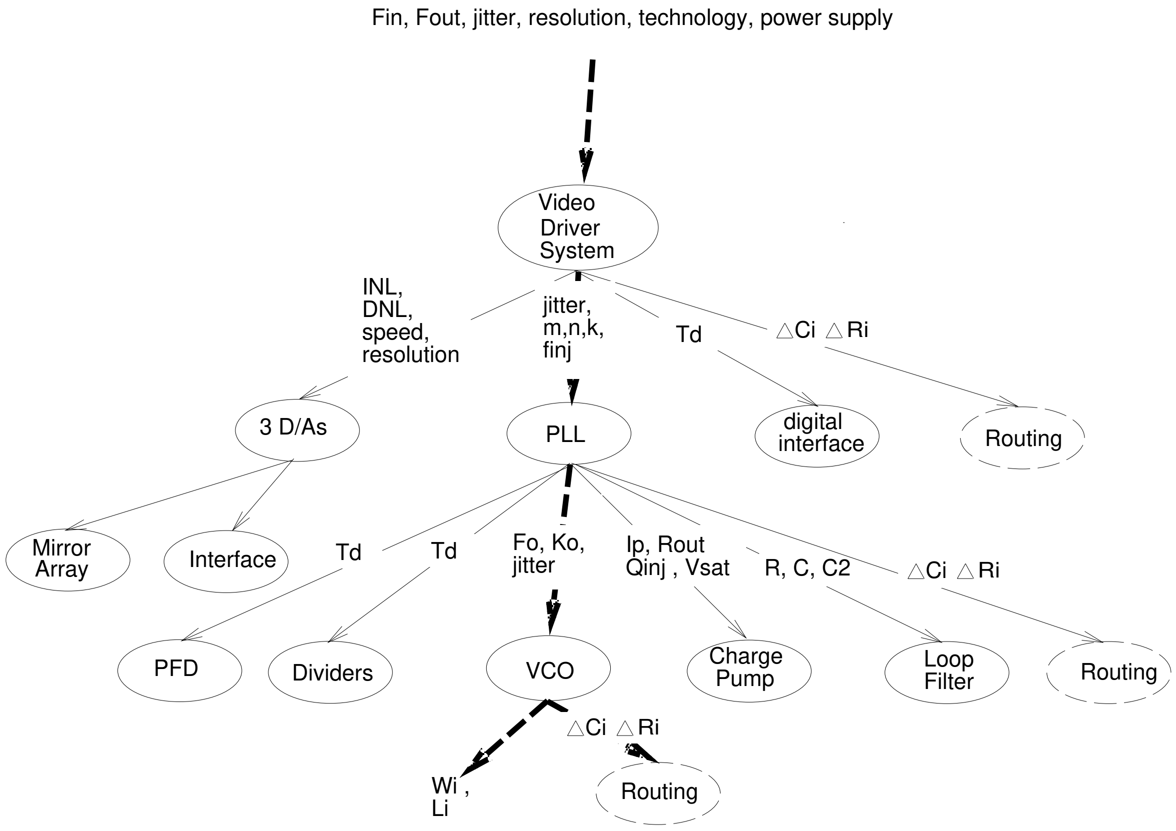



Figure 3.3: Video Driver System Hierarchy

## 3.4 Phase-Locked Loop Hierarchical Design

### 3.4.1 PLL Architecture and Specifications

The architecture selected for the PLL is a charge-pump PLL using a ring oscillator VCO (Figure 3.4). This is one of the most popular architectures for CMOS clock generators and is also used in [61, 65, 62, 66]. The main advantage is that it does not require any external components and can be easily integrated in a cheap CMOS process.

The function of a charge-pump PLL can be briefly described as follows:

1. The phase frequency detector (PFD) compares the input reference frequency $F_{ref}/m$, to the output of the voltage controlled oscillator (VCO) which is divided by $n$.

2. A pulse equal to the duration of the phase difference between the two compared signals is generated to control the charge-pump;

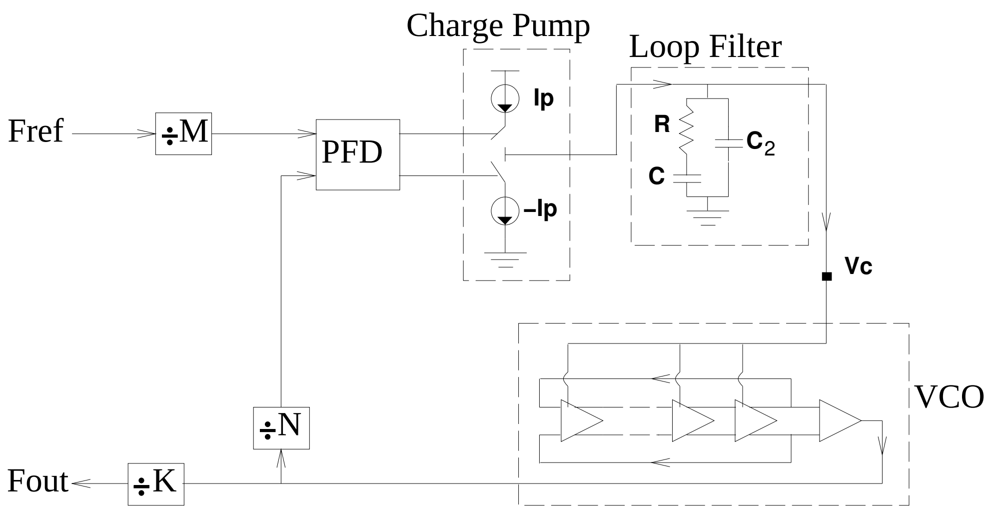



Figure 3.4: PLL Programmable Frequency Synthesizer

3. The current from the charge pump is integrated on the low pass filter ("loop filter") to control the VCO frequency.

4. The loop forces the signal generated by the VCO to have equal phase and frequency with the reference signal. Thus, by changing the divider values, various integer fractions of the input clock can be realized:

$$F_{out} = \frac{N}{M \cdot K} \cdot F_{ref} \tag{3.1}$$

The last divider by $K$ is used so that the PLL operating frequency range required is reduced. As a reference, an external crystal oscillator is used and the divider values needed for the various output frequencies are selected so that the desired operating frequencies are generated precisely. Table 3.2 summarizes the divider values that correspond to the desired output frequencies.

The efficient high-level design of a PLL requires a high degree of analog design expertise and can be an extremely time-consuming task. A brief analysis of PLL design fundamentals presented in the following section illustrates those difficulties and gives the foundation for the behavioral models used in this work.

| $F_{in}(MHz)$ | $F_{out}$ | $N$ | $M$ | $K$ |
|---|---|---|---|---|
| 14.318 | 25.175 | 176 | 25 | 4 |
| | 30.24 | 169 | 20 | 4 |
| | 31.5 | 88 | 10 | 4 |
| | 36 | 101 | 20 | 2 |
| | 40 | 95 | 17 | 2 |
| | 44.9 | 69 | 11 | 2 |
| | 50 | 70 | 10 | 2 |
| | 64 | 152 | 17 | 2 |
| | 65 | 118 | 13 | 2 |
| | 75 | 89 | 17 | 1 |
| | 80 | 95 | 17 | 1 |
| | 86 | 60 | 10 | 1 |
| | 127 | 133 | 15 | 1 |
| | 135 | 132 | 14 | 1 |

Table 3.2: Frequency Synthesizer Divider Values

### 3.4.2 Phase-Locked Loop Design Fundamentals

A simplified PLL block diagram is shown in Figure 3.5. Typically, a PLL consists of a Phase Detector (PD), a Loop Filter and a Voltage Controlled Oscillator (VCO).

The phase of the reference signal $F_{in}$ is compared to the VCO signal $F_{vco}$ by the PD. The output of the PD is typically proportional to the phase difference of the two signals and after being filtered, is applied to the VCO control voltage. The negative feedback loop is adjusting the VCO voltage so that the two compared signals have the same phase.

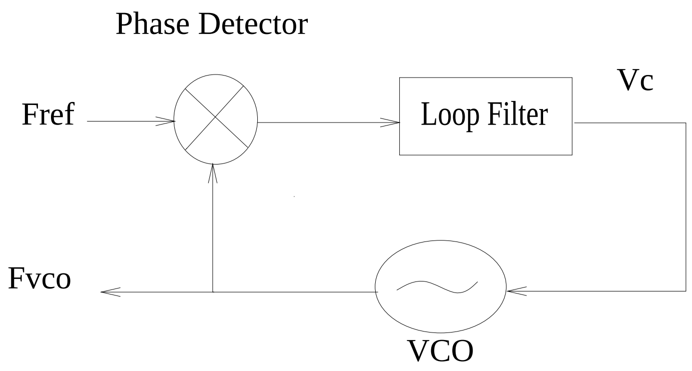



Figure 3.5: PLL Simplified Diagram

#### 3.4.2.1 PLL Linearized Analysis

Even though a PLL is a feedback system that exhibits a strongly non-linear behavior, a linearized analysis can provide significant design intuition [67]. Assuming that:

- the PD output for small phase differences is given by

$$v_d = K_d \cdot (\theta_i - \theta_o) \tag{3.2}$$

  where $\theta_i$ is the input phase, $\theta_o$ is the VCO phase and $K_d$ is a constant,

- the transfer function of the loop filter is $F(s)$,
- and the output frequency of the VCO is given by:

$$\omega_{vco}(s) = \omega_o + K_o \cdot V_c(s) \tag{3.3}$$

it can be easily proved that the PLL transfer function is given by:

$$\frac{\theta_o}{\theta_i} = H(s) = \frac{K_o \cdot K_d \cdot F(s)}{s + K_o \cdot K_d \cdot F(s)} \tag{3.4}$$

The quantity

$$BW = K_o \cdot K_d \tag{3.5}$$

is also referred to as *"PLL bandwidth"*. The precise form of the transfer function depends on the type of the loop filter and phase detector used.

The linearized PLL analysis can give some insight to the operation of the system in steady-state, modeling it as a phase transfer function in the s-domain. Finding the poles and zeros can help analyze the stability of the system in steady-state. However, the behavior of the loop is quite non-linear when the system is not "locked" to the applied input. To compute the "acquisition"time [2] the best method is to resort to simulation that can be extremely CPU intensive, even for simple behavioral models.

#### 3.4.2.2 Analysis of Charge Pump PLLs

The principal advantages of *charge-pump* PLLs over conventional PLLs using a multiplying phase detector are extended frequency range of operation and low cost. The main difference with conventional PLLs is the use of a sequential digital phase/frequency

[2] the time it takes for the PLL to lock after a change of the input frequency

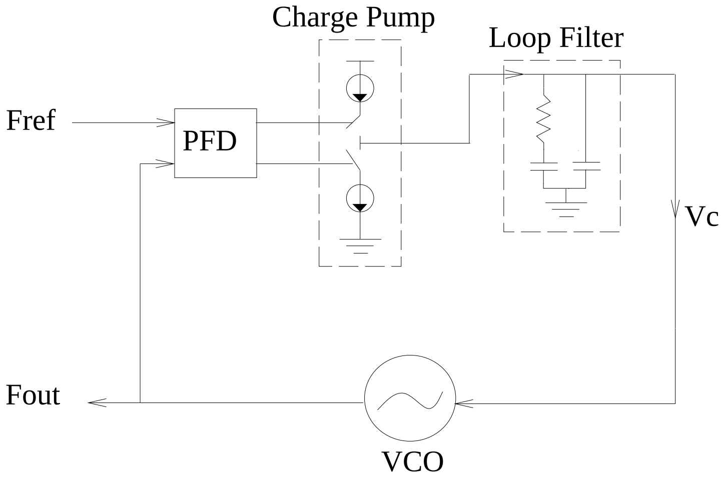



Figure 3.6: Charge Pump PLL

detector (PFD) that converts differences in the phase of the pulses which are being compared into digital waveforms. A charge-pump circuit is used to convert the digital waveform to a control voltage for the VCO. A typical implementation of a charge-pump PLL is shown in Figure 3.6.

A PFD is a sequential circuit that generates a digital pulse with duration equal to the phase difference between the two compared pulses. It has three states of operation:

1. the upper pulse comes earlier and an “up” pulse is generated;
2. the lower pulse comes earlier and a “down” pulse is generated;
3. the edges of the two pulses are perfectly synchronized and no pulse is generated.

The function of a PFD can be illustrated in Figure 3.7.

The pulses generated by the PFD turn on the upper or lower current source of the charge pump, which charges or discharges the loop filter to control the VCO control voltage. When the “up” pulse is on, a current $I_p$ is delivered to the loop filter, while when the “down” pulse is on, the current delivered is $-I_p$. In Figure 3.6 a second order RC filter is used.

Assuming that the phase differences between the pulses are small and the bandwidth of the PLL is much smaller than the input frequency, a continuous time approximation can be used to quantify the behavior of a charge pump PLL [68]. Assuming that:

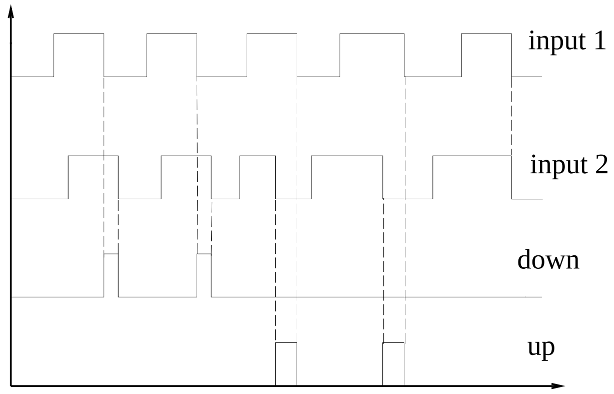



Figure 3.7: PFD I/O Relationship

- the charge pump current is $\pm I_p$,
- the filter impedance is $Z_F(s)$,
- the VCO gain is $K_o$,
- $\theta_i$, $\theta_o$ are the input and output phases respectively,
- $R, C_1, C_2$ are the loop filter components,

the loop transfer function can be expressed as:

$$\frac{\theta_o}{\theta_i} = H(s) = \frac{K_o \cdot I_p \cdot Z_F(s)}{2 \cdot \pi \cdot s + K_o \cdot I_p \cdot Z_F(s)} \tag{3.6}$$

If a second order loop filter is used, as in Figure 3.6, the transfer function becomes:

$$\frac{\theta_o}{\theta_i} = H(s) = \frac{G(s)}{1 + G(s)} \tag{3.7}$$

$$G(s) = \frac{K_o \cdot I_p}{2 \cdot \pi \cdot (C_1 + C_2) \cdot s^2} \cdot \frac{1 + s \cdot R \cdot C_1}{1 + s \cdot R \cdot \frac{C_1 \cdot C_2}{C_1 + C_2}} \tag{3.8}$$

where G(s) is the open loop transfer function. From (3.8) the *phase margin* of the PLL can be calculated, which gives a measure of the stability of the system. A large phase margin is desirable to ensure stability under variations of the parameters of the building blocks. This requires a pole at low frequency, which reduces the speed of the loop in responding to reference input frequency changes.

The time-averaged analysis is a good initial point to consider high-level trade-offs between stability and speed. However, the true nature of the system is sampled, so a more detailed analysis is needed in order to reveal potential instability problems due to the sampling operation. A z-domain analysis was performed in [68]. In the case of a second order loop filter as in Figure 3.6, the criterion for stability is:

$$K \cdot \tau_2 < \frac{4 \cdot (1+a)}{\frac{2\cdot\pi\cdot(b-1)}{b\cdot\omega_i\cdot\tau_2} \cdot \left[\frac{2\cdot\pi\cdot(1+a)}{\omega_i\cdot\tau_2} + \frac{2\cdot(1-a)\cdot(b-1)}{b}\right]} \tag{3.9}$$

where:

- $K = \frac{K_o \cdot I_p \cdot R}{2\cdot\pi}$,
- $\tau_2 = R \cdot C_1$,
- $b = 1 + C_1/C_2$,
- and $a = e^{[-\frac{2\cdot\pi\cdot b}{\omega_i\cdot\tau_2}]}$.

Stability analysis is valid when the PLL is "in lock", so changes in frequency are small and the operation of the system with respect to phase can be considered linear. However, during rapid changes of the input frequency, the PLL operation is quite nonlinear and the nonlinearities of the VCO, charge-pump have to be taken into account.

First, the VCO range of operation should be able to accommodate all the frequencies of operation of the PLL, plus the frequency excursions due to "peaking" during transient. Using an approximate analysis that assumes linear response of the loop, the frequency peaking is calculated in [68]:

$$|\Delta\omega_o| = 2 \cdot \pi \cdot K \cdot \left(\frac{b-1}{b}\right) \cdot \left[\frac{b-1}{b}\left(1 - e^{-\frac{b\|\theta_e\|}{\omega_i\cdot\tau_2}}\right) + \frac{|\theta_e|}{\omega_i \cdot \tau_2}\right] \tag{3.10}$$

Another source of error can be the *dead zone* of the PFD, which is the minimum phase difference to which the circuit can respond "fast enough" by generating a pulse [3]. Charge injection from the switches may also contribute to noise at steady state and cause instability.

To successfully account for all non-idealities, often PLL components are overdesigned. Still, the system-level performance is not guaranteed, since the analytical techniques used are based on approximations. Full verification using detailed simulation is typically done at the transistor level. Due to the nature of the PLL system, where a very high

[3] The difference of the pulses is smaller than the delay of PFD in generating a pulse

frequency signal may be "locking" following a very "slow" loop-filter constant, such simulations can be very lengthy. Even though the use of macro-models may reduce the simulation time, transient simulation techniques are still slow due to the large separation of the time constants typically present in a PLL.

#### 3.4.2.3 Timing Jitter

Very critical to the performance of a PLL used as a clock generator in video drivers is the random fluctuation of the actual phase of the clock, defined as *timing jitter*. Jitter may cause noisy patterns on the screen that can be quite distracting to the human eye. Generally, timing jitter of an oscillator is defined as the standard deviation of the period [4]:

$$\overline{\Delta\tau^2} = E\left[(T - E[T])^2\right] \tag{3.11}$$

where T is the output period, and

$$E[T] = \lim_{N \longrightarrow \infty} \left[\frac{1}{N}\sum_{i=1}^{N} T_i\right] \tag{3.12}$$

is the expected value of $T$.

The *phase jitter* can be accordingly defined as:

$$E\left[{\Theta_{TOT}}^2\right] = \left(\frac{2\pi}{T}\right)^2 \cdot \overline{\Delta\tau^2} \tag{3.13}$$

In the typical case where the reference clock is a crystal oscillator, its contribution to the overall PLL timing jitter can be ignored, since it is several orders of magnitude smaller than the contribution of the VCO. [5]

In a mixed signal environment, VCO timing jitter can be attributed to various sources such as:

- thermal noise,
- coupling of digital noise from the power supply,
- coupling of digital noise from the substrate.

[4] A more accurate discussion on the definition of timing jitter and phase noise which is beyond the scope of this work, can be found in [69].

[5] Typical values can be $\frac{\sqrt{\overline{{\Delta\tau_{VCO}}^2}}}{T} = 10^{-3}$ , $\frac{\sqrt{\overline{{\Delta\tau_{REF}}^2}}}{T} = 10^{-6}$.

Differential circuit techniques and careful layout can be used to eliminate noise from coupling. However, the fundamental limit on jitter performance is set by thermal noise. In section 3.5.1, the thermal noise jitter generation mechanism for a specific type of VCO is described.

A qualitative analysis of the effect of VCO jitter to the system performance is briefly the following: assuming that VCO jitter mainly depends on white thermal noise, timing jitter at the zero-crossing of the clock waveform at any time, is a statistically independent event, uncorrelated to any other jitter event at any other given point in time. Thus, the VCO phase error grows linearly with time, if not corrected by the PLL loop [69]:

$$[\Delta\tau_{VCO}(n \cdot T)]^2 = 2 \cdot n \cdot \Delta\tau^2 \tag{3.14}$$

where $\Delta\tau_{VCO}(n \cdot T)$ is the jitter of the VCO at the n-th period referred to time $t = 0$ and $\Delta\tau$ is the average timing jitter added at each zero-crossing of the oscillator. In a PLL configuration, the VCO phase error is corrected and converges to a finite value, which depends on the speed of the loop. If the PLL bandwidth is large, the error correction mechanism is faster, thus the jitter accumulation is smaller. The effect of the loop bandwidth to jitter from the reference crystal is exactly the opposite; since the incoming signal is filtered by the loop, a smaller loop bandwidth rejects more noise [6]. Typically, for PLLs with noisy integrated VCOs and crystal reference clocks, high bandwidth does not increase the contribution of the reference clock to the overall timing jitter in any significant manner.

In applications such as clock generation, where noise requirements are not as stringent as in oscillators for RF wireless communication systems, VCOs are typically implemented as ring oscillators. This allows easy integration in a cheap CMOS process [61, 65]. Assuming that $T \ll RC_1$ and $C_1 \gg C_2$, an analysis of a charge-pump PLL with a second order passive RC loop filter is done in [70]. Given that each of the $N$ stages of the VCO contributes $\overline{\Delta\tau_N}^2$, an expression is calculated for the PLL phase noise:

$$\sqrt{E\left[\Theta_{TOT}{}^2\right]} = \alpha \frac{2\pi\sqrt{\overline{\Delta\tau_N}^2}}{T} \tag{3.15}$$

where $\alpha$ is the PLL accumulation factor:

$$\alpha = \sqrt{\frac{1}{2I_pK_oRT}} \tag{3.16}$$

[6] see [67] for a more quantitative analysis

The jitter accumulation factor is proportional to the PLL bandwidth, as discussed in the qualitative analysis above: $\omega_L \simeq K_o I_p R$, for $T \ll RC_1$ and $C_1 \gg C_2$.

The analytical techniques for the calculation of timing jitter rely on approximations. Simulation is typically based on standard AC noise analysis that does not take into account the nonlinear change of the operating point. Alternatively, Monte-Carlo simulation can be used which can be extremely lengthy and inaccurate due to numerical noise and poor device modeling . As a result, verification of the overall PLL noise performance is usually done after fabrication. This may cause significant delays in the design cycle if the specifications are not met. The use of accurate, nonlinear device noise simulation techniques coupled with high-level behavioral modeling, may significantly improve the ability to predict overall system noise performance, thus accelerating the design cycle.

### 3.4.3 PLL Behavioral Models and Simulation

From section 3.4.2, it becomes obvious that the best method to accurately predict the behavior of a PLL is resorting to simulation. For the high-level mapping, a behavioral description of the PLL has to be used to map specifications onto constraints for the VCO, loop filter, charge-pump and PFD. It is important however, that the behavioral models are *implementation independent* and capture all the important second order effects determining the performance of analog circuits. Our description is based on the behavioral modeling and simulation techniques presented in [53, 69].

The continuous-time modules of the PLL are described by a set of differential equations:

$$\frac{\partial \theta_i}{\partial t} = 2\pi f_i(t) \tag{3.17}$$

$$\frac{\partial V_c}{\partial t} = \frac{I_{p_{eff}}}{C_2} - \frac{1}{RC_2}V_c + \frac{1}{RC_2}V_x \tag{3.18}$$

$$\frac{\partial V_x}{\partial t} = \frac{1}{RC}V_c - \frac{1}{RC}V_x \tag{3.19}$$

$$\frac{\partial \theta_j}{\partial t} = 2\pi F(V_c(t)) n_d \qquad \forall j, j = 1 \ldots n_d \tag{3.20}$$

where:

$$I_{p_{eff}} = ST \cdot I_p \left(1 + \frac{\Delta I_p}{I_p} \cdot ST\right) - ST \frac{\Delta V}{R_{out}} \tag{3.21}$$

$$F(V_c(t)) = F_0 + K_0 V_c \tag{3.22}$$

The state variables $\theta_i$, $V_c$, $V_x$, $\theta_j$ represent the phase of the input clock, the VCO control voltage, the voltage on capacitor $C$ and the phases of the $n_d$ VCO delay stages respectively. $F(V_c(t))$ is the instantaneous VCO frequency and is assumed to be a linear function when behavioral simulation is performed at the top-down phase. More accurate, higher order models are used at the bottom-up verification phase, once the performance of the VCO is extracted using transistor-level simulation. Finally, $ST$ =0, -1, 1, depending on the state of the PFD. Generally, digital circuits are represented by state transition tables.

The *behavioral representation of timing jitter* is crucial for our design, since it was used as an initial system constraint. The most fundamental timing jitter noise source is the thermal noise of the electronic circuits, which causes uncertainty in the digital waveform transitions. Assuming that the other timing jitter sources have been dealt with, timing jitter of circuit components can be attributed exclusively to thermal noise and be modeled as a white Gaussian random process. In addition to that, we can assume that when the system is in lock, the noise is small enough so that it does not drive the system out of lock. Based on those assumptions, the overall PLL jitter is predicted by adding random noise at the time of each VCO transition. Once a transition is detected by the simulator at time $t$ [7], a white Gaussian random variable $\tau_{jitt}$ is added which corresponds to the VCO timing jitter as defined by (3.11). Thus the exact transition point becomes:

$$t_{tran} = t + \tau_{jitt} \tag{3.23}$$

To find the PLL timing jitter the resulting waveform from the Monte-Carlo simulation is processed [53]. A complete listing of the behavioral parameters used is given in Table 3.3.

Based on those models, a mixed-signal simulator was developed in [53]. The simulation is based on an iterative algorithm to find the exact switching point of the digital waveform. The step can be arbitrarily reduced once a phase transition is computed, in order to achieve the necessary accuracy to simulate timing jitter.

Each one of the PLL performance constraints is generally a function of the variables described above. However, an optimization problem including all the parameters would be unnecessarily complicated, since many have little or no effect on the system performance. Given a particular technology, they could be chosen ad-hoc without affecting the "optimality" of the design. In the case of the VCO, given the CMOS technology used,

[7] This occurs when the phase is a multiple of $\pi$, $\phi = n \cdot \pi$

| *Module* | *Parameter name* | *Description* |
|---|---|---|
| **VCO** | $K_o$ | Voltage to frequency gain |
| | $F_o$ | Center frequency |
| | $\frac{\Delta T}{T}$ | Timing jitter |
| | $V_{min}$ | Lower saturation Voltage |
| | $V_{max}$ | Upper saturation Voltage |
| **Charge-pump** | $I_p$ | nominal charge-pump current |
| | $\frac{\Delta I_p}{I_p}$ | up and down current mismatch |
| | $R_{lin}$ | linear region resistance |
| | $R_{sat}$ | saturation region resistance |
| | $V_{satup}$ | upper source saturation voltage |
| | $V_{satlo}$ | lower source saturation voltage |
| **Loop Filter** | R | Resistor value |
| | $C_1$, $C_2$ | Capacitor values |
| **PFD** | | State transition table |
| | $t_{dead\_zone}$ | Dead zone |
| | $t_d$ | Delay |
| **Divider** | $t_d$ | Delay |

Table 3.3: PLL High Level Parameters

typical values can be easily chosen for $V_{min}$, $V_{max}$. The VCO center frequency $F_o$, can be set at the middle of the desired frequency range. Behavioral parameters of the charge-pump, such as $\frac{\Delta I_p}{I_p}$, $R_{lin}$, $R_{sat}$, may also be chosen ad-hoc, since they are fundamentally limited by the technology and small improvements do not affect the system performance. Delays in the digital dividers are technology-limited and do not have any significant effect on the system performance. Finally, the PFD dead zone problem can be eliminated at the expense of introducing a small steady-state leakage current, as shown in 3.5.3. Typical values of those parameters were used in behavioral simulation. By varying the values, no significant change in the performance was noticed, thus it was decided that they should not be used as optimization variables, in order to reduce the complexity of the problem.

Selecting "manually" the parameters which do not affect optimality does not invalidate the top-down approach. Those values will be constraints for the next levels of hierarchy and once met, the use of accurate behavioral simulation can guarantee the system performance. Table 3.4 summarizes the parameters that were chosen ad-hoc.

| *Module* | *Parameter name* | *Value* |
|---|---|---|
| **VCO** | $F_o$ | 100 MHz |
| | $V_{min}$ | 1.8 V |
| | $V_{max}$ | 5 V |
| **Charge-pump** | $\frac{\Delta I_p}{I_p}$ | 5% |
| | $R_{sat}$ | 2 $M\Omega$ |
| | $V_{satup}$ | 4.8 V |
| | $V_{satlo}$ | 0.2 V |
| **PFD** | $t_{dead\_zone}$ | 0 |
| | $t_d$ | 2 nsec |
| **Divider** | $t_d$ | 2 nsec |

Table 3.4: Constant High Level PLL Parameters

### 3.4.4 High Level Optimization Objective

To determine the remaining parameters which are crucial to the PLL performance, optimization can be used. Since blocks are described by behavioral parameters and no specific implementation is assumed, typical optimality measures such as power or area cannot be used. Power and area constraints at the system-level impose significant limitations to the individual blocks, thus making the design goals harder to achieve. Such constraints can be imposed at the individual blocks at the bottom of the hierarchy. Since the goal of the methodology is to accelerate the design cycle by making the design goal easier to achieve, the concept of the "flexibility function" [1], described also in section 2.3, can be used.

The *flexibility* of a high-level design parameter is a measure of what degree of difficulty the designer attributes to achieving a certain value for that design parameter. Thus, by selecting to maximize the overall flexibility of the design, given by a weighted sum of the individual flexibilities, the optimization goal becomes to maximize the "ease" of the design. Making the low-level parameters "easier" leads to faster design cycles and reduces the time to market, which is the goal of the methodology. Two types of functions were used, one hyperbolic and one parabolic:

$$flex_1(x) = -\alpha x^2 + c \qquad (3.24)$$

$$flex_2(x) = -\frac{\beta}{x} + c \qquad (3.25)$$

The hyperbolic type is used when a higher value of the parameter is harder to meet while the parabolic is used when the opposite is true. In the case of the parabolic function, the

| *flexibility type* | *Parameter name* | *"Easy"* | *"Hard"* |
|---|---|---|---|
| parabolic | $\sqrt{\Delta\tau^2_{VCO}}$ @140 MHz | 7 psec | 1.4 psec |
| hyperbolic | $K_0$ | 50 MHz/V | 80 MHz/V |
| | $I_p$ | 20 $\mu A$ | 200 $\mu A$ |
| | $R, C_1, C_2$ | 0.05 $mm^2$ | 0.1 $mm^2$ |

Table 3.5: Flexibility Coefficients

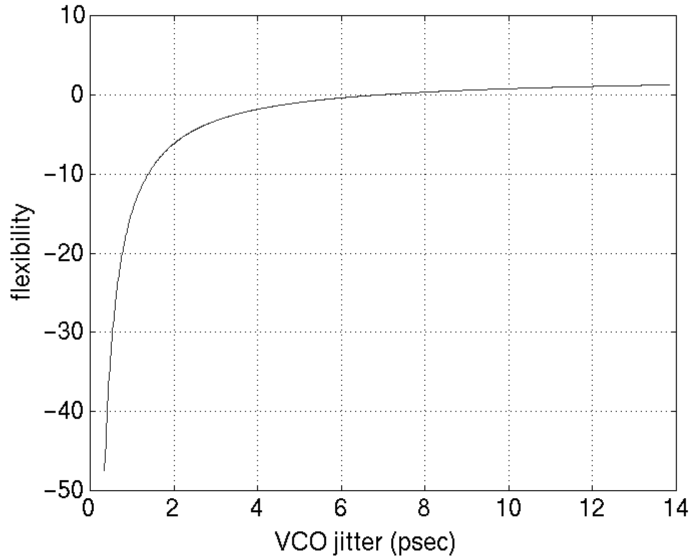



Figure 3.8: $\Delta\tau_{VCO_{rms}}$ Flexibility Function

intuitive criterion is that a zero parameter value is almost impossible to achieve, while for higher parameters it becomes easier and for high enough values the flexibility "increase" is insignificant. The opposite is true for the hyperbolic function.

The criterion used to build the flexibility functions was attributing a value of $flex(x) = -10$ for a parameter value that is "hard" to obtain and $flex(x) = 0$ for a parameter value that is "easy" to obtain. The coefficients are subsequently calculated by solving (3.24), ( 3.25). Weights on the flexibility functions for the individual high-level parameters are determined by trial and error, depending on the optimization results (they are iteratively adjusted so that the optimization result seems "easy" to the designer). Table 3.5 summarizes the values used for the flexibility functions of the high-level PLL parameters. The flexibility functions for parameters $\Delta\tau_{VCO_{rms}}$ and $K_0$ are shown in Figures 3.8, 3.9.

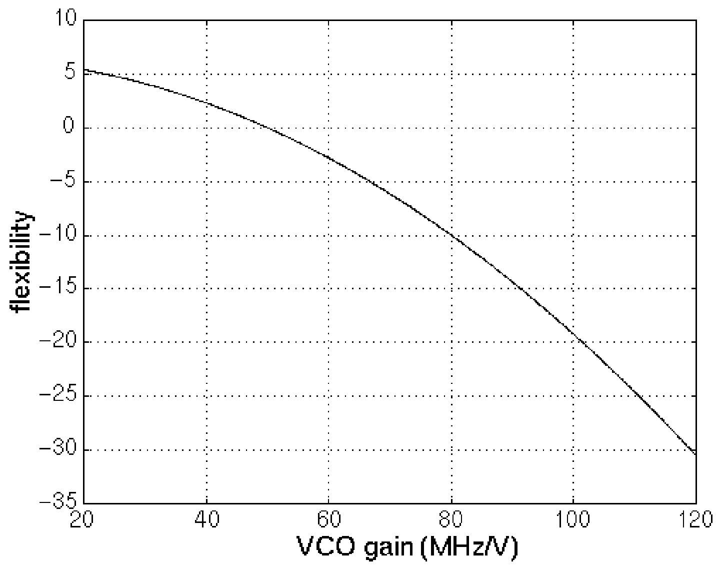



Figure 3.9: $K_o$ Flexibility Function

### 3.4.5 High Level Optimization Problem Formulation

The constraints for the PLL performance can be derived directly from Tables 3.1, 3.2. The first constraint refers to stability at all combinations of divider values. However, if stability has to be checked at each operating frequency, the optimization problem can be unnecessarily complex. From (3.9), stability can be guaranteed at all intermediate operating frequencies, once guaranteed at the lowest frequency [8]. At high frequencies, VCO "overload"[9] can affect the stability of the system. Consequently, after verifying those assumptions using behavioral simulation, only two extreme frequencies for the VCO were used in the optimization process. To ensure stability, a maximum *lock time* constraint $T_{acq}$ is used.

The second critical constraint is timing jitter. Following the analysis presented in section 3.4.2.3, only the worst case for jitter accumulation was considered. This occurs is at the lowest value of the VCO gain (highest divider $n$ value).

Finally, it is very critical for any circuit design optimization problem to yield a result that is *tolerant to parameter variations*. For example, typical R, C variations due

[8]This is verified by extensive behavioral simulation

[9]This refers to the case where the VCO cannot respond to change of the input voltage due to limited operating range

| *Type* | *PLL input Frequency* | *N-divider* | *VCO frequency* | *Value* |
|---|---|---|---|---|
| Stability | 0.56 MHz | 100<br>250 | 56 MHz<br>140 MHz | |
| $\frac{\Delta T}{T}$ | 0.56 MHz | 250 | 140 MHz | $\leq 0.007$ |
| Phase Margin | | | | $\geq 45^o$ |

Table 3.6: PLL Specifications

to the fabrication process can be as high as 30% of the nominal value. In the high-level optimization problem this was taken into account indirectly, by adding a *phase margin constraint*. This can be calculated by considering the linearized open-loop equation for the PLL ( 3.8). A phase margin of $45^o$ was found to be adequate to verify stability under typical parameter variations.

All PLL constraints are summarized in Table 3.6. Constraints have been tightened with respect to the ones of Table 3.1 in order to provide for an extra safety margin.

Given the above constraints, the high level optimization problem can be expressed as:

$$\begin{aligned} max \ \sum_{i=1}^{n} flex_i(x_i) & & \quad (3.26) \\ s.t. \ T_{acq}(x_1, x_2, \ldots x_n, c) &\leq T_{max} \quad (3.27) \\ PM(x_1, x_2, \ldots x_n, c) &\geq 45^o \quad (3.28) \\ \Delta\tau_{PLL140\ MHz}(x_1, x_2, \ldots x_n, c) &\leq 50\ psec \quad (3.29) \end{aligned}$$

where $n$ is the number of parameters used in the optimization: $K_o, \Delta\tau_{VCO}, I_p, R, C_1, C_2$.

A number of other parameters, such as charge-pump output resistances, digital delays and current mismatches, which have a minimal effect to the PLL performance and can be manually selected, are represented by vector $c$.

### 3.4.6 High Level Optimization Algorithm

To solve the above optimization problem, a non-linear optimization method should be used. There are several methods in the literature dealing with nonlinear optimization, typically gradient methods that use first or second derivative information of the constraint and objective functions. Such methods are used in nonlinear optimization software like MINOS [71].

Typical disadvantages of many non-linear optimization algorithms are:

- The values of the first and second order derivatives must be very accurate for the method to converge. When simulators are used to calculate the constraint functions, finite accuracy can cause large errors in the first and second order derivatives.
- Depending on the initial point used, the algorithm may find a local minimum or maximum.
- Often the gradients of the constraint function are not defined outside the feasible region, as in the case of the PLL where we cannot define timing jitter when the system is unstable.

One solution can be the use of simulated annealing, as in [72]. The drawback is that such a solution can be computationally too expensive. Furthermore, it does not use the inherent gradient behavior of circuit performance parameters, thus discarding useful information that can accelerate the optimization process.

A quite efficient method that can be used when the gradient computation is inherently inaccurate, is the supporting hyperplane algorithm, also used in [63], [2]. The algorithm can be used to solve any problem of the form:

$$\begin{aligned} &minimize \;\; \mathbf{c}^T\mathbf{x} \\ &subject\;\; to \;\; \mathbf{g}(\mathbf{x}) \leq \mathbf{0} \end{aligned} \tag{3.30}$$

where:

- $\mathbf{x} \in E^n$,
- $\mathbf{g}(\mathbf{x}) \in E^p$,
- $g_i(\mathbf{x})$ are continuously differentiable functions
- the space defined by the constraints $\mathbf{g}(\mathbf{x})$ is convex.

The algorithm works as follows: After an initial feasible point is given, an unconstrained optimization is performed. Then, the non-linear constraints are checked and if the solution point $\mathbf{P_0}$ is feasible then the algorithm stops and the solution is a global minimum. If a constraint $g_i$ is violated, then the point $\mathbf{u_{k+1}}$ is found on the line joining

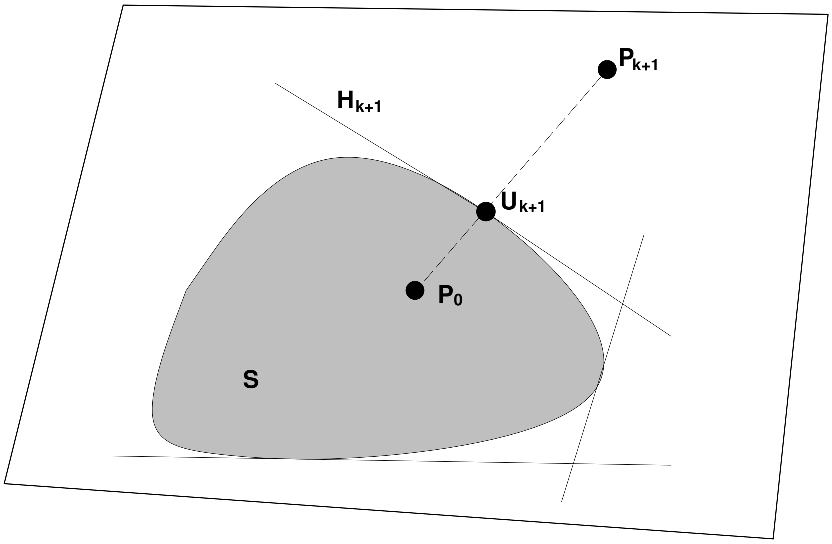



Figure 3.10: Supporting Hyperplane Method

the initial feasible point $\mathbf{P_0}$ and the last solution $\mathbf{P_{k+1}}$, that lies on the boundary of the feasible region $S$. Then a linear constraint is added such as:

$$\nabla g_j(\mathbf{u_{k+1}})(\mathbf{x} - \mathbf{u_{k+1}}) \leq 0 \tag{3.31}$$

Following that, the linearized constrained optimization problem is solved again. This process is repeated until a global minimum is found that meets the non-linear constraints. The algorithm is depicted graphically in Figure 3.10. The key idea is to approximate the feasible space with a polytope, using the tangent hyperplanes at the border points of the feasible space. Then, the linear problem is solved using linear optimization techniques.

Principal advantages of this method are:

- Only first derivative information is needed to compute the equations for the tangent plane. Thus, the problem created by simulator inaccuracies is smaller than in methods using second derivative information.
- Derivatives are only needed inside the feasible space, where the constraint functions are well defined. In the case of the PLL a great problem is eliminated, since the jitter constraint is not defined when the system is unstable.
- The linear subproblem can be solved easily and efficiently by employing a linear optimization tool, in this case MINOS [71].

- Provided that the feasible space is convex, a global optimum is guaranteed.

The most significant disadvantage of the method is that in most circuit design problems, convexity cannot be guaranteed. Given the complexity of circuit design problems, often described by hundreds or thousands of ordinary differential equations, it is impossible to verify convexity in most practical cases. However, the algorithm has been found to work in many complex problems [63], [2]. Typically, when the algorithm fails to converge due to inconvexity, the following strategies may render the solution space convex:

1. reduction of the number of optimization variables,

2. reduction of the range of values for each optimization variable.

This strategy was followed in the PLL high-level optimization and a solution was achieved.

The algorithm may also fail when gradients' computation is not accurate due to the numerical noise of the simulator. This may cause the new added constraint not to exclude the last non-feasible optimal, putting the algorithm in an infinite loop. It can also cause the complete elimination of the feasible region as in the case of non-convex problems.

Great care has been taken to avoid as much as possible the above problems. Since the derivatives are calculated using finite differences, the step for the finite difference is large enough to eliminate the effect of simulator inaccuracy. However, the steps must be taken always towards the feasible direction otherwise the algorithm may wander in the infeasible region where the constraint functions are undefined. The step of the algorithm starts with a high default value and is reduced if the problem is infeasible. If the step is reduced beyond the limit where the difference is higher than the simulator error, a different point is used for drawing the tangent plane which is more "inside" the feasible region. This process may cause loss of optimality but makes the algorithm more robust.

The pseudocode for the optimization algorithm is depicted in Figure 3.11. This algorithm was used in both optimizations done for the PLL design, the high level optimization and the VCO circuit optimization.

### 3.4.7 High Level Optimization Implementation

To solve the optimization problem, a C++ interface with the MINOS [71] optimization package was used. The supporting hyperplane algorithm was implemented separately in C++ using MINOS as a linear solver.

```
set_initial_ffeasible_point;
repeat
    do_linear_optimization;
    foreach non_linear_constraint
        check_constraint;
    if  not_feasible
        find_border_point_binary_search;
        compute_gradients;
        add_new_linear_constraint;
until optimal_is_feasible
```

Figure 3.11: Optimization Algorithm

The phase margin constrained expressed by (3.28), was analytically evaluated using (3.8). Stability and timing jitter constraints were calculated using a modified version of the behavioral simulation tool described in [53] that uses all the models discussed in section 3.4.3. If instability is detected during the timing jitter simulation, the simulated point is in the infeasible region and as a result, no gradients need to be computed. According to (3.15), (3.16), higher PLL bandwidth results to lower jitter. However, from 3.8 it can be seen that high bandwidth can make the system unstable. Since low jitter has a high "flexibility" as described in 3.4.4, the optimization algorithm tends to "push" the solution close to the instability region. To avoid this problem a tight phase margin constraint is added. In order to speed up the optimization, the phase margin constraint is always computed first and only if satisfied, the timing jitter constraint is calculated using behavioral simulation. The parameter values for the initial feasible point needed by the supporting hyperplane algorithm were selected by using behavioral simulation.

The supporting hyperplane algorithm needs a convex feasible space in order to guarantee convergence. In the case of the PLL, using the parameters of Table 3.5 and the constraints of (3.29) results in a non-convex feasible space. To avoid this problem, the value of parameter $K_0$ was held constant and separate optimizations were performed for different values of $K_0$. Since the VCO parameters suffer from large temperature and process

| | | Optimization results | | | |
|---|---|---|---|---|---|
| **Parameters** | $K_0$ *(MHz/V)* | 35 | 40 | 50 | 60 |
| | $\Delta\tau_{VCOrms}$ *(psec)* | 3.1 | 3.33 | 3.46 | 3.33 |
| | $I_p$ $(\mu A)$ | 18.7 | 15.8 | 14.58 | 11.44 |
| | $R$ $(K\Omega)$ | 258.2 | 200.5 | 198.7 | 179.6 |
| | $C_1$ *(pF)* | 40 | 57.8 | 51.95 | 66.9 |
| | $C_2$ *(pF)* 5 | 5 | 5 | 5 | 5 |
| **Constraints** | $\Delta\tau_{PLLrms} \leq 50$ *(psec)* | 48.08 | 50.42 | 51.05 | 50.32 |
| | *Phase margin* $\leq 45(^o)$ | 43.5 | 43.6 | 43.17 | 44.55 |
| **Objective** | *Flexibility* | 4.34 | 2.79 | 1.14 | -3.11 |
| **CPU Time (sec)** | | | 7606.1 | 9198 | 11529 |
| **Iterations** | | 11 | 8 | 8 | 9 |

Table 3.7: Optimization Results

variations which may cause $K_0$ to have variations of up to $\pm 30\%$ of the desired value, this coarse selection of $K_0$ does not create a problem.

### 3.4.8 High Level Optimization Results

Table 3.7 presents the optimization results for different values of $K_0$. The optimizations were performed on a DEC Alpha-Server 2100 5/250 with 256Mb of memory and 4 CPU's. In every case the optimization converges in not more than 11 iterations and within reasonable computation time. Table 3.8 shows the initial points used in each optimization. In every case, the results of Table 3.7 show a great improvement over the initial solution. To help the algorithm converge faster, small tolerances are allowed in the constraints to to detect if a point belongs to the "border" region. The small violation of the phase margin constraint does not have any effect on the system stability as verified by behavioral simulation using high-level parameter tolerances as high as $\pm 30\%$. The timing jitter was intentionally over-constrained to account for other jitter sources such as power supply or substrate coupling.

From the results of Table 3.7 the one that corresponds to $K_0 = 35$ (MHz/V) yields the highest flexibility. However the result for $K_0 = 40$ (MHz/V) is selected, since behavioral simulation shows that it is less affected by parameters variations. Figure 3.12 shows the output of the behavioral transient simulation for the selected parameters. A step in the $n$-divider value is used to verify the locking behavior. In Figure 3.13, the VCO and PLL jitter at every switching event are shown as a function of time. The open loop characteristic

| | | **Initial solutions** | | | |
|---|---|---|---|---|---|
| **Parameters** | $K_0$ *(MHz/V)* | 35 | 40 | 50 | 60 |
| | $\Delta\tau_{VCO_{rms}}$ *(psec)* | 1.03 | 1.03 | 1.03 | 1.03 |
| | $I_p$ $(\mu A)$ | 5 | 5 | 5 | 5 |
| | $R$ $(K\Omega)$ | 220 | 220 | 220 | 220 |
| | $C_1$ *(pF)* | 200 | 200 | 220 | 200 |
| | $C_2$ *(pF)* | 5 | 5 | 5 | 5 |
| **Objective** | *Flexibility* | -38.6 | -39.5 | -48.9 | -44.67 |

Table 3.8: Initial Solutions for Optimization Algorithm

| **Parameter** | **Value** | **Tolerance** |
|---|---|---|
| $K_0$ | 40 MHz/V | ± 10 MHz/V |
| $F_0$ | 100 MHz | ± 30 MHz |
| $I_p$ | 16 $\mu A$ | $\pm 1$ $\mu A$ |
| $R$ | 200 $K\Omega$ | $\pm 60$ $K\Omega$ |
| $C_1$ | 57.8 pF | $\pm 6$ $pF$ |
| $C_2$ | 5 pF | $\pm 1$ $pF$ |

Table 3.9: PLL Parameter Tolerances

for gain and phase is depicted in Figure 3.14, calculated using (3.8).

Since parameters variations have only been taken into account indirectly using the phase margin constraint, it is necessary to verify the performance of the PLL at the worst case variations of the high-level parameters. Those variations are determined by the fabrication process and are typical for phase-locked loops. The optimization solution was found to meet the performance constraints for the parameter variations of Table 3.9.

## 3.5 Low-Level Synthesis

Following our proposed methodology, once the high-level parameters are selected, they are used as constraints for the low-level building block design. Optimization and circuit simulation can be used to map those constraints onto an architecture of transistors, device sizes and layout parasitics. The most important block that affects the PLL performance is the VCO and will be discussed extensively.

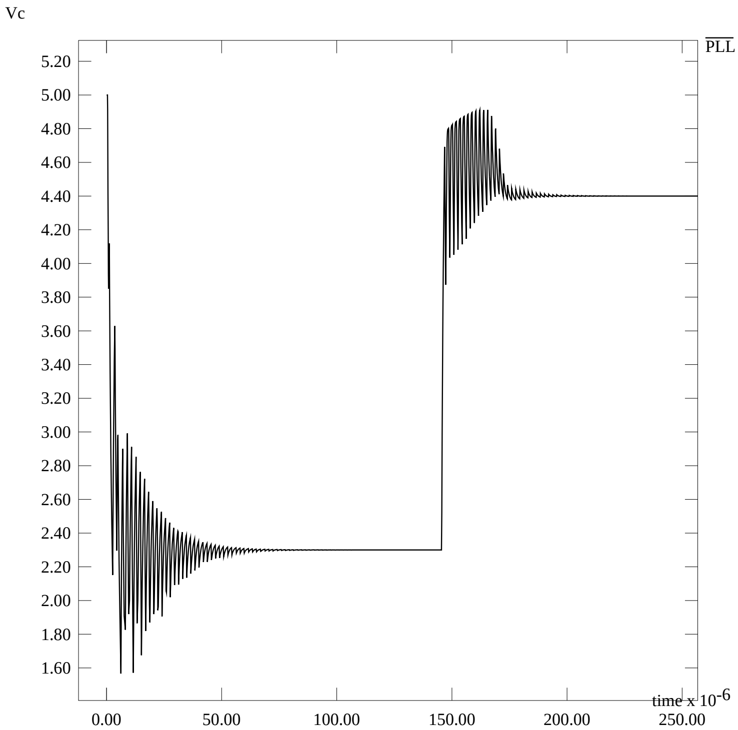



Figure 3.12: PLL Behavioral Simulation

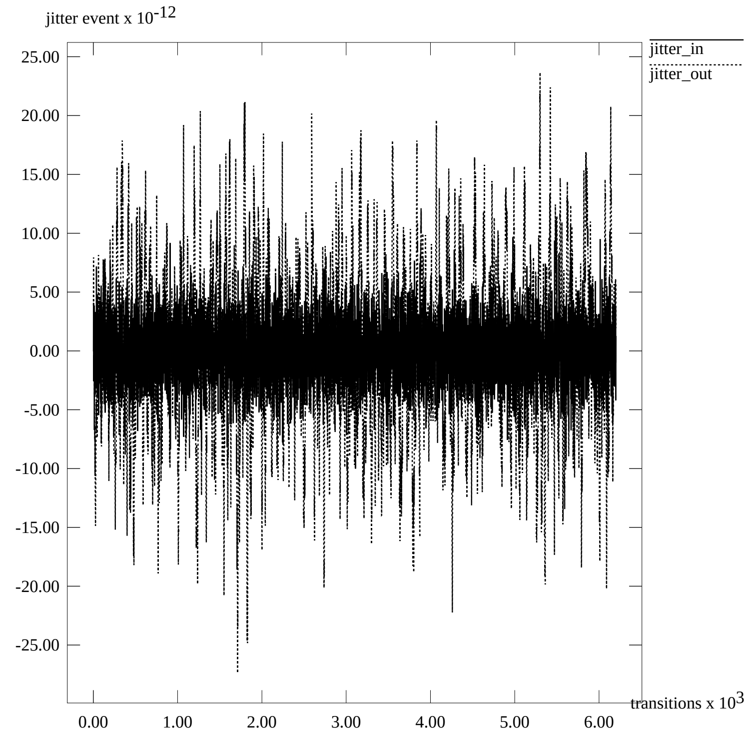



Figure 3.13: PLL Jitter Simulation

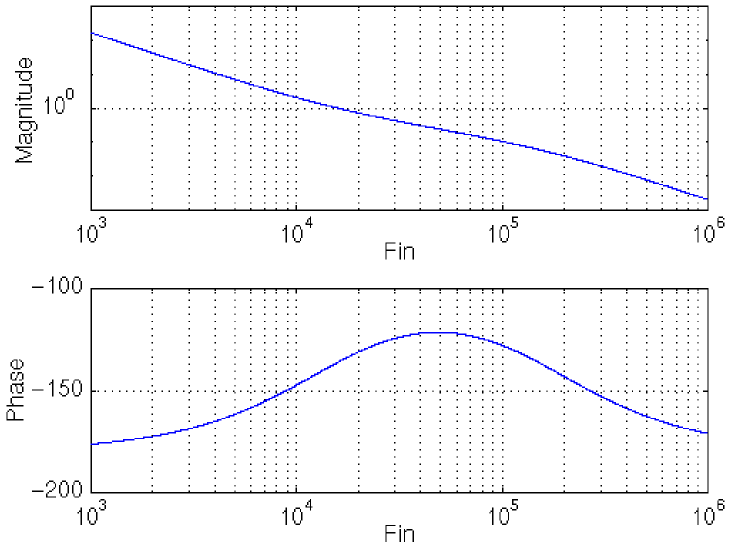



Figure 3.14: PLL Open Loop Characteristic

### 3.5.1 Voltage Controlled Oscillator

#### 3.5.1.1 VCO Design Fundamentals

Various VCO architectures may be used, depending on the requirements of the application:

- Voltage Controlled Crystal Oscillators (VCXO) are typically used when high phase accuracy is desirable, for example in RF local oscillators. However they are expensive and cannot be integrated. Quadrature phase generation is also problematic and the tuning range is small.
- LC-tuned oscillators are also high accuracy alternatives. The main disadvantage is the absence of inductors in most IC technologies. The use of discrete inductors increases the overall cost. As in the case of VCXOs the tuning range is not very large. Recently, developments in IC processes and careful modeling of parasitics, have made the use of integrated inductors possible. However, the quality factor typically achieved is in the order of two to five [73], which limits the possible applications.
- Ring oscillator VCOs are the most suitable for integration in current digital IC technologies. They have a wide tuning range and can generate different phases of the output waveform, which is useful in many applications. The main disadvantage is the poor jitter/phase noise performance.

The simplest type of delay stage that can be used in a VCO is a single-ended inverter. To control the delay, a current-starved inverter may be used, as in Figure 3.15. In this cell, the charging/discharging current is controlled in order to vary the delay. The main disadvantage is poor power supply noise rejection, which makes it unsuitable for a mixed signal environment. Digital switching causes high supply noise that can be coupled to the output of the cell, thus increasing the timing jitter.

To improve the power supply rejection, differential delay stages are often used [61, 65, 74], as in Figure 3.16. Typically, the loads of the differential stage are MOS transistors in the triode region in order to achieve faster switching. If $N$ delay stages are used, the

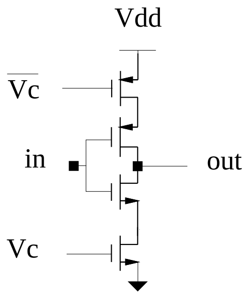



Figure 3.15: Current-Starved Ring Oscillator VCO Cell

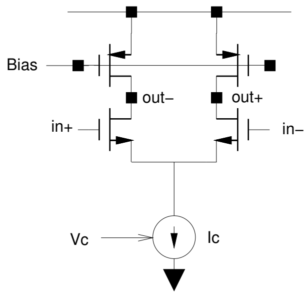



Figure 3.16: Differential Delay Cell

oscillation frequency is given by:

$$F_{out} = \frac{1}{2 \cdot N \cdot T_d} \tag{3.32}$$

where $T_d$ is the delay of one stage.

Differential delay stages do not completely eliminate the problem of power supply rejection, since the transistors of the differential pair are at different operating regions during the transition. However, the fundamental limit on the noise performance of differential stages is set by the device generated noise. For a VCO used in a PLL configuration, we are mainly interested in the thermal noise limit. Flicker noise is mainly at low frequencies and is rejected by the PLL loop which acts as a high pass filter for the VCO generated jitter [70].

In [75] the average jitter in a ring oscillator VCO using differential stages is calculated:

$$\frac{\Delta\tau_{rms}}{t_d} = \sqrt{\frac{kT}{C_L}} \cdot \frac{1}{V_{GS} - V_T} \cdot \xi \tag{3.33}$$

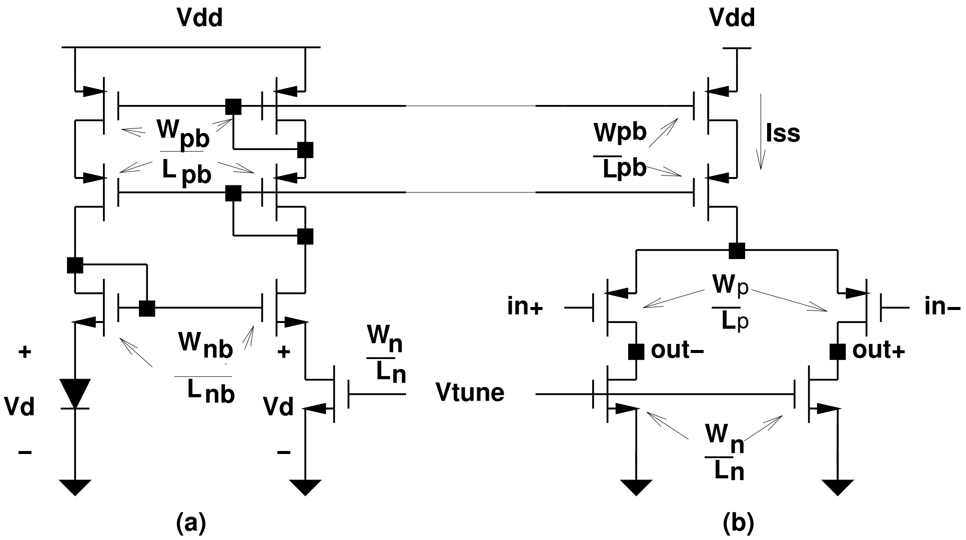



Figure 3.17: VCO Delay Cell and Bias Circuit

where $C_L$ is the capacitive load of each stage of the ring oscillator, $t_d$ is the delay of one stage and $\xi$ is a design parameter given by:

$$\xi = \sqrt{1 + \frac{2}{3}\alpha_v(1 - e^{-t/\tau}) + \frac{2\sqrt{2}}{3}\alpha_v e^{-t/\tau}} \propto \frac{1}{I} \tag{3.34}$$

In (3.34), $\alpha_v$ is the small signal gain of the cell and $\tau \simeq t_d$. From the above equations, it can be inferred that using more power may reduce timing jitter.

#### 3.5.1.2 VCO Architecture and Parameter Selection

As explained in section 3.4.1, a ring oscillator VCO topology was chosen since the noise performance required in not a limiting factor and a wide operating range is desirable. Furthermore, this topology can be integrated in a cheap CMOS process. The delay stage used in this design is a modified version of the topology described in [61] and is shown in Figure 3.17. The bias is a bootstrapped circuit forcing the voltage across the NMOS load transistor operating in the triode region, to be equal to the voltage across the diode. Assuming a simple triode region model and ignoring short channel effects, the bias current is given by:

$$I_{ss} = \mu_n C_{ox}\frac{W_n}{L_n}(V_{tune} - V_{THN} - \frac{V_d}{2})V_d \tag{3.35}$$

The bias current $I_{ss}$ is mirrored through the top current mirror to the delay cell current. The NMOS loads are made equal to the NMOS bias transistor, so when the current fully

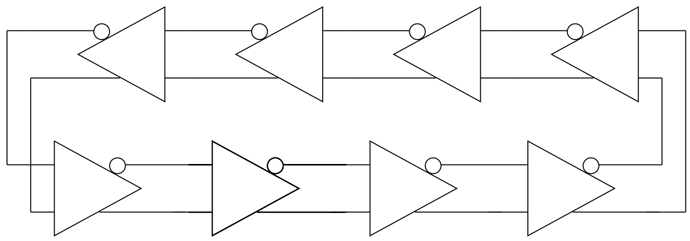


Figure 3.18: Ring Oscillator VCO

switches to one branch, the maximum swing across the output of the delay cell is limited to be equal to the voltage across the diode $V_d$, which is approximately a constant.

Using a first order approximation, the delay of one stage is given by:

$$t_d = \frac{V_d C_L}{I_{ss}} \tag{3.36}$$

where $C_L$ is the capacitive load of the output node. Thus the frequency of oscillation for a N-stage ring oscillator is:

$$F_o = \frac{I_{ss}}{2NV_d C_L} \tag{3.37}$$

which shows at first order a linear dependency with $V_{tune}$. For the given HP $1\mu m$ technology, it was found by simulation that 8 cells are needed to give the desired VCO gain. The VCO frequency-to-voltage characteristic is linear at first order, provided that the NMOS bias transistor is in the triode region of operation. For low values of $V_{tune}$ the transistor is in saturation and (3.35) is not valid. The necessary condition for oscillation is that the gain $a_v$ of each cell is "high enough". Typically, $a_v \simeq 1.5$ guarantees oscillation.

The bias circuit parameters $W_{nb}, L_{nb}, W_{pb}, L_{pb}$ do not affect the VCO performance as long as the current mirrors are in saturation and have a high output impedance. The parameters that affect the VCO performance are $W_n, L_n, W_p, L_p$. Since the objective of the optimization is to minimize power dissipation, parameter $L_p$ can also be set to the minimum possible value so that the load capacitance is minimized and the current $I_{ss}$ needed to achieve a certain delay is smaller according to (3.37). The transistor sizes selected for the bias circuit and $L_p$ are given in Table 3.10. The remaining parameters can be obtained by circuit-level optimization.

| $W_{nb}$ | $L_{nb}$ | $W_{pb}$ | $L_{pb}$ | $L_p$ |
|---|---|---|---|---|
| $10\mu m$ | $1\mu m$ | $20\mu m$ | $1\mu m$ | $1\mu m$ |

Table 3.10: VCO Pre-selected Parameters

#### 3.5.1.3 VCO Optimization Taking into Account Parasitics

To ensure that the performance constraints are met after the layout is generated, it is critical that layout parasitics are taken into account during the optimization phase. In our specific problem, the optimization algorithm attempts to adjust $I/C_{out}$ in order to achieve the desirable frequency performance, while minimizing power dissipation. This will result in the use of minimum size transistors, so the final circuit will be extremely sensitive to layout parasitic capacitances. This effect was observed in preliminary optimizations. To avoid this problem an optimization strategy taking into account parasitics was developed.

Let $\mathbf{P}$ be the performance vector, $\mathbf{C}$ the parasitics vector and $\mathbf{\Delta P_{max}}$ the corresponding maximum allowed performance degradation due to those parasitics. Assuming a linear model around the nominal performance and small parasitics, the performance degradation $\Delta P_i$ is given by:

$$\Delta P_i = \left[\mathbf{S}_{\mathbf{C}}^{P_i}\right]^T \cdot \mathbf{\Delta C} \tag{3.38}$$

where

$$\left[\mathbf{S}_{\mathbf{C}}^{P_i}\right]^T = \left[ \left.\frac{\partial P_i}{\partial C_1}\right|_{C_1=C_{1\,NOM}}, \left.\frac{\partial P_i}{\partial C_2}\right|_{C_2=C_{2\,NOM}}, \cdots, \left.\frac{\partial P_i}{\partial C_{N_C}}\right|_{C_{N_C}=C_{N_C\,NOM}} \right] \tag{3.39}$$

and $\mathbf{\Delta C}$ is the deviation from the nominal value of the parasitics. Given an estimate $\mathbf{\Delta C_{max}}$ of the maximum deviations from the nominal parasitics, the following constraint can be used to ensure normal operation in the presence of worst-case parasitics:

$$\left[\mathbf{S}_{\mathbf{C}}^{\mathbf{P}}\right]^T \cdot \mathbf{\Delta C_{max}} \leq \mathbf{\Delta P_{max}} \tag{3.40}$$

In the VCO optimization problem, the performance degradation constraints are evaluated at a maximum deviation of 50% from a nominal estimate of $15fF$ for the parasitics at the output of the VCO cell. Another two circuit specific constraints were added regarding the minimum and maximum output voltages. Those constraints are necessary for the level restoring circuit to work. The remaining constraints are obtained from the PLL optimization results in Tables 3.9, 3.7. Constraints on the voltage-to-frequency gain and the center

| *Parameter name* | *Parameter description* | *Constraint Value* |
|---|---|---|
| $F_{max}$ | output at $V_{in} = 4.8\,V$ | $\leq$ 173 MHz<br>$\geq$ 163 MHz |
| $F_{min}$ | output at $V_{in} = 1.8\,V$ | $\leq$ 58 MHz<br>$\geq$ 38 MHz |
| $\Delta\tau_{VCOrms}$ | rms VCO jitter at 140 MHz | $\leq$ 3 psec |
| $V_{max}$ | maximum output voltage | $\geq 0.5V$ |
| $V_{min}$ | minimum output voltage | $\leq 0.1V$ |
| $\frac{\Delta F_{max}}{F_{max}}$ | maximum variation of output due to layout parasitics | $\leq 10\%$ |

Table 3.11: VCO Optimization Constraints

frequency, can be converted to constraints for a minimum and a maximum VCO frequency. Table 3.11 summarizes the constraints for the VCO optimization problem. The objective of the low-level optimization problem is to minimize power.

The overall optimization problem for the VCO can be expressed as:

$$minimize\ \ Power_{VCO}(W_n, L_n, W_p) \tag{3.41}$$
$$subject\ to\ \ F_{maxVCO}(W_n, L_n, W_p) \leq F_{maxmax} \tag{3.42}$$
$$F_{maxVCO}(W_n, L_n, W_p) \geq F_{maxmin} \tag{3.43}$$
$$F_{minVCO}(W_n, L_n, W_p) \leq F_{minmax} \tag{3.44}$$
$$F_{minVCO}(W_n, L_n, W_p) \geq F_{minmin} \tag{3.45}$$
$$\Delta\tau_{VCOrms}(W_n, L_n, W_p) \leq \Delta\tau_{max} \tag{3.46}$$
$$V_{max}(W_n, L_n, W_p) \geq V_{max} \tag{3.47}$$
$$V_{min}(W_n, L_n, W_p) \leq V_{min} \tag{3.48}$$
$$\frac{\Delta F_{max}}{F_{max}} \leq P_{tol} \tag{3.49}$$

The optimization problem is solved using the supporting hyperplane algorithm described in 3.4.6. All constraints are evaluated using SPICE simulations except for the timing jitter constraint that is evaluated using (3.33). Since transistor sizes can only take discrete values, the results are truncated to the nearest feasible value, taking into account the technology specific parameters $\Delta L$, $\Delta W$, $\lambda = 0.6\mu m$. Table 3.12 summarizes the optimization results.

| | | Initial | Final |
|---|---|---|---|
| **Parameter name** | $W_n$ | 3.8 $\mu m$ | 2.6 $\mu m$ |
| | $L_n$ | 4 $\mu m$ | 4 $\mu m$ |
| | $W_p$ | 55 $\mu m$ | 36 $\mu m$ |
| **Performance variable** | $F_{max}$ | 165 MHz | 163 MHz |
| | $F_{min}$ | 50 MHz | 48 MHz |
| | $\Delta\tau_{VCOrms}$ | 1.25 psec | 1.5 psec |
| | $V_{max}$ | 0.67 V | 0.66 V |
| | $V_{min}$ | 0 V | 0 V |
| | $\frac{\Delta F_{max}}{F_{max}}$ | 6.8% | 10% |
| **Objective** | Power | 8.6 mW | 6.2 mW |
| **Iterations** | | 4 | |
| **CPU time** | | 2069 CPU sec | |

Table 3.12: VCO Optimization Results

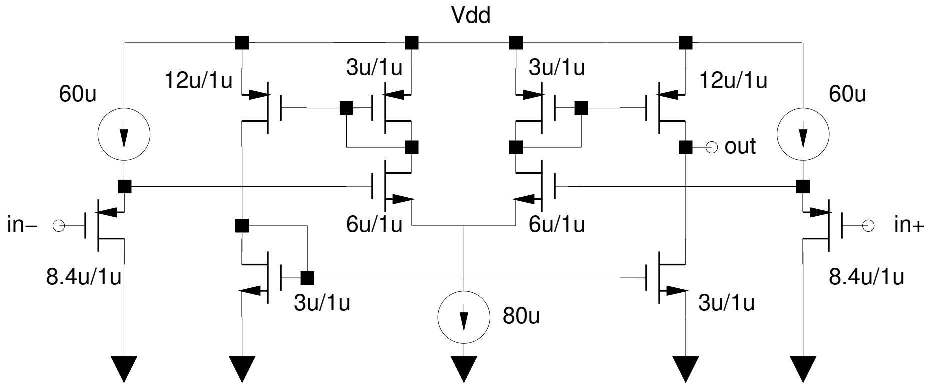



Figure 3.19: Level Restoring Circuit

### 3.5.2 Level restoring circuit

The specific VCO topology used, produces an $0 - 0.7V$ output signal, that has to be converted to full-swing CMOS logic levels. This task is performed by the level restoring circuit of Figure 3.19.

The differential signal is shifted by two level-shifters and then applied to a unity gain differential pair. The output of the differential pair is amplified to $0 - 5V$ and buffered by a buffer chain. The total static power dissipation of the level restorer is 1mW.

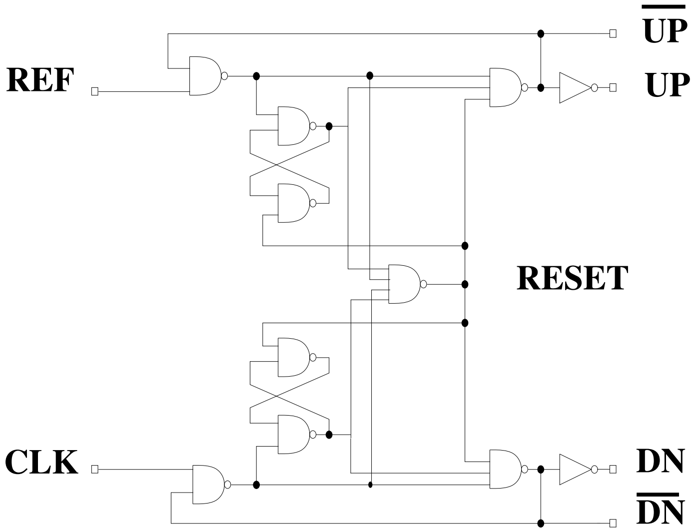



Figure 3.20: Phase-Frequency Detector

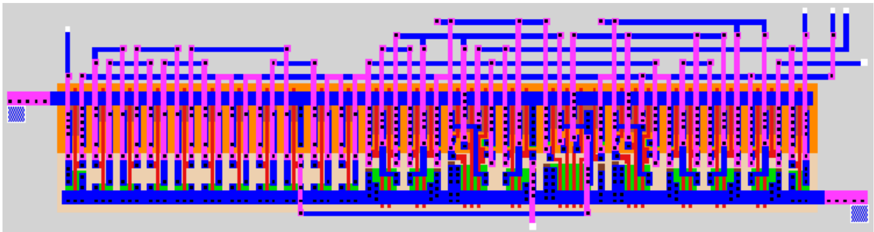


Figure 3.21: PFD Layout

### 3.5.3 Phase-Frequency Detector

The phase-frequency detector used is the one of Figure 3.20. The "dead zone" problem is avoided by adding delays in the loop [65]. The PFD generates two minimum-length UP and DOWN pulses, even when the compared waveforms are perfectly synchronized. Those pulses are identical in length, so the up and down currents cancel each other, assuming good matching. When the pulses are not perfectly synchronized, the difference between the pulses equals the phase difference between the compared clocks. The first order function of the PFD is identical to the one described in Figure 3.7. The PFD layout was automatically synthesized from the input schematic using OCTTOOLS [76] and is shown in Figure 3.21.

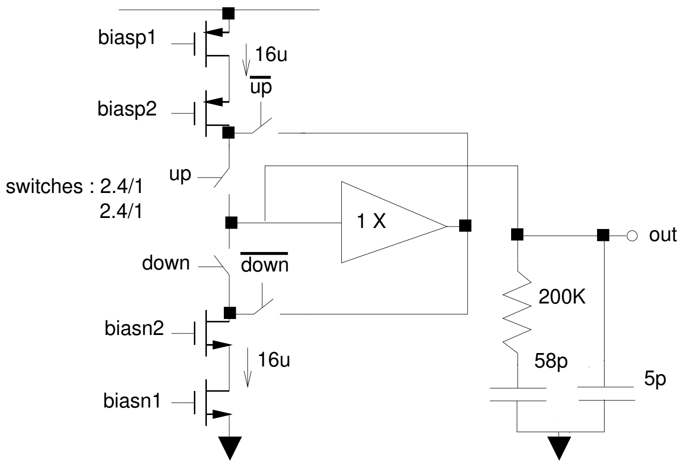



Figure 3.22: Charge-Pump Circuit

### 3.5.4 Charge Pump

The charge-pump used is shown in Figure 3.22 and is based on the design presented in [65]. Since the PFD eliminates the "dead zone" by turning both sources on at the same time, a current mismatch between the sources can inject extra noise to the output node. To avoid variations of the output current due to the output voltage, high-impedance cascode current sources are used. The cascode sources are biased from a high swing cascode bias circuit shown in Figure 3.23, which is based on [77]. To achieve good matching of the current sources, common-centroid layout is used.

When the pulse controlling a source is low, normally the source would be turned off. This may cause charge injection at the switching time and slow response. To avoid this problem, the currents are redirected to the output of a unity gain buffer kept at the same voltage as the output of the loop filter. This prevents the current fluctuations which are due to the finite output impedance of the current sources.

### 3.5.5 Dividers

The divider cell used is the synchronous programmable divider of Figure 3.24. It is basically a down-counter which resets to a pre-loaded integer after it reaches zero. To achieve fast operation, the true single-phase clock logic family (TSPC) is used. The basic D flip-flop element of the family is shown in Figure 3.25.

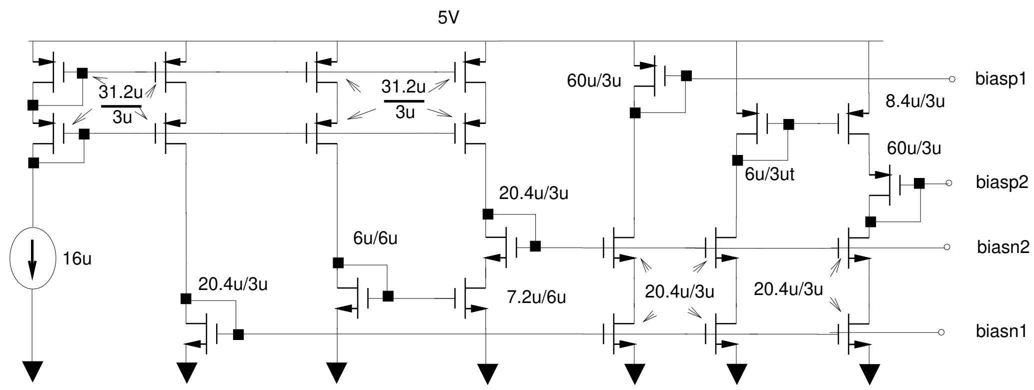



Figure 3.23: Charge-Pump Bias Circuit

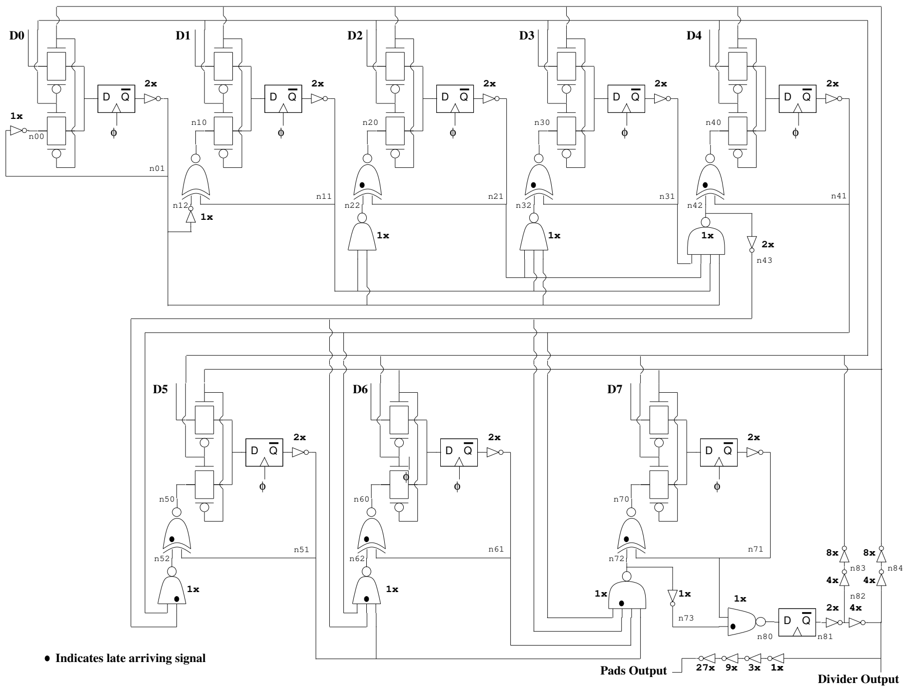



Figure 3.24: Programmable Divider

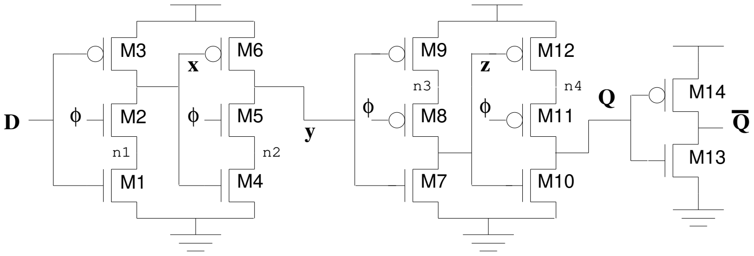



Figure 3.25: TSPC Flip-Flop

## 3.6 Physical Synthesis

### 3.6.1 Voltage Controlled Oscillator

The hierarchical, constraint-driven approach is also used at the layout phase for the VCO. The constraints set in the optimization problem of (3.41)-(3.49) are used in the layout generation. In addition to that, constraints for all parasitics are generated using the constraint generation techniques described in [78]. The sensitivities of every performance parameter with respect to every parasitic resistance and capacitance are calculated automatically and then, given a maximum allowable performance deviation, bounds are imposed on every parasitic using quadratic optimization.

A parametric layout generator was written for the specific VCO topology. It uses a fixed floor-plan and takes as parameters the number of delay cells, the device sizes and the constraints on the parasitics. The algorithm for the layout generation is shown in Figure 3.26. $\Delta W$ is the minimum increment ($\lambda$) allowed by the process design rules, $P_j$ is the performance $j$ and $i$ is the number of the parametric wires.

The final VCO layout is shown in Figure 3.27. The layout generation process in described in detail in [42].

### 3.6.2 PLL

Since parasitic constraints are not as critical for the other PLL sub-blocks, full custom layout was used. On-chip decoupling capacitors were placed between the supply and ground to minimize the effect of power supply coupling to the overall jitter performance. The final PLL system was routed using Mosaico [79] and the result is shown in Figure 3.28. The PLL area is approximately 0.45 $mm^2$ using the 1.0$\mu m$ HP CMOS technology with two

**begin**{
    **foreach**($P_j$)
    **foreach**($R_i, C_i$) calculate( $\frac{\partial P_j}{\partial C_i}, \frac{\partial P_j}{\partial R_i}$ );
**100:** calculate($R_{imax}, C_{imax}$); /* quadratic optimization*/
    **foreach**(i) {
    set $W_i = W_{imin}$ and $L_i = L_{imin} \implies C_i = C_0\, W_{min}\, L_{min}$ ;
    **do** {
        evaluate $R_i = \rho\, \frac{W_i}{L_i}$;
        **if** ($R_i < R_{imax}$) **then** exit;
        **else** $W_i = W_i + \Delta W$ ;
    } **while** ($C_i < C_{imax}$);
    **if** (($C_i > C_{imax}$) or ($R_i > R_{imax}$)) **then** "infeasible" ; $\implies$ **goto** 100;
    **else** "constraint enforced" ;
    }
}**end**

Figure 3.26: Layout Generation Algorithm

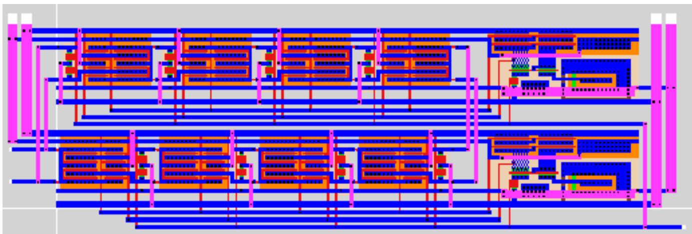


Figure 3.27: Ring Oscillator VCO Layout

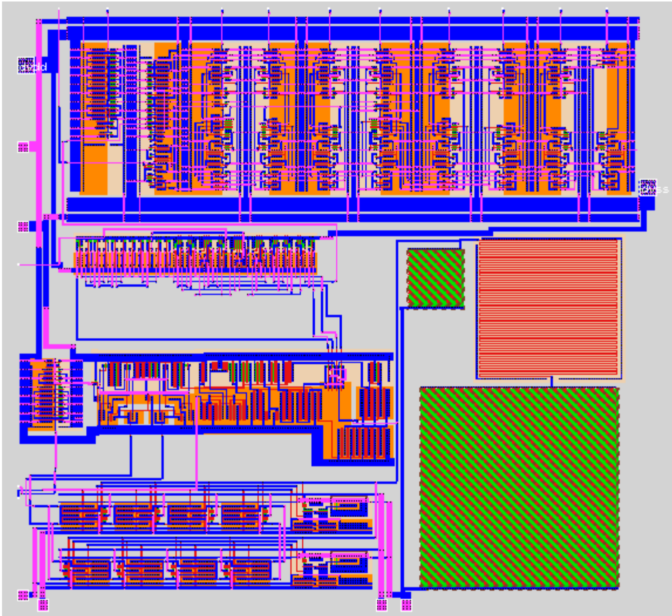


Figure 3.28: PLL Layout

metal layers and a linear polysilicon capacitor option.

### 3.6.3 D/A Converters

A module generator [80, 27] was used to directly synthesize the D/A layout from specifications. The result is shown in Figure 3.29.

### 3.6.4 Video Driver System

The final layout for the video driver system was assembled using Mosaico. It is shown in Figure 3.30. The die is approximately 3.4 $mm$ $x$ 3.9 $mm$ = 13.26 $mm^2$. Different analog and digital supplies were used and additional supplies were provided for the VCO in order to avoid as much as possible supply coupled noise.

## 3.7 Bottom-up Verification

The overall design was verified using hierarchical simulation. First the functionality of the lowest level sub-blocks, such as VCO, dividers, charge-pump, was verified using Spice and the parameters needed for the higher level blocks were extracted. The timing

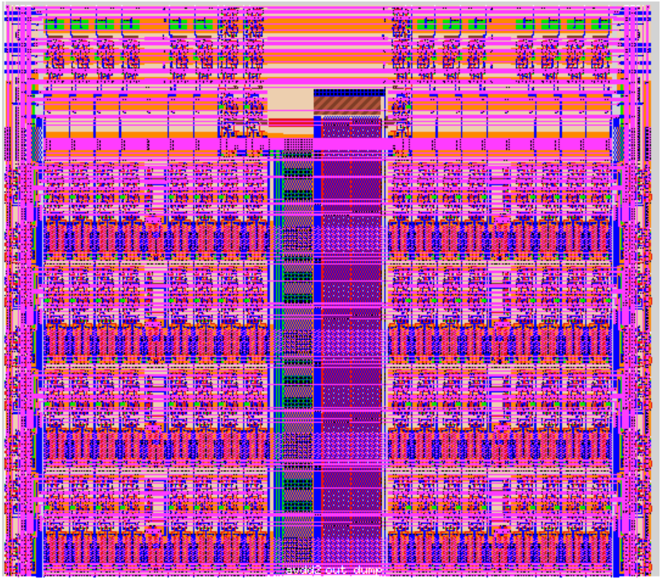


Figure 3.29: D/A Layout

jitter performance of the VCO was extracted using the non-Monte Carlo, nonlinear noise simulator described in [56]. After low-level parameters were extracted, behavioral simulation was used to verify the performance of the whole system.

The extracted parameters of the blocks used in the PLL behavioral simulation are summarized in Table 3.13. To illustrate the value of bottom-up verification, the result of a PLL full circuit simulation is compared to the result of the behavioral simulation in Figure 3.31. The waveform simulated is the control voltage of the VCO when the PLL is in acquisition mode and is performed in order to detect stability. It can be seen that the two waveforms are almost identical. The use of behavioral simulation based on careful modeling of second order effects, helped to significantly accelerate the verification process without noticeably affecting the accuracy of the simulation: 560 CPU seconds were needed for the behavioral simulation, while the full circuit simulation took 20 CPU hours [10]. Both simulations were performed on a DEC Alpha-Server 2100 5/250 with 256Mb of memory and 4 CPU's.

Figures 3.32, 3.33 show the result of a behavioral simulation for the timing jitter using the extracted parameters for $F_{out} = 100\ MHz$. The first plot shows the PLL and VCO rms timing jitter as a function of the distance from the reference transition. The

[10] using macro-models for the dividers

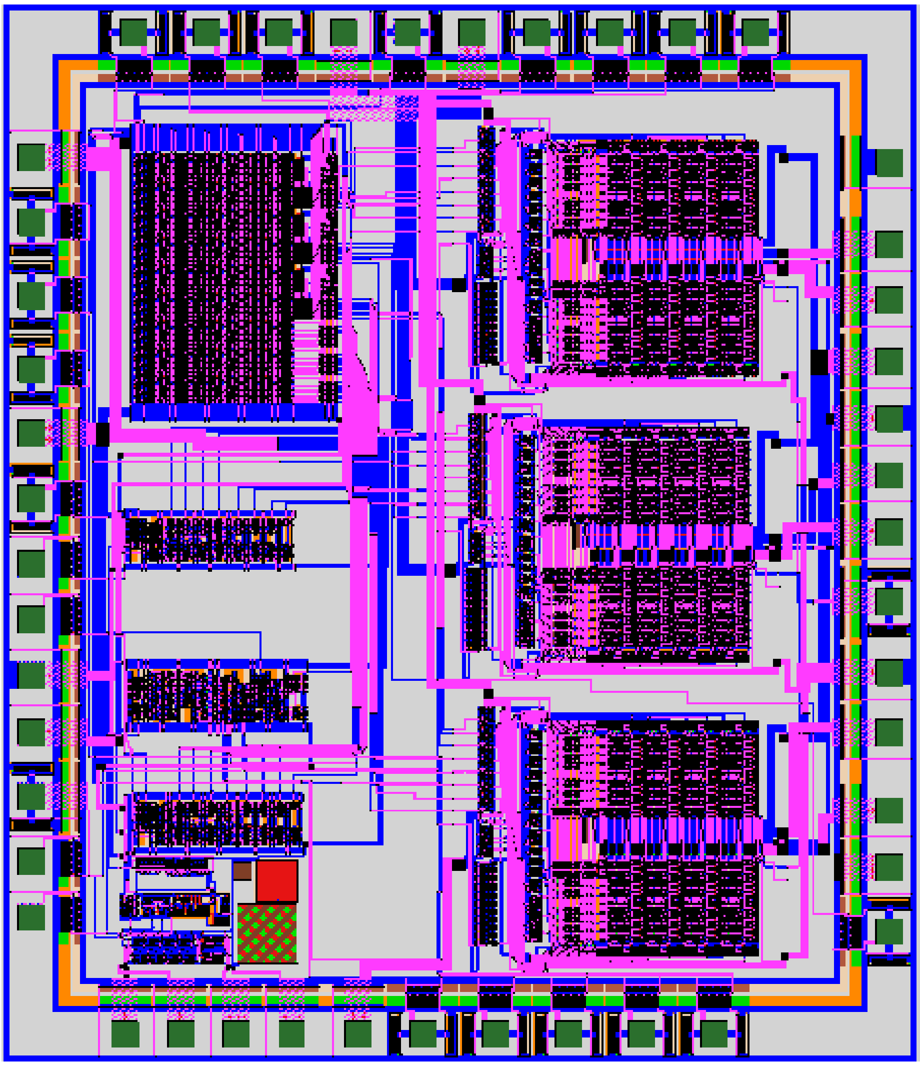


Figure 3.30: Video Driver System Layout

| *Module* | *Parameter name* | *Value* |
|---|---|---|
| **VCO** | $K_o$ | 39 MHz/V |
| | $F_o$ | 110 MHz/V |
| | $\Delta\tau_{VCOrms}$ | 4 psec |
| | $V_{min}$ | 1.4 V |
| **Charge-pump** | $I_p$ | 16 uA |
| | $\frac{\Delta I_p}{I_p}$ | 10%(est) |
| | $R_{linup}$ | 20 $K\Omega$ |
| | $R_{satup}$ | 500 $M\Omega$ |
| | $V_{satup}$ | 4.7 $V$ |
| | $R_{linlo}$ | 13.3 $K\Omega$ |
| | $R_{satlo}$ | 2000 $M\Omega$ |
| | $V_{satlo}$ | 0.33 $V$ |
| **Loop Filter** | R | 200 $K\Omega$ |
| | $C_1$ | 58 pF |
| | $C_2$ | 5 pF |
| **PFD** | $t_{dead\_zone}$ | 0 |
| | $t_d$ | 2 nsec |
| **Divider** | $t_d$ | 2 nsec |

Table 3.13: PLL Extracted Parameters

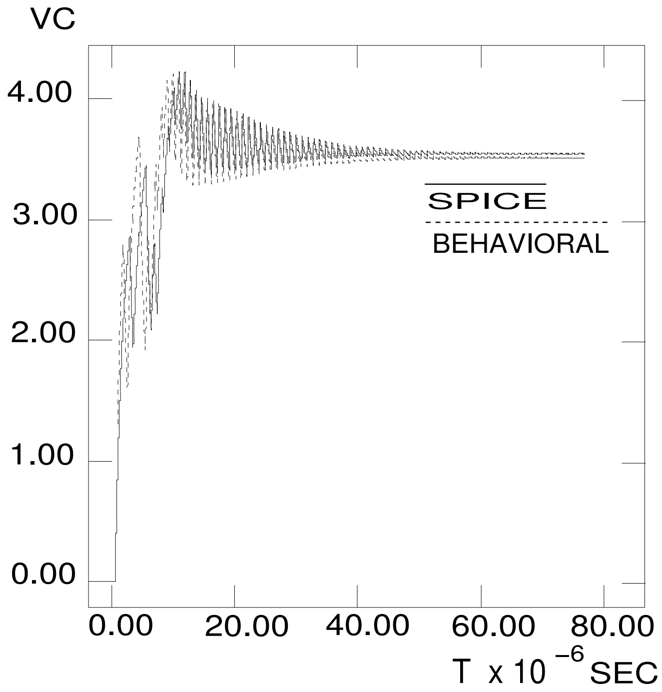



Figure 3.31: PLL Verification

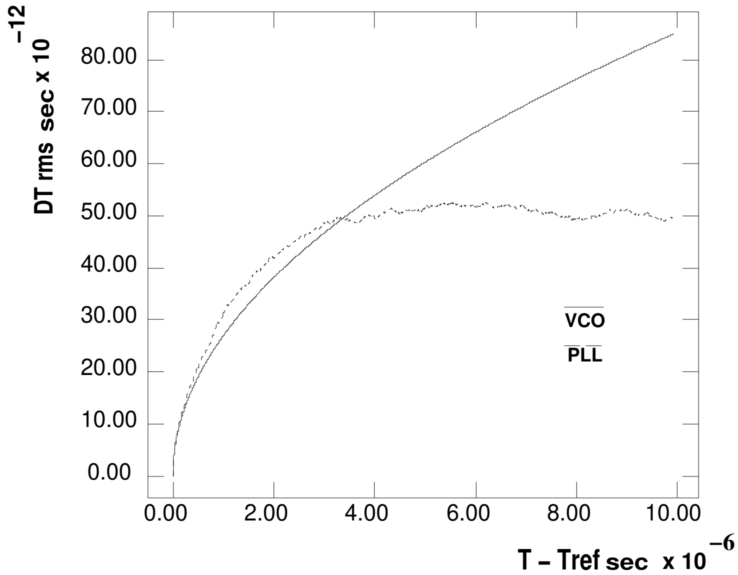



Figure 3.32: PLL Jitter Verification (a) $\Delta T_{rms}$

| *Type* | *PLL input Frequency* | *N-divider* | *VCO frequency* | *Constraint* | *Result* |
|---|---|---|---|---|---|
| Stability | 0.56 MHz | 100 | 56 MHz | ok | ok |
| | | 250 | 140 MHz | ok | ok |
| $\frac{\Delta T}{T}$ | 0.56 MHz | 250 | 140 MHz | $\leq 50\ psec$ | $47\ psec$ |
| Phase Margin | | | | $\geq 45^o$ | $45^o$ |

Table 3.14: PLL Verification Results

PLL timing jitter increases and finally converges to its final value, as expected. The second plot shows the square of the PLL and VCO rms timing jitter. As expected, $\Delta T_{VCO}{}^2$ accumulates linearly to infinity, since there is no correction from the PLL loop while the PLL jitter converges to a final value.

The results of the bottom-up verification are summarized in Table 3.14. The projected performance is based only on the calculation of the thermal jitter of the VCO, which sets the fundamental performance bound. Still, the performance of the actual system is expected to be close to the one predicted, since care has been taken to reduce as much as possible all other jitter sources.

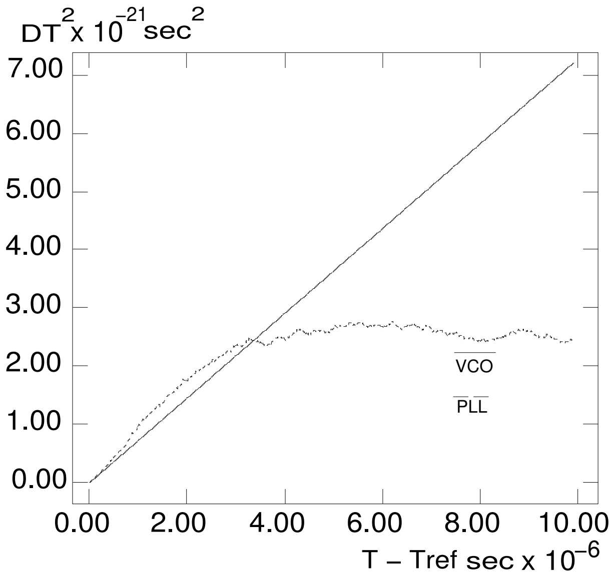



Figure 3.33: PLL Jitter Verification (b) $\Delta T^2$

## 3.8 Experimental Results

The chip was fabricated on a MOSIS HP 1.0$\mu m$ technology with two layers of metal and a linear polysilicon gate capacitor option. A die photo is shown in Figure 3.34. The 17,000 transistor system occupies an area of 3.4 $mm$ x 3.9 $mm$= 13.26 $mm^2$. A printed circuit board was designed and manufactured in order to measure the performance of the chip and is shown in Figure 3.35.

Experimental results show that the D/A INL and DNL performance is 0.16 LSB and 0.05 LSB respectively and that the settling speed requirements are also met ($T_{set}$ = 6 nsec). Figures 3.36,3.37 show experimental integral and differential nonlinearity (INL, DNL) data from six D/A converters as a function of the input code.

The PLL frequency generator meets the specifications for generating frequencies from 25 MHz to 130 MHz. Figure 3.39 shows the output waveform at 130 MHz. A small reduction of the maximum achievable speed is due to an error in the parameters file used in the synthesis phase. Detailed timing jitter measurements were done using a *Tektronix 11801B* high bandwidth digitizing oscilloscope with the same waveform feeding the signal and the trigger inputs. Figure 3.38 shows an output waveform at 100 MHz and the corresponding jitter histogram at a transition edge 7 $\mu s$ from the reference, so that the

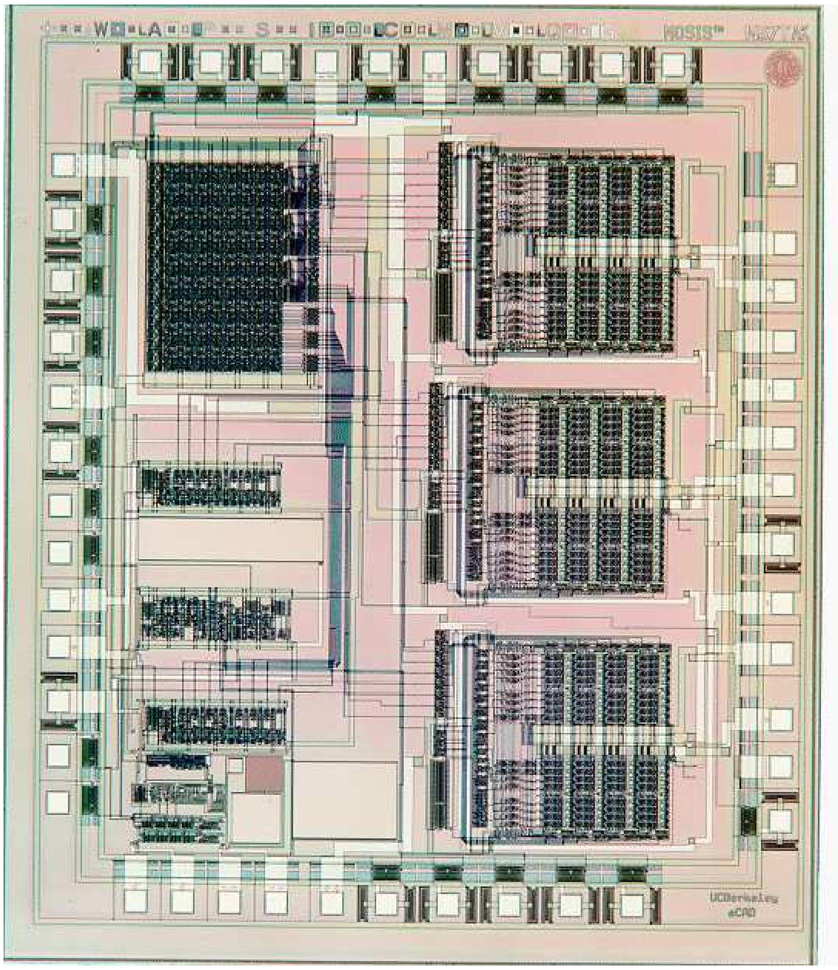



Figure 3.34: Video Driver System Die Photo

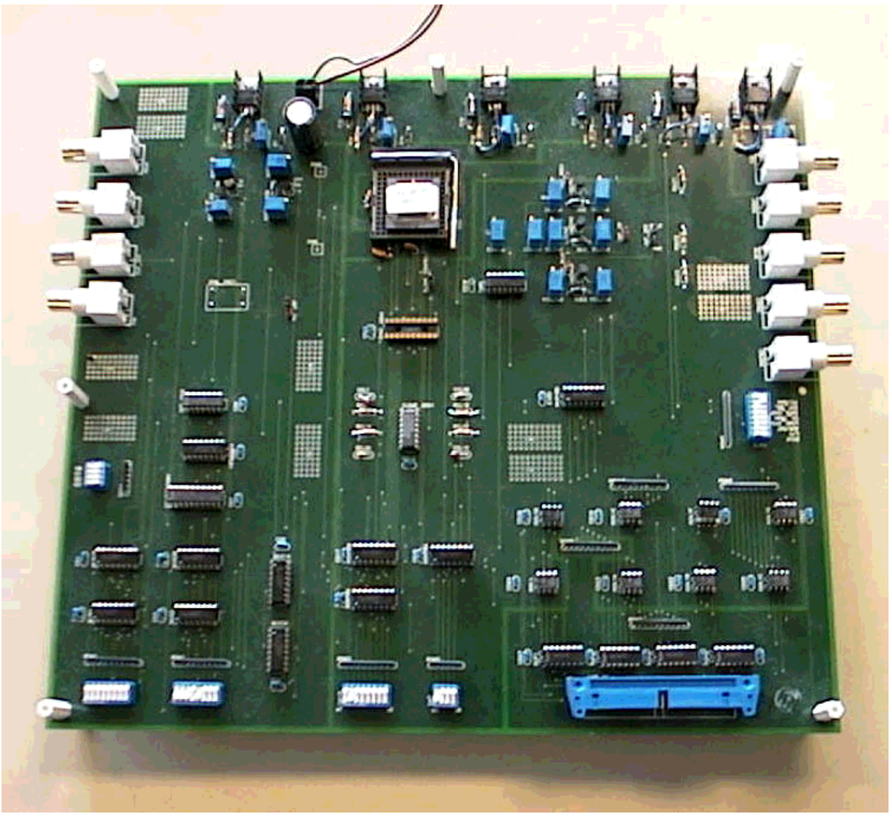


Figure 3.35: RAMDAC PCB Photograph

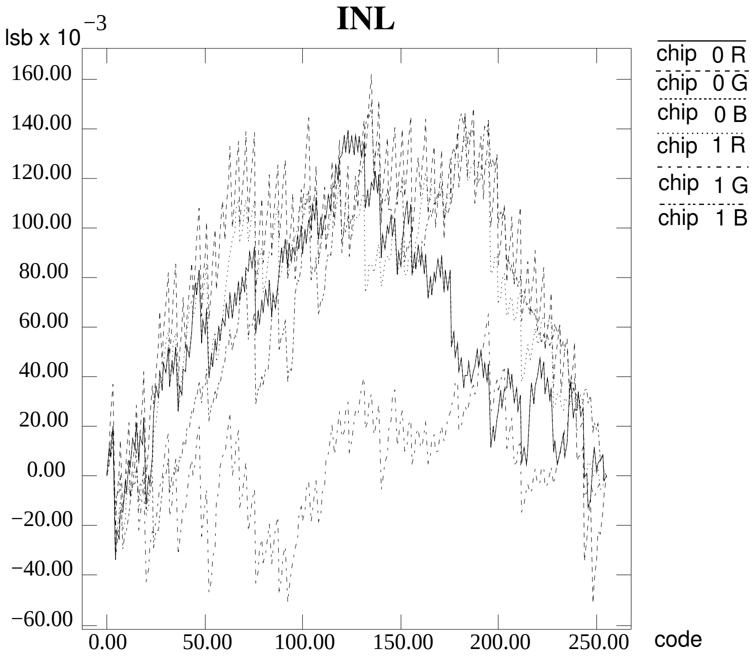



Figure 3.36: INL Measurement Results

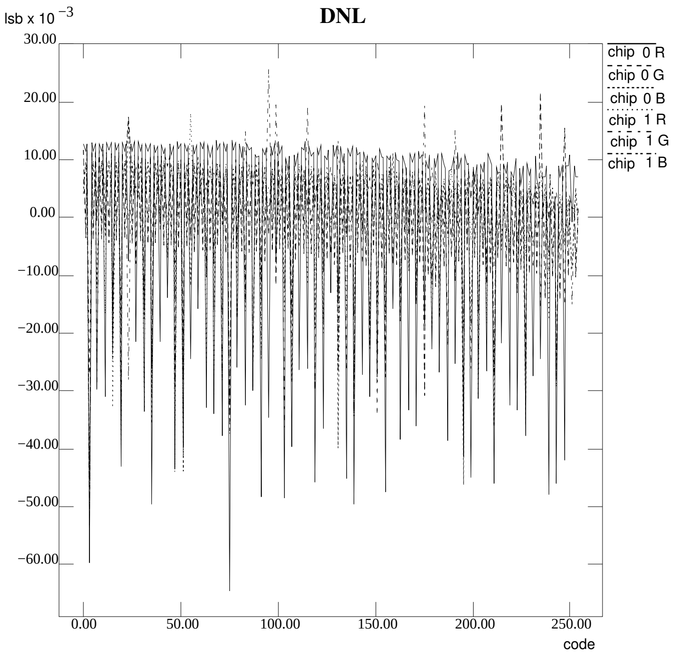



Figure 3.37: DNL measurement results

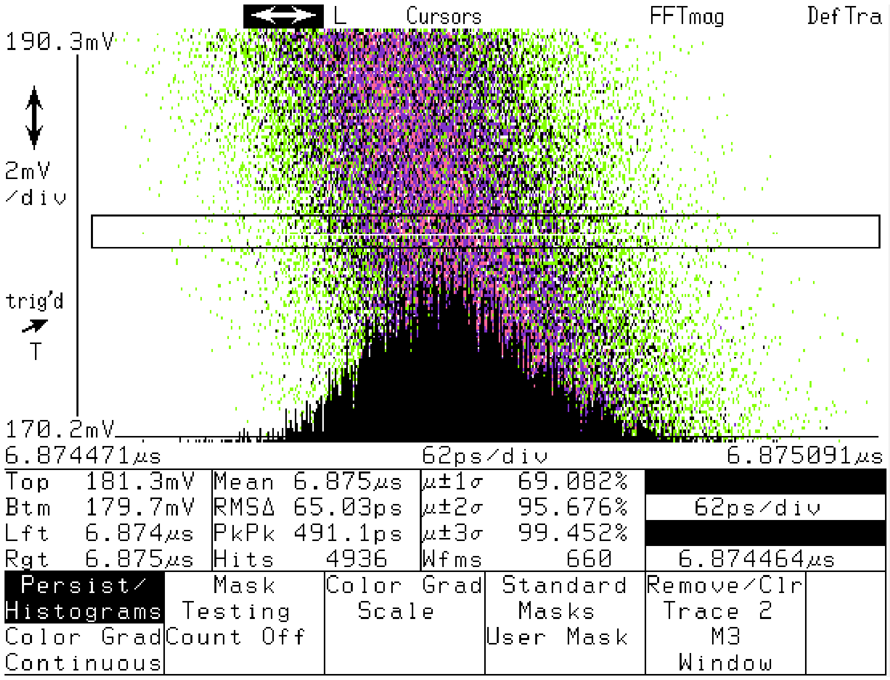



Figure 3.38: Jitter Histogram

accumulated jitter converges to its final value. The rms jitter at 100 MHz is *65 ps* (0.65 %), which is close to the specifications. The results are in agreement with predictions within 30 % for the worst case chip. Component process variations affecting the PLL bandwidth, simplified noise models for the devices, power supply and substrate coupling may cause the measured value to deviate from the predicted value. Furthermore, reflections and coupling from the testing board can significantly affect the measurements. Taking all those effects into account, the agreement between results and predictions is quite satisfactory.

The performance of the video driver is typical for similar chips [62] achieving comparable frequency range. The frequency range and timing jitter performance is comparable to microprocessor programmable clock generators [65]. The great advantage of the methodology was the acceleration of the design time. It is estimated that a total of nine months was needed from specifications to working fabricated parts. However, this design time also included the development of the methodology, tools and optimization techniques. A recent re-design of a PLL with different specifications to be used as RF local oscillator at $1900MHz$, took approximately two weeks from specifications to a sized circuit schematic [81]. This indicates the value of the methodology once the design flow is established.

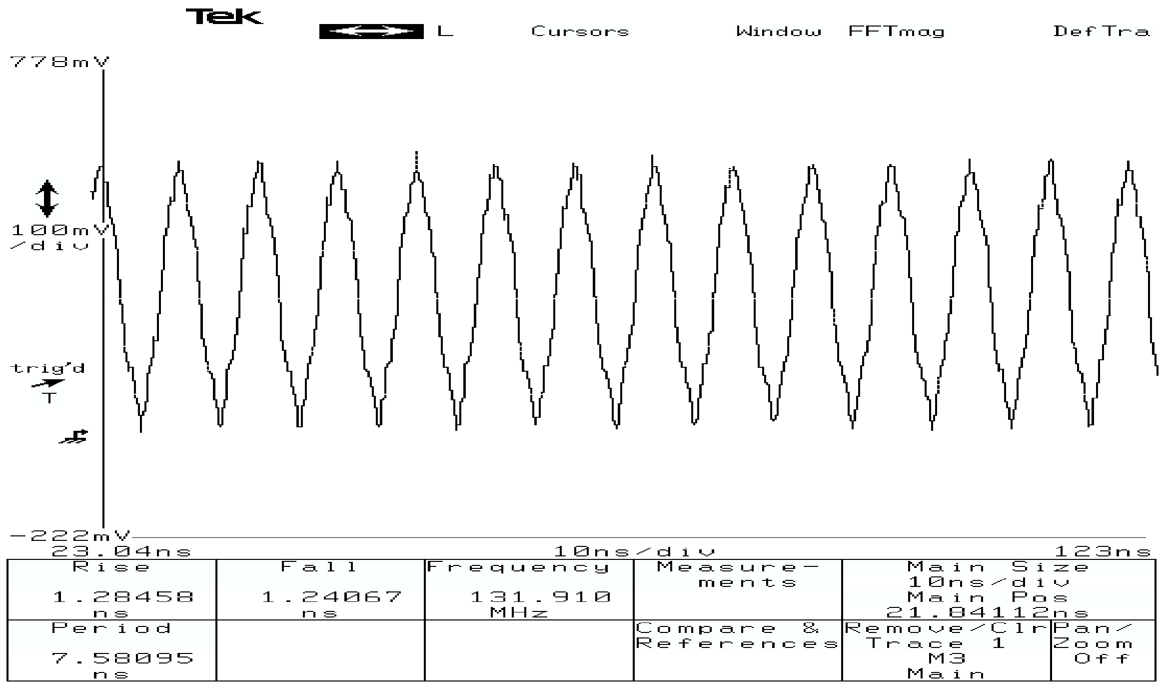



Figure 3.39: Frequency Synthesizer Output

## 3.9 Conclusion

A video driver system has been designed following the top-down, constraint-driven paradigm. To apply the methodology in the design of such a complex complex mixed-signal system, new behavioral modeling, simulation and optimization techniques had to be used and developed. Fundamental to the approach was the use of behavioral simulation and optimization for hierarchical constraint propagation.

Following the methodology, *a jitter constraint was imposed at the system level* and was propagated hierarchically all the way to the circuit blocks *using behavioral simulation.* Typically, jitter is ignored at the system level design and circuit techniques are used at the lowest level of the design hierarchy to minimize it, thus leading to over-design and costly design and fabrication iterations. The present approach combined with the tools used, can have a significant impact on the design of similar systems.

The chip was fabricated on an HP 1.0$\mu m$ technology via MOSIS. Detailed testing results show that the *specifications are met at the first fabrication run* . The performance of the video driver clock generator is typical for video drivers and programmable clock generators. The value of the methodology is proved by accelerating the design cycle in a complex industrial-strength chip. The overall design cycle together with the development of the necessary techniques is estimated to be about nine months. A recent re-design of the PLL with different specifications using the design flow established with this example, shows a great reduction of the overall design cycle.

The design of this system [11] expands the methodology to a higher level of complexity than the previous design examples [2, 3]. The video driver system system was also used to test a layout methodology using an extracted model of the substrate for optimal placement in [48].

[11] This work is also presented in [82].

# Chapter 4

# Behavioral Simulation for RF Systems

## 4.1 Introduction

The share of radio frequency (RF) circuits for mobile communications in the semiconductor market has dramatically increased over the past years. The design of RF systems is being done at high level in a non-systematic way, which often leads to costly design and fabrication iterations. While the analog/RF part of the system typically presents the biggest challenges and is the bottleneck of the design process, designers typically rely on ad-hoc techniques and simple analytical models to perform high-level estimation.

As a result, RF communication systems are ideal candidates for the expansion of the proposed top-down, constraint-driven design methodology. For an efficient methodology, it is crucial to perform quick explorations of system-level architectures using behavioral simulation and modeling. The desired model for an RF block must be a *black box* that captures all important first and second order effects and at the same time hides internal architectural details. The simulation engine must be independent of any particular model, so that it is possible to simulate different architectures in the same environment instead of having a different dedicated simulator for each specific architecture.

Figure 4.1 depicts schematically the hierarchical decomposition/methodology for an RF receiver system. To select an RF architecture, high-level performance and environmental constraints have to be mapped onto performance constraints of the lower level

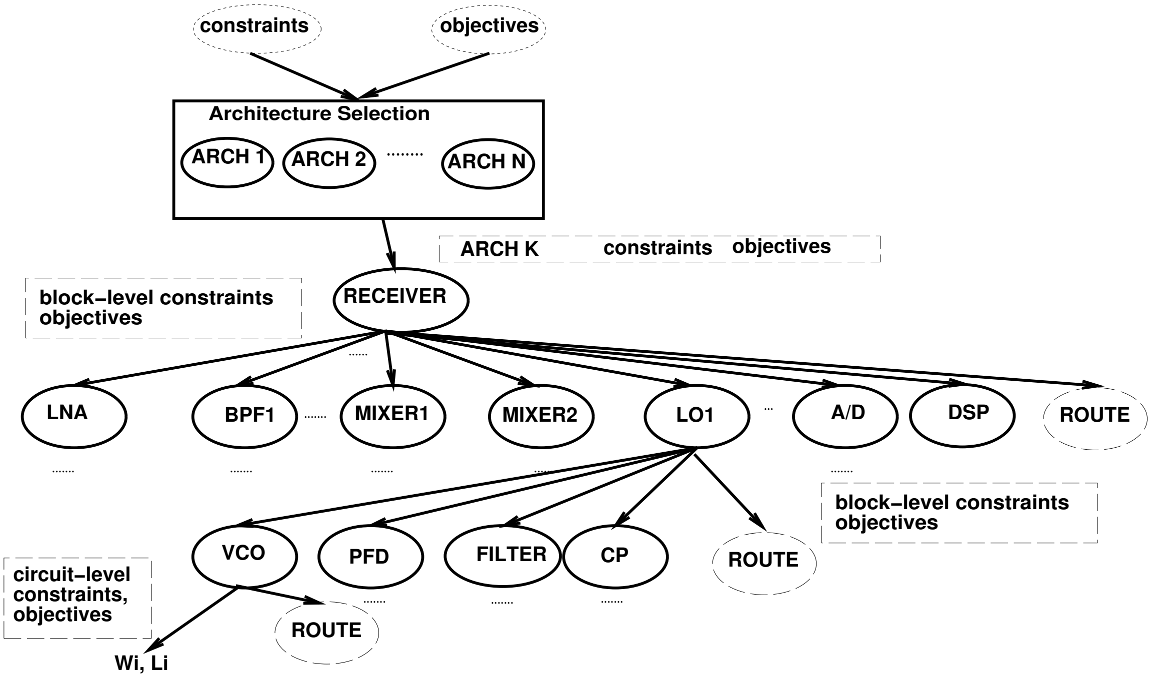



Figure 4.1: RF Receiver Hierarchical Design Decomposition

blocks. This process is continued recursively until the layout phase or until a library module is found that meets the specifications. In Figure 4.1, the design of a PLL (LO1) is shown as a lower level problem in the hierarchy of an RF system.

In RF systems, the typical figure of merit is the total signal-to-noise-and-distortion ratio (SNDR). Typically, specifications are given in the frequency domain as the desired SNDR in the presence of certain environmental conditions, such as out-of-band signals or in-band, co-channel interference and intermodulation. Typical performance metrics include gain or attenuation as a function of frequency, second and third order distortion, phase noise and noise figure (NF).

Once block-level characterization is obtained, either from specifications or from transistor-level simulations, analytical approximations [83] can be used to calculate the overall SNDR. Those approximations can be inaccurate when applied to systems with feedback or coupling, do not take into account phase and it is generally up to the designer to model correctly all the possible interactions between non-idealities. A more systematic and accurate approach could be to use macromodels in time-domain simulators. However, such models tend to add more implementation details than necessary and then again abstract only the necessary frequency domain performance. For example, the desired frequency-domain behavior of a filter has to be implemented with a specific pole-zero representation, which may result in very long simulation times. In addition to that, frequency-dependent nonlinearities are hard to model using this approach. Macromodels in available harmonic-balance simulators also rely on a time-domain representation of blocks.

To expand the applicability of the proposed methodology in the RF system design domain, a new technique to model and simulate RF systems is presented in this chapter. It uses a purely frequency-domain representation based on Volterra series models and harmonic balance simulation. Unlike previous approaches [84] that compute the high-order Volterra transfer functions from device equations, Volterra-based modeling is done only at the behavioral level, using pre-computed look-up tables to find the values for high-order Volterra transfer functions. Harmonic balance is used to solve arbitrary interconnects of blocks. Using those models, the Jacobian of the system can be swiftly evaluated and solved since there is no dependency on time-domain differential equation-based models.

In 4.2 the most common performance metrics that characterize RF systems and building blocks are discussed. In 4.3 other approaches to high-level modeling of RF systems are briefly presented. In 4.4 the theoretical background of the proposed approach is given. Finally, 4.5-4.8 present the proposed approach including experimental results.

## 4.2 RF Performance Metrics

Unlike digital systems, that can be generally described by the zero/one abstraction, there exist many different performance metrics characterizing different classes of analog and mixed signal systems. The most important characterizing RF systems are briefly presented here:

**s-parameters:** linear RF systems are often characterized as two-ports with an associated s-parameter matrix. Instead of voltages and currents, signals are represented as incident and reflected waves, $a_k(t)$ and $b_k(t)$, at each port:

$$\begin{aligned} a_k(t) &= \frac{1}{2\sqrt{r_k}}\left[v_k(t) + r_k i_k(t)\right] \\ b_k(t) &= \frac{1}{2\sqrt{r_k}}\left[v_k(t) - r_k i_k(t)\right] \end{aligned}$$

where $v_k$, $i_k$ the current and voltage at port $k$ and $r_k$ its characteristic impedance. With the scattering parameters $S_{ij}$ a two-port is described as:

$$\begin{aligned} B_1(f) &= S_{11}(f)A_1(f) + S_{12}A_2(f) \\ B_2(f) &= S_{21}(f)A_1(f) + S_{22}A_2(f) \end{aligned}$$

where $A_k(f), B_k(f)$ are the Fourier transforms of the incident and reflected waves at each port. With trivial matrix transformations S-parameters can be converted to typical voltage or current gains, transconductances, transimpedances and impedances.

**conversion gain:** it is defined for frequency converters as the ratio of the amplitude of the output at the output frequency to the input at the input frequency:

$$G_{conv}(f_{in}) = \frac{V_{out}(f_{in} \pm f_{LO})}{V_{in}(f_{in})} \tag{4.1}$$

where $f_{LO}$ is the frequency of the oscillator.

**Noise Figure (NF):** it is a measure of the thermal noise performance of a system. It is measured in dB and is defined as:

$$NF = 10 \log \left( 1 + \frac{\overline{V^2_{n_{add}}}}{\overline{V^2_{n_{in}}}} \right) \tag{4.2}$$

where $\overline{V^2_{n_{in}}}$ is the noise at the output due to the input noise and $\overline{V^2_{n_{add}}}$ is the noise at the output contributed by the system.

$V_{IP_3}$ **(input referred third order intercept point)** : is the voltage (or $P_{IP_3}$ for power) that has to be applied at the input so that the output due to the linear gain is equal in amplitude to the output due to the third order intermodulation ($IM_3$).

$V_{IP_2}$**:** same as above for the second order intermodulation term.

**Dynamic Range:** it is measured in dB:

$$DR = 10 \log \left( \frac{P_{max}}{P_{min}} \right) \tag{4.3}$$

where $P_{max}$, $P_{min}$ are the maximum and minimum signal powers that can be applied to the system due to distortion and noise.

$P_{-1dB}$ **(one dB compression point):** the input that has to be applied to the system so that the gain is compressed by one dB due to distortion.

**Phase Noise:** it is defined as the random fluctuation in the *phase* of an oscillator signal:

$$x(t) = Acos(\omega t + \phi + \phi_n(t)) \tag{4.4}$$

where $\phi_n(t)$ is the phase noise. The effect of phase noise is that the oscillator signal is not spectrally pure, thus the unwanted phase noise mixes with unwanted signals causing signal degradation due to interference at the desired band. It is typically measured in dBc/Hz:

$$P_{noise}(\Delta F) = 10\log\left(\frac{P_n(\Delta F)}{P_{carrier}}\right) \tag{4.5}$$

where $P_n(\Delta F)$ is the power spectral density of the noise at offset $\Delta F$ and $P_{carrier}$ the power of the carrier signal.

## 4.3 Approaches to Behavioral Modeling and Simulation of RF Communication Systems

### 4.3.1 Analytical Models

The simplest way to partition RF high-level constraints onto building block constraints is to use analytical formulas, for example the cascade formula for NF (Friis formula) and input-referred third-order intercept point ($V_{IP_3}$):

$$\frac{1}{V_{IP_3TOT}} = \frac{1}{V_{IP_31}} + \frac{G_1}{V_{IP_32}} + \ldots + \frac{G_1 \ldots G_{n-1}}{V_{IP_3n}} \tag{4.6}$$

$$NF_{TOT} = NF_1 + \frac{NF_2 - 1}{G_1} + \ldots + \frac{NF_n - 1}{G_1 \ldots G_{n-1}} \tag{4.7}$$

All possible interactions between external excitations and block-level non-ideal parameters such as parasitic gain paths, offset, phase noise, distortion, have to be considered. The total signal-to-noise-and-distortion-ratio (SNDR) would have to be adequate to give the desired bit error rate (BER). For example, if two strong signals are present at the input of a nonlinear block, $x_1 = |x_1|cos(\omega_1 t), x_2 = |x_2|cos(\omega_2 t)$, with $|x_1| = |x_2| = |x|$, the amplitude of the third order intermodulation component at the output at frequency $2\omega_1 - \omega_2$, $|x_{IM_3}|$, is given by:

$$|x_{IM_3}| = \frac{3}{4}a_3|x|^3 \tag{4.8}$$

where

$$a_3 = \frac{4}{3} a_1 {V_{IP_3}}^2 \tag{4.9}$$

$a_1$ is the linear gain and $V_{IP_3}$ is the third order intercept point. It is obvious that for many excitations, many combinations of inputs have to be considered.

The advantage of this method is its simplicity. The apparent drawbacks are that it ignores the specific nature of the non-idealities and that it oversimplifies the system by considering it a cascade of blocks. Also many effects are oversimplified or ignored, such as the raise of the noise floor due to a strong blocker, the phase cancellation of the intermodulation components etc.

The methodology for determining the block-level constraints of an RF system is the following:

1. find the worst case environmental constraints under which the system has to operate (minimum signal, minimum signal in the presence of worst case interference etc)

2. for each one of the above cases, find all possible interactions creating noise and interference; the total noise and interference should be less than the required to meet the BER constraint.

### 4.3.2 High-Level Description Language Macro-Models

A different method is to use macro-models in a time or mixed time-frequency domain simulator, such as SpectreRF [8, 9, 10] or Hspice [85]. This approach has the advantage that can use existing simulators and link with the lower level implementation. A significant disadvantage though is that more time-domain implementation-related details have to be added, thus reducing the generality of the models. For example, the "attenuation" specification has to be mapped onto a specific filter implementation with poles and zeros. Usually, RF systems have frequencies that are orders of magnitude apart; together with high-order filters, this can cause a time domain simulator to be extremely slow and have problems in convergence.

Figure 4.2 shows a macro-model of a typical RF building block developed in SpectreHDL [86], a high-level description language for analog circuits.

A complete macro-model for a mixer in SpectreHDL is given in Appendix B.

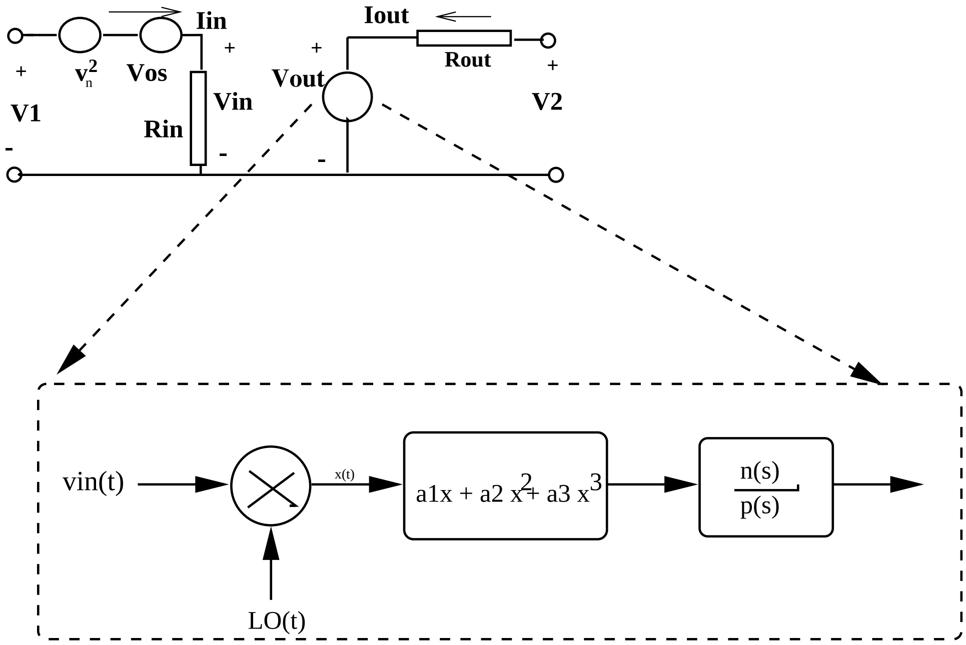



Figure 4.2: Mixer Behavioral Model in SpectreHDL

### 4.3.3 Communication System Static Data-Flow (SDF) Macro-Models

Another approach would be to co-simulate the RF block and the digital implementation using a data-flow, time-domain based system simulator such as Ptolemy [21] or Spw [37]. Simple RF time-domain macro-models could be used as in section 4.3.2

The most significant advantages of this method are:

- the exact effect of the interference characteristics to the overall BER can be simulated,
- digital DSP algorithms can be co-simulated with the RF architecture

The most serious disadvantages include:

- RF models are rather simplistic;
- an accurate simulation could be quite long, even with behavioral models, especially for systems with a low bit error rate (BER);
- time domain models are needed which have the disadvantages discussed in section 4.3.2.

## 4.4 Theoretical Background

Volterra series have often been used in the analysis of nonlinear circuits. In 4.4.1, a brief introduction to the Volterra theory of nonlinear systems is given. Practical reasons limit the application of Volterra series-based analysis to mildly nonlinear circuits. However, using the linear, time-varying system analysis presented in 4.4.2 combined with Volterra-series, can help model systems driven by strong, periodical nonlinearities, which is the typical case for RF circuits. Such systems are for example mixers, which are driven by strong periodical local oscillators, but the RF inputs can be viewed as a combination of weak harmonic signals, thus exciting a weakly nonlinear behavior from RF input to IF output [1]. This analysis is the basis for the behavioral modeling techniques for RF systems presented in 4.5. The simulation is based on *harmonic balance*, which is briefly introduced in 4.4.4. Since the performance of the new modeling technique is compared to SpectreRF, a brief introduction to *shooting methods* is presented in 4.4.5. Cyclostationary noise is discussed in 4.4.3, since the modeling techniques developed can be expanded to include noise modeling of periodically linear, time-varying systems.

### 4.4.1 Volterra Theory of Non-linear Systems

#### 4.4.1.1 Basic Principles

The Volterra theory [2] of nonlinear systems has often been applied to the analysis of nonlinear circuits. Typically, a nonlinear circuit can be described by the following equation (MNA analysis [89]):

$$\frac{dy(t)}{dt} - f(t, y(t)) + x(t) = 0 \tag{4.10}$$

where the term $\frac{dy(t)}{dt}$ represents the contribution of nonlinear and linear reactive circuit components such as capacitors, inductors, $x(t)$ represents the sources and $f(t, y(t))$ represents nonlinear and linear non-reactive components, such as resistors. If the functional $\phi = T[y]$ is defined such as:

$$\phi(t) = \frac{dy(t)}{dt} - f(t, y(t)) \tag{4.11}$$

[1] This does not apply when the input signal is "strong", as it may happen in RF circuits operating in compression, or in power amplifiers. Those cases may also be included in the models, using certain techniques discussed in 4.5

[2] This brief summary of Volterra series is based on the original work of Volterra [87] and the overview provided in [88].

provided that,

- $y(t)$ is a set of continuous functions with continuous first derivatives,
- $f(y(t), t)$ is a a set of continuous functions of $t$,

then the solution to $\phi(t) = 0$ is given by [88]:

$$y(t) = T^{-1}[\phi(t)] = F[\phi(t)] \tag{4.12}$$

Assuming that $\phi(t)$ is changed by an amount $\epsilon x(t)$ then the solution becomes:

$$y_\epsilon(t) = F[\phi(t) + \epsilon x(t)] \tag{4.13}$$

where $\epsilon$ is a parameter and $x(t)$ is a continuous function. If, $t$, $\phi(t)$, $x(t)$ are assumed to be fixed and $t$ is sufficiently small, $y_\epsilon(t)$ can be considered an ordinary function of $\epsilon$. If we assume that $y_\epsilon(t)$ has continuous derivatives up to $n$-th order with respect to $\epsilon$ on the interval $0 \leq \epsilon \leq b$, then Taylor's theorem can be applied:

$$y_\epsilon(t) = \left. y_\epsilon \right|_{\epsilon=0} + \left. \frac{\vartheta y_\epsilon}{\vartheta \epsilon} \right|_{\epsilon=0} \epsilon + \left. \frac{\vartheta^2 y_\epsilon}{\vartheta \epsilon^2} \right|_{\epsilon=0} \frac{\epsilon^2}{2!} + \ldots + \left. \frac{\vartheta^{n-1} y_\epsilon}{\vartheta \epsilon^{n-1}} \right|_{\epsilon=0} \frac{\epsilon^{n-1}}{(n-1)!} + \left. \frac{\vartheta^n y_\epsilon}{\vartheta \epsilon^n} \right|_{\epsilon=\theta} \frac{\epsilon^n}{n!} \tag{4.14}$$

where $0 \leq \theta \leq \epsilon \leq b$. From ( 4.13), $y_\epsilon(t)$ is the solution of:

$$\frac{dy_\epsilon(t)}{dt} = f(t, y_\epsilon(t)) + \phi(t) + \epsilon x(t) \tag{4.15}$$

that satisfies the initial conditions $y_\epsilon(0) = 0$. The first term of the solution $y_\epsilon(t)$ is the solution of $\phi(t) = 0$. The second term is $\epsilon$ multiplied by:

$$\left. \frac{\vartheta y_\epsilon}{\vartheta \epsilon} \right|_{\epsilon=0} = \lim_{\epsilon \to 0} \frac{F[\phi(t) + \epsilon x(t)] - F[\phi(t)]}{\epsilon} \tag{4.16}$$

which is the differential of $F$ with respect to $x(t)$ at $\phi(t)$. The higher order terms represent higher order differentials and the last term represents the error term of the Taylor approximation.

It has been shown by Volterra [87] that an $n$-order differential of a functional can be written as:

$$\delta^n F[\phi, x] = n! \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h_n(t, \tau_1, \tau_2, \ldots, \tau_n) x(\tau_1) x(\tau_2) \ldots x(\tau_n) d\tau_1 d\tau_2 \ldots d\tau_n \tag{4.17}$$

Equation (4.14) converges when the last term converges to zero. For a continuous function with continuous derivatives up to order $n$, this can always be guaranteed for any

$\epsilon$, if $x(t)$ is small enough. Substituting the value of functional differentials into the Taylor approximation (4.14) for $\epsilon = 1$ and assuming *time-invariance*, we get the familiar expression for the solution of (4.10):

$$\begin{aligned} y(t) = \lim_{n\to\infty} \quad & \int_{-\infty}^{\infty} h_1(\tau_1)x(t-\tau_1)d\tau_1 \\ & + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_2(\tau_1,\tau_2)x(t-\tau_1)x(t-\tau_2)d\tau_1 d\tau_2 \\ & + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} h_3(\tau_1,\tau_2,\tau_3)x(t-\tau_1)x(t-\tau_2)x(t-\tau_3)d\tau_1 d\tau_2 d\tau_3 \\ & + \ldots \\ & + \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty} h_n(\tau_1,\tau_2,\ldots,\tau_n)x(t-\tau_1)x(t-\tau_2)\ldots x(t-\tau_n)d\tau_1 d\tau_2\ldots d\tau_n \end{aligned} \tag{4.18}$$

In most practical cases the solution of a circuit problem can be assumed to satisfy the continuity and differentiability conditions stated above, thus (4.18) can describe the response of a nonlinear, time-invariant system. However, it should be noted that the series may actually diverge for large excitations $x(t)$, even though (4.10) may have a perfectly defined solution [88].

The first order term of (4.18) expresses the typical impulse response of a *linear, time invariant system.* The higher order terms express higher order nonlinearities. In general, this approach can be useful when the output can be approximated by a small number of terms:

$$y(t) = \sum_{n=1}^{N} y_n(t) \tag{4.19}$$

where

$$y_n(t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\ldots\int_{-\infty}^{\infty} h_n(\tau_1,\tau_2,\ldots,\tau_n)x(t-\tau_1)x(t-\tau_2)\ldots x(t-\tau_n)d\tau_1 d\tau_2\ldots d\tau_n \tag{4.20}$$

The functions $h_n(\tau_1,\tau_2,\ldots,\tau_n)$ are the *Volterra kernels.* Figure 4.3 shows schematically a system represented by Volterra kernels.

In the following section, a methodology for solving nonlinear circuits using Volterra series is presented. Even though Volterra series can be used to represent any nonlinear system meeting the conditions described above, complexity limits their application to weakly nonlinear circuits that can be approximated by a small number of terms.

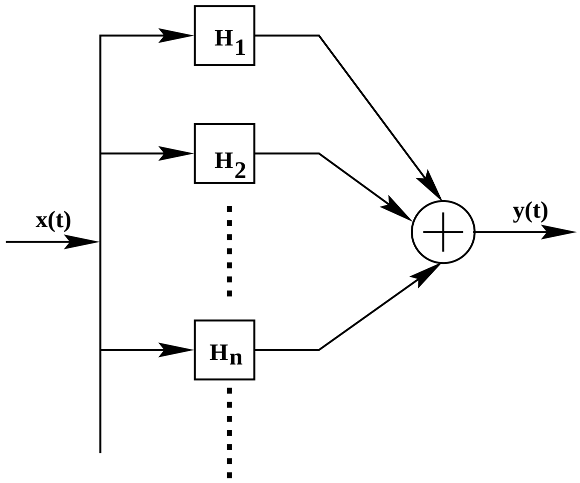



Figure 4.3: System Represented by Volterra Kernels

#### 4.4.1.2 Sinusoidal Analysis of Weakly Non-linear Systems Using Volterra Series

The Volterra series representation has been extensively used to analyze systems with weak nonlinearities with *memory* [84, 90, 91, 92]. Due to the complexity of the analysis, the technique is usually limited to small circuits.

To analyze a system using Volterra series, it is typically assumed that the excitations are sinusoidal:

$$x(t) = \sum_{q=1}^{Q} |E_q| \cdot cos(2\pi f_q t + \theta_q) = \frac{1}{2} \sum_{q=-Q}^{Q} E_q \cdot e^{j2\pi f_q t} \tag{4.21}$$

where $E_{-q} = E_q^*$.

If we define the $n$-th order transfer function as the $n$-dimensional Fourier transform of the $n$-th order Volterra kernel, we have:

$$H_n(f_1, \ldots, f_n) = \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} h(\tau_1, \ldots, \tau_n) \cdot e^{-j2\pi(f_1\tau_1 + \ldots + f_n\tau_n)} d\tau_1 \ldots d\tau_n \tag{4.22}$$

Taking the $n$-dimensional Fourier transform of (4.20) and using (4.21), the $n$-th order response of the system is:

$$y_n(t) = \frac{1}{2^n} \sum_{q_1=-Q}^{Q} \cdots \sum_{q_n=-Q}^{Q} E_{q_1} \ldots E_{q_n} H_n(f_{q_1}, \ldots, f_{q_n}) e^{j2\pi(f_{q_1}+\ldots+f_{q_n})t} \tag{4.23}$$

Thus the output is the sum of all responses up to $n$-th order and each $n$-th order response is the sum of all possible combinations of the input frequencies $-f_Q, \ldots, -f_1, f_1, \ldots, f_Q$ taken $n$ at a time and multiplied by the $n$-th order transfer function.

For a real, causal system, it can be trivially proven that:

$$H_n(-f_1, \ldots, -f_n) = H_n{}^*(f_1, \ldots, f_n) \tag{4.24}$$

where $H_n{}^*$ denotes the conjugate of $H_n$.

In (4.23), a particular combination of input frequencies appears in all possible $(n; m_{-Q} \ldots m_Q)$ orderings:

$$(n; m_{-Q} \ldots m_Q) = \frac{n!}{(m_{-Q}!) \ldots (m_{-1}!)(m_1!) \ldots (m_Q!)} \tag{4.25}$$

where $m_q$ is the number of times that frequency $f_q$ appears in the combination.

It can be proven [93] that, without loss of generality, nonlinear transfer functions can be assumed to be symmetric. If $h_n^*(\tau_1, \tau_2, \ldots, \tau_n)$ is a non-symmetric Volterra kernel, the value of the integral in (4.20) does not change if any of $\tau_1, \ldots, \tau_n$ are interchanged because the limits of the integration are from $-\infty$ to $\infty$. So if we define the *symmetric* kernel:

$$h_(\tau_1, \tau_2, \ldots, \tau_n) = \frac{1}{p!} \sum_{\substack{\text{all possible} \\ \text{rearrangements of } \tau_i}} h_n^*(\tau_1, \tau_2, \ldots, \tau_n) \tag{4.26}$$

then the corresponding system has the same response as the original system. Thus, in the following discussion, higher order transfer functions will be assumed symmetric.

To analyze a weakly nonlinear circuit using Volterra series, first the Volterra transfer functions $H_n$ are computed, using the *harmonic input method* [84]. Then (4.23) is used to compute the total response at the desired node.

The *harmonic input method* is as follows:

- First apply a single exponential input, $e^{j2\pi f_1 t}$, and determine the *first order* transfer function using the fundamental circuit equations, KCL, KVL and device equations.

- Then, apply the sum of two *incommensurate* exponentials, $e^{j2\pi f_1 t} + e^{j2\pi f_2 t}$, and compute $H_2(f_1, f_2)$ from the response at frequency $f_1 + f_2$ using $H_1(f)$

- Finally, recursively compute all the orders up to $n$, applying a sum of positive, incommensurate frequencies to the system in order to get $H_n(f_1, f_2, \ldots, f_n)$, using all $H_q, q < n$.

This method is applied whenever the circuit components are characterized by a differential equation model. Since the $n$-th order transfer functions have to be computed for all possible combinations of input frequencies, the complexity increases dramatically [3] with the order of nonlinearity and circuit size. This limits the application of the method to small, mildly nonlinear circuits [4] with few harmonic excitations.

### 4.4.2 Linear Time-Varying (LTV) Systems

As discussed in the previous section, due to its complexity, Volterra series-based analysis can only be applied if the system is assumed to be *weakly nonlinear*. However, in RF communication systems, basic building blocks such as *frequency converters* are strongly nonlinear. In a system such as a mixer, the local oscillator excites strong nonlinearities, while the signal input can be considered as a small signal exciting linear behavior from RF to IF.

Such systems can be modeled as *linear, time-varying* systems with the following input-output relationship [94]:

$$y(t) = \int_{-\infty}^{\infty} h(t;\tau)x(\tau)d\tau_1 \tag{4.27}$$

where the impulse response is time-varying with $t$. In such systems the strong excitation is typically periodical so $h(t;\tau)$ varies periodically and can be expanded into a Fourier series [94]:

$$h(t;\tau) = \sum_{q=-\infty}^{\infty} h^q(\tau) \cdot e^{j2\pi q f_0 t} \tag{4.28}$$

where $f_0$ is the frequency of the periodically varying strong excitation. Taking the Fourier transform of $h(t;\tau)$, the *periodically time-varying* transfer function, $H(t,f)$, can be expanded to a Fourier series [94]:

$$H(t,f) = \sum_{q=-\infty}^{\infty} H^q(f) \cdot e^{j2\pi q f_0 t} \tag{4.29}$$

[3] approximately $O(q^n)$, where $n$ is the order of nonlinearity and $q$ the number of input harmonics

[4] usually up to third order

where $H^q(f)$ is the q-th *sideband* transfer function of the *linear, time-varying system.* A sideband transfer function has the same effect with a linear transfer function but at the same time shifts the input spectrum by $q \cdot f_0$.

Assuming sinusoidal excitations as in (4.21), the output of a such a system becomes:

$$y(t) = \sum_{i=-Q}^{Q} \sum_{q_i=-\infty}^{\infty} H^{q_i}(f_i) \cdot E_i \cdot e^{j2\pi(f_i+q_i f_0)t} \tag{4.30}$$

Equation (4.30), denotes that the output of the LTV system is the sum of all possible shifts of the input harmonics multiplied by the appropriate sideband transfer function.

### 4.4.3 Cyclo-Stationary Noise Analysis

Noise in strongly nonlinear systems is generally a non-stationary process. The autocorrelation matrix of a non-stationary process can be defined as:

$$R_{xx}(t_1, t_2) = E\left[x(t_1)x^T(t_2)\right] \tag{4.31}$$

If $y$ is the output of an LTV system as in (4.27) and $R_{xx}(t_1, t_2)$ the input autocorrelation spectrum defined in (4.31), then:

$$R_{yy}(t_1, t_2) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} h(t_1, \tau_1) R_{xx}(\tau_1, \tau_2) h^T(t_2, \tau_2) d\tau_1 d\tau_2 \tag{4.32}$$

If the excitations of the system are periodical with period $T$, it can be assumed [13] that input and output noise of the system is *cyclo-stationary* [95, 96]. A random variable $x$ is cyclo-stationary if:

$$R_{xx}(t_1 + T, t_2 + T) = R_{xx}(t_1, t_2) \tag{4.33}$$

If (4.33) is satisfied, then the $R_{xx}(t_1, t_2)$ can be expressed as:

$$R_{xx}(t_1, t_2) = \sum_{l=-\infty}^{\infty} R_{xx_l}(t_2 - t_1) e^{jl\omega_0 t_2} \tag{4.34}$$

The Fourier transforms $S_{xx_l}(\omega)$ of the *harmonic covariances* $R_{xx_l}$ are the *harmonic* power spectral densities of the cyclo-stationary random variable.

It can be proven [13] that, when a cyclo-stationary noise source with a periodic autocorrelation function $R_{SS}(t, \tau) = \sum_{i=-\infty}^{\infty} R_{x_i}(\tau) e^{ji2\pi f_0 t}$ passes through a linear time-varying system with period $\omega_0$ as in (4.30), the cyclo-stationary components of the output spectrum are given by:

$$
\begin{aligned}
S_{xx_l}(-\omega) = \sum_{k=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} \quad & H_i(\omega - i\omega_0) S_{uu_k}(-\omega + i\omega_0) \cdot \\
& H^T_{l-i-k}(-\omega + (i+k)\omega_0) \qquad (4.35)
\end{aligned}
$$

where $S_{uu_k}$ is the $k-th$ cyclo-stationary component of the spectrum.

In [13] an efficient technique based on *harmonic balance* is used to compute the harmonic power spectral densities of a nonlinear circuit from network equations.

Recently, it has been demonstrated that when the periodic nonlinear excitations of a system contain phase noise, the cyclo-stationarity assumption for the outputs of the system is not valid. A detailed analysis can be found in [16].

### 4.4.4 Harmonic Balance Simulation Techniques

Harmonic balance [6] is a technique that has been extensively used to simulate nonlinear circuits in the frequency domain [97].

If we assume that:

- a system is described by a set of differential equation such as:

$$f(x, \dot{x}, u) = 0 \qquad (4.36)$$

- the excitation vector $u$ is periodic with period $T_0$,
- a solution $x$ exists and is periodic with period $T_0$ such as:

$$x(t) = \sum_{i=-\infty}^{\infty} X(i) e^{ji\omega_0 t}, \;\; \omega_0 = \frac{2\pi}{T}_0 \qquad (4.37)$$

then to find the solution, we substitute (4.37) in (4.36) and the resulting equation is written as a Fourier series:

$$f(x(t), \dot{x}(t), u(t)) = \sum_{i=-\infty}^{\infty} F(X, U, k) \cdot e^{ji\omega_0 t} \qquad (4.38)$$

where:

$$
\begin{aligned}
X &= [\ldots, X(-1), X(0), X(1), \ldots]^T \\
U &= [\ldots, U(-1), U(0), U(1), \ldots]^T \\
u(t) &= \sum_{i=-\infty}^{\infty} U(i) \cdot e^{ji\omega_0 t} \qquad (4.39)
\end{aligned}
$$

Finally the nonlinear algebraic system of equations

$$F(X, U, k) = 0, \forall k \tag{4.40}$$

is solved for $X$. To make the system solvable, the number of harmonics is typically truncated to a finite number $K_{max}$. Different approaches have been proposed to solve the nonlinear system: optimization [98], nonlinear relaxation [99] and Newton's method [7]. The convergence of harmonic balance algorithms can be problematic when highly nonlinear circuits are simulated.

### 4.4.5 Shooting Methods

Shooting [5] methods [8] have been extensively used to simulate RF circuits driven by $T$-periodic excitations. Assuming that the response of the system is also periodic with period $T$, the following boundary condition must be satisfied:

$$v(T + t_0) - v(t_0) = 0 \tag{4.41}$$

where $v(t)$ is the response of the circuit. If an initial state $v(t_0)$ is known, then transient analysis can be used to find the response at time $t_1 > t_0$. In general:

$$v(t_1) = v(t_0) + \phi(v(t_0), t_0, t_1) \tag{4.42}$$

where $\phi$ is the state transition function for the differential equation describing the circuit. Using (4.41), (4.42) the following nonlinear algebraic system is formulated:

$$\phi_T(v(0), 0) = 0 \tag{4.43}$$

where $\phi_T(v(t_0), t_0) = \phi(v(t_0), t_0, T + t_0)$. Equation (4.43) can be solved using standard Newton iteration methods. When applying Newton's method, the sensitivity of the final state $v(T)$ with respect to changes in the initial state $v(0)$ are computed, in order to correct the initial state to match $v(T)$. Transient simulation has to be used to evaluate numerically the state transition function $\phi$ over a period $T$.

The only restriction for applying shooting methods to a circuit is the assumption of $T$-periodicity of the response. Thus, there is a significant advantage over harmonic-balance in simulating circuits such as switching mixers, which require a large number of

[5]This summary is based on the overview of simulation methods for RF circuits of [17].

harmonics to represent the switching operation. However, the efficiency of the method can be degraded when the input of a system contains harmonics that are widely spaced. If the input contains two different harmonics of frequencies $f_1$, $f_2$ [6] where $f_1 \gg f_2$, the common period is the inverse of the maximum common denominator of $f_1$, $f_2$. This may significantly delay the numerical computation of $\phi$. Also a large amount of memory is required to save the waveform at all time-steps.

Using shooting methods, the periodically-varying steady-state solution of a circuit can be obtained. In circuits where a large periodic input (such as the local oscillator of a mixer) co-exists with small excitations (such as the RF inputs of a mixer), the periodically-varying operating point can be obtained using the large signal as input to the shooting method. Then, linearization around this point can be used to obtain the *sideband* transfer functions, as defined in 4.4.2. A very popular implementation of a shooting method which also includes this analysis, is the commercial simulator SPECTRERF [8].

## 4.5 Behavioral Modeling of RF Systems Using Volterra Series

### 4.5.1 Modeling of Deterministic Effects

For modeling systems with weak nonlinearities [7], one abstract approach that uses only high-level frequency-domain description, is to use the Volterra theory, described in 4.4.1. Weakly nonlinear blocks can be described as $N$-ports with a Volterra input-output frequency domain mapping function [100]. In addition, RF system specifications are given with the assumption of sinusoidal inputs to approximate the modulated signal, thus the input can be considered to be a sum of complex exponential functions as in (4.21).

Assume a system that has:

1. $N$-inputs, $N$-outputs ($N$-port),
2. weakly nonlinear behavior,
3. harmonic excitations $X = \sum_{k=1}^{K} X_k e^{j2\pi f_k t}$, where $X_k = [x_{1k}, \ldots, x_{Nk}]^T$ is the vector of port excitations at frequency $f_k$,

[6] For example: $f_2 = 1MHz$, $f_1 = 1GHz$, as it is common in RF systems with multiple mixers.

[7] Theoretically any nonlinear system can be described but the analysis becomes very complex, since many high-order terms have to be taken into account; see 4.4.1.2

4. output vector $Y$,

5. nonlinearities are important up to order $M$.

Using (4.23), the response of an $N$-port at port $n$ is:

$$y_n = \sum_{m=1}^{M} y_n^{(m)} \tag{4.44}$$

$y_n^{(m)}$ is the total response at port $n$, of frequency mixes of $m$-th order, given by:

$$\begin{aligned} y_n^{(m)} = \sum_{n_1,\ldots,n_m,k_1,\ldots,k_m} & H^{(m)}_{nn_1\ldots n_m}(f_{k1},\ldots,f_{km}) \cdot \\ & x_{n1}(f_{k1})\ldots x_{nm}(f_{km}) \cdot \\ & e^{j2\pi[\sum_{i=1}^{m} f_{ki}]t} \end{aligned} \tag{4.45}$$

where $H^{(m)}_{nn_1,\ldots,n_m}(f_{k1},\ldots,f_{km})$ is the value of the m-th order transfer function from ports $n_1,\ldots,n_m$ to port $n$ for input frequencies $f_{k1},\ldots,f_{km}$ and $x_{nm}(f_{km})$ is the coefficient of the input harmonic $f_{km}$ at port $x_{nm}$

This representation cannot model strong nonlinear behavior, for example the behavior of a mixer. However, excluding strong compression, in a typical RF receiver the signals driving the strong nonlinearities are periodical. Thus, the system can be modeled using the sideband transfer functions in (4.29).

While a mixer is a strongly nonlinear module, the input-output behavior from RF to IF is typically *weakly nonlinear* and thus can be represented with a Volterra input-output map as in (4.45). Thus, if each coefficient of (4.30) is expanded to a Volterra series, both periodical strong nonlinearities and weak nonlinearities can be modeled. This method has also been used in [88, 101, 102]. Using (4.30), (4.45) becomes:

$$\begin{aligned} y_n^{(m)} = \sum_{l,n_1,\ldots,n_m,k_1,\ldots,k_m} & H^{(m),l}_{nn_1\ldots n_m}(f_{k1},\ldots,f_{km}) \cdot \\ & x_{n1}(f_{k1})\ldots x_{nm}(f_{km}) \cdot \\ & e^{j2\pi[\sum_{i=1}^{m} f_{ki}+lf_0]t} \end{aligned} \tag{4.46}$$

Thus the output is the sum of all responses up to $n$-th order of every sideband transfer function. Each $n$-th order response is the sum of all responses of all combinations of order $n$ with all possible sidebands.

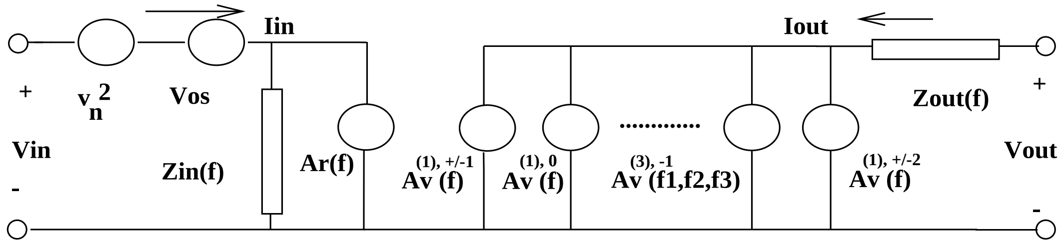



Figure 4.4: Mixer Behavioral Model

By using this representation, most RF building blocks can be modeled in the frequency domain. For example, the model for a mixer taking into account third order distortion, reverse leakage gain, offset, conversion gain, RF leakage and 2nd spurious response could be the following (at a specific frequency $f_j$):

$$
\begin{aligned}
0 &= -I_{in}(f_j) + A_{i_r}(f_j)I_{out}(f_j) + \frac{V_{in}(f_j)}{Z_{in}(f_j)} \\
V_{os}(f_j) &= -V_{out}(f_j) + Z_{out}(f_j)I_{out}(f_j) + \\
& \quad A_v{}^{(1),0}(f_j)V_{in}(f_j) + \\
\sum_{i:f_i \pm f_{LO}=f_j} & \quad A_v{}^{(1),\pm 1}(f_i)V_{in}(f_i) + \\
\sum_{i:f_i \pm 2f_{LO}=f_j} & \quad A_v{}^{(1),\pm 2}V_{in}(f_i) + \\
\sum_{i_1,i_2:f_{i_1}+f_{i_2} \pm f_{LO}=f_j} & \quad A_v{}^{(2),\pm 1}(f_{i_1}, f_{i_2})V_{in}(f_{i_1})V_{in}(f_{i_2}) + \\
\sum_{i_1,i_2,i_3:f_{i_1}+f_{i_2}+f_{i_3} \pm f_{LO}=f_j} & \quad A_v{}^{(3),\pm 1}(f_{i_1}, f_{i_2}, f_{i_3})V_{in}(f_{i_1})V_{in}(f_{i_2})V_{in}(f_{i_3}) \qquad (4.47)
\end{aligned}
$$

where $A_v{}^{(m),l}$ is the sideband $l$, order $m$, of the voltage gain. Figure 4.4 depicts the model. Instead of currents and voltages, also s-parameters can be used as in [100].

### 4.5.2 Considerations on Noise Modeling

The representation in 4.5.1 is compatible with the frequency domain *cyclo-stationary* noise analysis [13] presented in 4.4.3. If frequency domain noise spectral density functions are used to model the noise performance of the RF blocks, the sideband transfer functions defined in (4.29) can be used to compute the overall system noise performance. Thus we can have a unified representation of both deterministic and stochastic non-idealities.

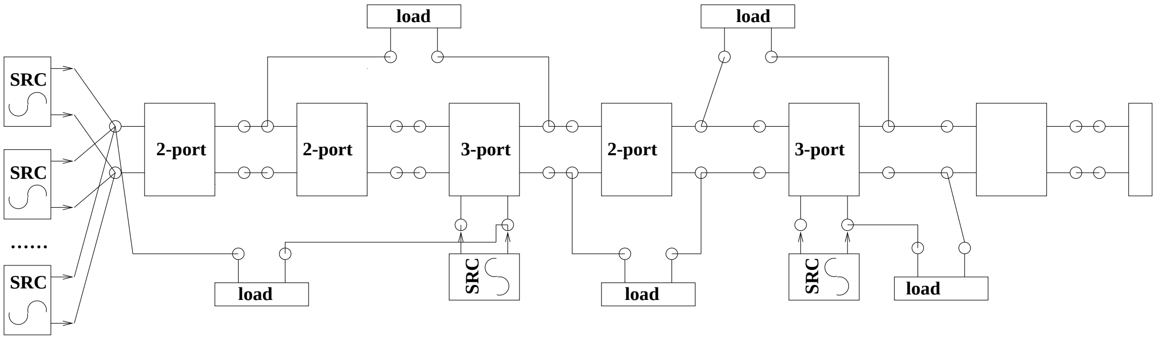



Figure 4.5: Sample Input System for VHSim

The noise performance of RF blocks is typically characterized by their average *Noise Figure* (NF). If the NF of a block is represented as a stationary white noise source, (4.35) and *harmonic balance* can be used to compute the output spectrum [13]. The same representation can be used also when the noise of the block is cyclo-stationary. However, the mixing of sideband spectra can effect the overall noise performance of a system only in cases where the local oscillators (LOs) of the system have a simple integer relationship, $n_1 Lo_1 = n_1 LO_2$, where $n_1, n_2$ are small integers. Furthermore, in [16] it is shown that the presence of phase noise in an oscillator signal may refute the cyclo-stationarity assumption. For those reasons, this work focuses on deterministic effects.

## 4.6 VHSim: Harmonic Balance High-Level Simulation Using Volterra Series-Based Models

The models discussed in 4.5.1 were used to implement a frequency domain behavioral simulator, VHSim.

It takes as inputs an arbitrary interconnection of RF blocks, as in Figure 4.5, with performance parameters in the frequency domain and real harmonic sources, and finds the frequency spectrum of currents and voltages at all nodes. A Spice-like input file format is used. A sample input file is shown in Figure 4.6. Performance parameters can be either obtained from the specifications of the block in the frequency domain or extracted from transistor-level simulators.

According to 4.5, two types of models have been implemented:

1. Generic $n$-ports with arbitrary $H^{(m),l}_{nn_1 \ldots n_m}(f_{k1}, \ldots, f_{km})$ transfer function as in (4.46).

```
source1 source 1 111 f 915.003e6 mag -87
source2 source 111 1111 f 916.002e6 mag -47
source3 source 1111 0 f 917.002e6 mag -47
antenna amp 1 0 2 0 av 0 nf 0 ip3 20
filterRF amp 2 0 3 0 av RFfilter.inf gain -3
lna amp 3 0 4 0 av lna.inf ip3 -6 av3 av3.inf nf 3 gain 20
mixerRF mixer 4 0 5 0 av 6 nf 15 ip3 10 lo 905e6
filterIF amp 5 0 6 0 av filterIF.inf nf 4 ip3 20 gain -5
ampIF amp 6 0 7 0 av 20 ip3 -10 nf 20
mixerIF mixer 7 0 8 0 av 0 lo 10e6 ip3 -10 nf 20
amp1 amp 8 0 9 0 av aa.inf ip3 -10 nf 30 gain 40
load imp 9 0 mag 50
options fmax 1.111e9 abstol 1e-12 reltol 1e-12
output 9 8 7 6 5 4 3 2 1
```

Figure 4.6: Simulator Input File

Those models give the user the highest flexibility to characterize a block in an arbitrary way but are harder to use.

2. Specific subclasses of $n$-ports, such as loads, sources, amplifiers, filters and mixers, so that systems can be easily instantiated. A typical mixer model is shown in Figure 4.4.

Appendix A describes the input language of VHSIM in more detail.

### 4.6.1 Mapping of Performance Parameters and Simulation Results to Volterra Transfer Functions

For linear parameters, the results of AC circuit simulations can be included. The coefficients of linear time-varying components (4.29) such as *conversion gain* for a mixer, can be extracted by simulators using periodical steady-state analysis as in [8]. Linear interpolation is used to find the value of the parameter at the desired frequency.

Second and third order nonlinearities are described by the magnitude and phase of the appropriate high-order Volterra transfer functions at different input frequency mixes. For example, the third order nonlinearity of an amplifier can be obtained by simulating the intermodulation performance at different input frequencies. Defining all $H^{(3)}(f_1, f_2, f_3)$ to be equal for all perturbations of $f_1, f_2, f_3$ according to (4.26), third order intermodulation at $2f_1 - f_2$, can be trivially mapped onto the value of the third-order Volterra coefficient at $H^{(3)}(f_1, f_1, -f_2)$.

Higher order nonlinearities are specified in two ways:

- the $IP_3$, $IP_2$ or the $P_{-1dB}$ are given for a block and are mapped onto the value of a *constant* Volterra transfer function, which is essentially the third order coefficient of a polynomial nonlinearity

- the second or third order gain is specified at different frequency mixes and a look-up table is formed for the third order Volterra transfer function

To find the value of a kernel at an arbitrary frequency mix, we use the sum of the input frequencies as a variable to do linear interpolation at the Volterra input-output map look-up table. If many values are found that correspond to different input frequencies with the same sum, $f_1$ is used to perform interpolation among the selected values and so on. The interpolation is based in the following ordering:

$$
\begin{aligned}
&\forall(f_1, f_2, \ldots, f_n) : f_1 \leq f_2 \leq \ldots \leq f_n, \\
&(f_{1_a}, f_{2_a}, \ldots, f_{n_a}) < (f_{1_b}, f_{2_b}, \ldots, f_{n_b}) \equiv \quad && (f_{1_a} + f_{2_a} + \ldots + f_{n_a} < f_{1_b} + f_{2_b} + \ldots + f_{n_b}) \\
& && \vee \quad ((f_{1_a} + f_{2_a} + \ldots + f_{n_a} = f_{1_b} + f_{2_b} + \ldots + f_{n_b}) \\
& && \quad \wedge (f_{1_a} < f_{1_b})) \\
& && \vee \quad \ldots \\
& && \vee \quad ((f_{1_a} + f_{2_a} + \ldots + f_{n_a} = f_{1_b} + f_{2_b} + \ldots + f_{n_b}) \wedge \\
& && \quad (f_{1_a} = f_{1_b}) \wedge \ldots \wedge (f_{n-2_a} = f_{n-2_b}) \\
& && \quad \wedge (f_{n-1_a} < f_{n-1_b}))
\end{aligned} \tag{4.48}
$$

The user has the flexibility to define all pairs, triplettes, so that second and third order distortion have the appropriate value in all desired frequency combinations.

Frequency-domain specifications or simulation results for specific parameters are included in separate files. Figure 4.7 shows an example of a third order distortion specification file while Figure 4.8 shows an example of a filter specification file.

### 4.6.2 Computation of Output Harmonics and Intermodulation Combinations

The simulation algorithm used is *harmonic balance*. If $f_i$ are the frequencies of the $K$ input *harmonic* sources and $f_{LO_i}$ the frequencies of the $L$ periodical strong nonlinearities, the output spectrum contains all possible $f_j$ such as:

$$
\begin{aligned}
f_j &= \sum_{i=1}^{K} m_i f_i + \sum_{i=1}^{L} l_i f_{LO_i}, \\
m_i, l_i &: \sum_{i=1}^{K} |m_i| \leq M_{max}, \sum_{i=1}^{L} |l_i| \leq L_{max}, f_j \leq f_{max}
\end{aligned} \tag{4.49}
$$

where $M_{max}$ is the maximum order nonlinear product, $L_{max}$ the maximum side-band sum and $f_{max}$ the maximum important frequency assumed present at the system. Only non-negative frequencies $f_j$ are considered since the system is real, causal with real excitations so for the solution $X(f)$, it is assumed that $X(f) = X^*(-f)$ if $f < 0$.

To further reduce the theoretically infinite number of output harmonics for the harmonic balance simulation, various heuristics were used based on the topology of the

```
f1 f2 f3 dB rad
-900e6 900e6 0e6 -60 0
-900e6 900e6 800e6 -60 0
-900e6 900e6 900e6 -60 0
-900e6 1000e6 0e6 -60 0
-900e6 1000e6 1000e6 -60 0
-900e6 1100e6 0 -60 0
-900e6 1200e6 900e6 -60 0
-1100e6 1100e6 1100e6 -50 0
-1100e6 900e6 900e6 -50 0
-1100e6 1000e6 1100e6 -50 0
-1100e6 1000e6 1000e6 -50 0
-1100e6 900e6 1100e6 -50 0
0 0 0 -100 0
900e6 900e6 900e6 -110 0
900e6 900e6 0 -110 0
-2400e6 2400e6 900e6 -200 0
-2400e6 900e6 900e6 -200 0
```

Figure 4.7: Third Order Specification Distortion File

```
902e6 -3 0
900e6 -6 0
928e6 -3 0
930e6 -6 0
950e6 -15 0
880e-6 -15 0
800e6 -30 0
1000e6 -30 0
```

Figure 4.8: Filter Specification File

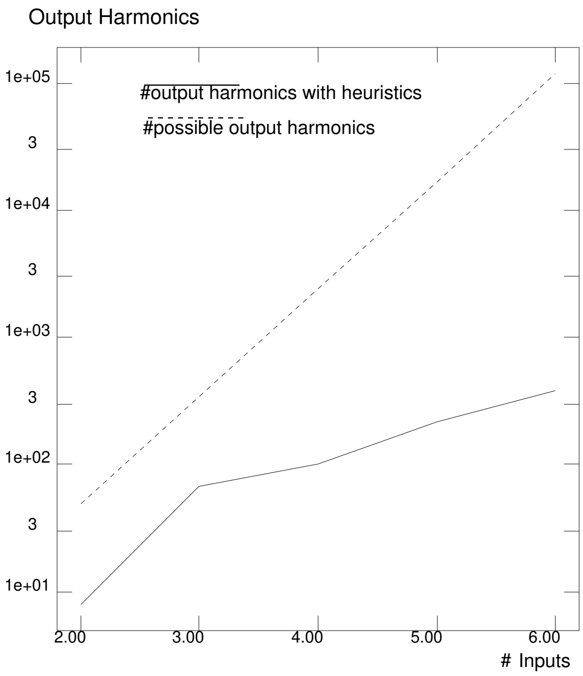



Figure 4.9: Output vs Input Harmonics

system. For a receiver system with multiple local oscillators for example, the distortion terms of *frequency converted* versions of $F_{in}$ such as $F_{in} - F_{LO_2}$ where $F_{LO_2}$ is the frequency of the second local oscillator, are eliminated. This is because in the receiver chain, the strong signals are $F_{in}, F_{in} - F_{LO_1}, F_{in} - F_{LO_1} - F_{LO_2}$. A term like $F_{in} - F_{LO_2}$ can only be a leakage term so it cannot excite important nonlinear products. Currently, up to third-order nonlinear models for blocks have been considered, but the method is expandable to higher orders at the expense of complexity.

Figure 4.9 shows the number of output harmonics as a function of the number of input harmonics with and without imposing the heuristics discussed above. The results are for one or two local oscillators and for 1-4 input sources imposing the constraints discussed above. The "unconstrained" case refers to the total number of output combinations if:

$$f_j = \sum_{i=1}^{K} m_i f_i + \sum_{i=1}^{L} l_i f_{LO_i},$$
$$|l_i| \leq L_{max}, |m_i| \leq L_{max}, \tag{4.50}$$

It is clear that a great reduction of the complexity is achieved.

Once the output harmonics have been defined, the intermodulation products need to be determined, which means that all possible second and third order combinations need to be created. Since the intermodulation of weak non-linearity products gives negligible products, only the intermodulation combinations of both non-negative and negative input harmonics, or frequency converted versions of the input harmonics, are considered. Those harmonics are described by:

$$\mathcal{F} : f_a \in \mathcal{F} \iff f_a = f_i + \sum_{i=1}^{L} l_i f_{LO_i}, \;\; |l_i| \leq 1 \tag{4.51}$$

where $f_i$ is any input harmonic, and $f_{LO_i}$ are the local oscillators' frequencies. To find the $n$-th order intermodulation products, the following procedure is followed:

1. find all $f_n : f_n = f_{a1} + \ldots + f_{an} + l_i f_{LO_i}$ where $f_{a1}, \ldots, f_{an} \in \mathcal{F}$, $|l_i| \leq 1$ ;
2. if $f_n \in \mathcal{F}_{out}$, where $\mathcal{F}_{out}$ is defined by (4.49) add the combination to the list $n$-th order combinations for $LO_i$ of output frequency $f_n$.

Figure 4.10 shows the maximum number of pairs and triplettes that correspond to an output frequency, as a function of the number of input frequencies. It is assumed that $L_{max} = M_{max} = 3$.

### 4.6.3 Assembly of High-Order Sparse Matrix Equations

The simulator assembles the sparse-matrix equations for all possible frequencies $f_j$ determined in (4.49):

$$\begin{aligned}
AI(f_j) &= 0 && (4.52)\\
V(f_j) - A^T E(f_j) &= 0 && (4.53)\\
\sum_{f_i + q_l f_{LO_l} = f_j} K_I^{(1) q_l f_{LO_l}}(f_i) I(f_i) + & && \\
\sum_{f_i + q_l f_{LO_l} = f_j} K_V^{(1) q_l f_{LO_l}}(f_i) V(f_i) + & && \\
\sum_{f_{i_1} + f_{i_2} + q_l f_{LO_l} = f_j} K_V^{(2) q_l f_{LO_l}}(f_{i_1}, f_{i_2}) \cdot & && \\
V(f_{i_1}) \otimes V(f_{i_2}) + & && \\
\sum_{f_{i_1} + f_{i_2} + q_l f_{LO_l} = f_j} K_I^{(2) q_l f_{LO_l}}(f_{i_1}, f_{i_2}) \cdot & &&
\end{aligned}$$

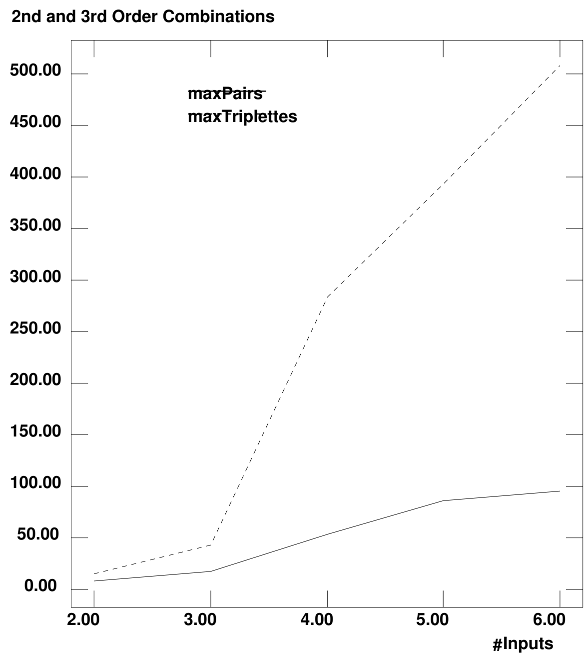



Figure 4.10: Triplettes, Pairs vs Output Harmonics

$$
\begin{aligned}
& I(f_{i_1}) \otimes I(f_{i_2}) + \\
\sum_{f_{i_1}+f_{i_2}+f_{i_3}+q_l f_{LO_l}=f_j} & K_V^{(3) q_l f_{LO_l}}(f_{i_1}, f_{i_2}, f_{i_3}) \cdot \\
& V(f_{i_1}) \otimes V(f_{i_2}) \otimes V(f_{i_3}) + \\
\sum_{f_{i_1}+f_{i_2}+f_{i_3}+q_l f_{LO_l}=f_j} & K_I^{(3) q_l f_{LO_l}}(f_{i_1}, f_{i_2}, f_{i_3}) \cdot \\
& I(f_{i_1}) \otimes I(f_{i_2}) \otimes I(f_{i_3}) \quad = \quad S(f_j) \qquad (4.54)
\end{aligned}
$$

where $V, I, E$ are the vectors of branch currents, branch voltages and node voltages respectively. $K_{V,I}^{(i) q_l f_{LO_l}}$ are assembled from the block-level models as in (4.47), $A$ is the connectivity matrix and "$\otimes$" is the Kronecker product of vectors. In the higher order products, $f_i$ are determined by (4.51). Equations (4.52), (4.53) are Kirchoff's current and voltage laws respectively (KCL, KVL) and (4.54) is the generalized nonlinear branch equation in the frequency domain. If $J$ is the number of possible non-negative output frequencies determined in (4.49), $N$ the number of nodes and $B$ the number of branches, (4.52)-(4.54) is a nonlinear, complex system of $J \cdot (N + 2B)$ equations in the frequency domain. In matrix notation the system in (4.52)-(4.54) can be re-written as:

$$
\begin{bmatrix} 0 \\ \vdots \\ 0 \\ 0 \\ \vdots \\ 0 \\ S(f_1) \\ \vdots \\ S(f_J) \end{bmatrix} =
$$

$$
\begin{bmatrix}
\begin{bmatrix} A & 0 & \cdot & 0 \\ 0 & A & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & A \end{bmatrix} & 0 & 0 \\
0 & \begin{bmatrix} I & 0 & \cdot & 0 \\ 0 & I & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & I \end{bmatrix} & \begin{bmatrix} -A^T & 0 & \cdot & 0 \\ 0 & -A^T & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & 0 & -A^T \end{bmatrix} \\
\begin{bmatrix} K_{I_1}^{f_1-f_1}(f_1) & \cdot & K_{I_1}^{f_1-f_J}(f_J) \\ \cdot & \cdot & \cdot \\ K_{I_1}^{f_J-f_1}(f_1) & \cdot & K_{I_1}^{f_J-f_J}(f_J) \end{bmatrix} & \begin{bmatrix} K_{V_1}^{f_1-f_1}(f_1) & \cdot & K_{V_1}^{f_1-f_J}(f_J) \\ \cdot & \cdot & \cdot \\ K_{V_1}^{f_J-f_1}(f_1) & \cdot & K_{V_1}^{f_J-f_J}(f_J) \end{bmatrix} & 0
\end{bmatrix} \cdot
\begin{bmatrix} I(f_1) \\ \vdots \\ I(f_J) \\ V(f_1) \\ \vdots \\ V(f_J) \\ E(f_1) \\ \vdots \\ E(f_J) \end{bmatrix} +
$$

$$
\begin{bmatrix} K_{I_1}^{f_1+f_1}(-f_1) & \cdot & K_{I_1}^{f_1+f_J}(-f_J) \\ \cdot & \cdot & \cdot \\ K_{I_1}^{f_J+f_1}(-f_1) & \cdot & K_{I_1}^{f_J+f_J}(-f_J) \end{bmatrix} \cdot \begin{bmatrix} I(-f_1) \\ \vdots \\ I(-f_J) \end{bmatrix} + \begin{bmatrix} K_{V_1}^{f_1+f_1}(-f_1) & \cdot & K_{V_1}^{f_1+f_J}(-f_J) \\ \cdot & \cdot & \cdot \\ K_{V_1}^{f_J+f_1}(-f_1) & \cdot & K_{V_1}^{f_J+f_J}(-f_J) \end{bmatrix} \cdot \begin{bmatrix} V(-f_1) \\ \vdots \\ V(-f_J) \end{bmatrix} +
$$

$$
\begin{bmatrix} K_{V_2}^{2f_J+f_1}(-f_J,-f_J) & \cdot & K_{V_2}^{-2f_J+f_1}(f_J,f_J) \\ \cdot & \cdot & \cdot \\ K_{V_2}^{2f_J+f_J}(-f_J,-f_J) & \cdot & K_{V_2}^{-2f_J+f_J}(f_J,f_J) \end{bmatrix} \cdot \left( \begin{bmatrix} V(-f_J) \\ \vdots \\ V(f_J) \end{bmatrix} \otimes \begin{bmatrix} V(-f_J) \\ \vdots \\ V(f_J) \end{bmatrix} \right) +
$$

$$
\begin{bmatrix} K_{V_3}^{3f_J+f_1}(-f_J,-f_J,-f_J) & \cdot & K_{V_3}^{-3f_J+f_1}(f_J,f_J,f_J) \\ \cdot & \cdot & \cdot \\ K_{V_3}^{3f_J+f_J}(-f_J,-f_J,-f_J) & \cdot & K_{V_3}^{-3f_J+f_J}(f_J,f_J,f_J) \end{bmatrix} \cdot \left( \begin{bmatrix} V(-f_J) \\ \vdots \\ V(f_J) \end{bmatrix} \otimes \begin{bmatrix} V(-f_J) \\ \vdots \\ V(f_J) \end{bmatrix} \otimes \begin{bmatrix} V(-f_J) \\ \vdots \\ V(f_J) \end{bmatrix} \right) \quad (4.55)
$$

where $K_{V_n}^{f_i}(f_1, \ldots, f_n)$ is the sideband at frequency $f_i$, if $f_i = lf_{LO}$ and $lf_{LO}$ define an existing sideband transfer function of a module, otherwise it is zero and $n$ is the order of the nonlinearity. All matrices are very sparse.

### 4.6.4 Solution Algorithm

Defining:

$$
X(f)^T = [I(f_1) \ldots I(f_J) V(f_1) \ldots V(f_J) E(f_1) \ldots E(f_J)] \quad (4.56)
$$

(4.55) can be re-written as:

$$
\begin{bmatrix} S(f) \\ S(-f) \end{bmatrix} = \mathrm{Q}_1 \cdot \begin{bmatrix} X(-f) \\ X(f) \end{bmatrix} + \mathrm{Q}_2 \cdot \left( \begin{bmatrix} X(-f) \\ X(f) \end{bmatrix} \otimes \begin{bmatrix} X(-f) \\ X(f) \end{bmatrix} \right) + \mathrm{Q}_3 \cdot \left( \begin{bmatrix} X(-f) \\ X(f) \end{bmatrix} \otimes \begin{bmatrix} X(-f) \\ X(f) \end{bmatrix} \otimes \begin{bmatrix} X(-f) \\ X(f) \end{bmatrix} \right) \quad (4.57)
$$

or

$$F(X(f), X(-f)) = 0 \tag{4.58}$$

Since for real system with real excitations $X(-f) = X^*(f)$, (4.58) becomes:

$$F(X(f), X^*(f)) = 0 \tag{4.59}$$

If we define:

$$\begin{aligned} X &= X_R + jX_I \\ F &= F_R + jF_I \\ \overline{X}^T &= \left[ {X_R}^T {X_I}^T \right] \\ \overline{F}^T &= \left[ {F_R}^T {F_I}^T \right] \end{aligned}$$

(4.59) can be converted into a real system of equations that can be solved with the Newton-Raphson iteration:

$$\overline{X}^{n+1} = \overline{X}^n - \frac{\partial \overline{F}(\overline{X}^n)}{\partial \overline{X}^n}^{-1} \cdot \overline{F}(\overline{X}^n) \tag{4.60}$$

where

$$\frac{\partial \overline{F}(\overline{X})}{\partial \overline{X}} = \begin{bmatrix} \frac{\partial F_R}{\partial X_R} & \frac{\partial F_R}{\partial X_I} \\ \frac{\partial F_I}{\partial X_R} & \frac{\partial F_I}{\partial X_I} \end{bmatrix} \tag{4.61}$$

It can be easily proven that:

$$\frac{\partial F_R}{\partial X_R} = Re\left[\frac{\partial F}{\partial X} + \frac{\partial F}{\partial X^*}\right] \tag{4.62}$$

$$\frac{\partial F_R}{\partial X_I} = Im\left[\frac{\partial F}{\partial X} + \frac{\partial F}{\partial X^*}\right] \tag{4.63}$$

$$\frac{\partial F_I}{\partial X_R} = -Im\left[\frac{\partial F}{\partial X} - \frac{\partial F}{\partial X^*}\right] \tag{4.64}$$

$$\frac{\partial F_I}{\partial X_I} = Re\left[\frac{\partial F}{\partial X} - \frac{\partial F}{\partial X^*}\right] \tag{4.65}$$

where $\frac{\partial F}{\partial X^*}$ denotes the Jacobian of $F$ with respect to the conjugate of vector $X$. At every step of the iteration, the linear system is very sparse and can be solved efficiently using sparse matrix techniques [103]. The constant portion of the Jacobian coming from the linear term of (4.57) is evaluated only once and then is updated at every iteration adding the terms from the nonlinear portion of(4.57). The algorithm for the iterative solution of (4.61) is shown in Figure 4.11. A typical flow of the overall behavioral simulation methodology is shown in Figure 4.12.

evaluate constant term of $\frac{\partial F}{\partial X}$; /* from (4.57) $Q_1$ matrix */
evaluate constant term of $\frac{\partial F}{\partial X^*}$; /* from (4.57) $Q_1$ matrix */
set X = 0; n=0
**repeat**
    n= n+1;
    evaluate higher-order terms of $\frac{\partial F}{\partial X}$; /* from (4.57) $Q_2, Q_3$ matrices */
    evaluate higher-order terms of $\frac{\partial F}{\partial X^*}$; /* from (4.57) $Q_2, Q_3$ matrices */
    evaluate $\overline{F}(\overline{X}^n)$;
    find $\overline{X}^{n+1}$;
**until** $|\overline{X}^{n+1} - \overline{X}^n| \leq \delta_{TOLERANCE}$

Figure 4.11: Iterative Solution Algorithm

### 4.6.5 Models Implemented - Limitations of Current Implementation

As shown in 4.4.1, Volterra series can be used to model any practical nonlinear function, if enough terms of the Volterra expansion (4.18) are used. To reduce computational complexity, only up to third-order Volterra transfer functions were considered in the models used. Spurious responses for mixers, which can be modeled with the linear term of higher order sideband transfer functions ${A_v}^{(1),\pm n}$, were considered up to third order [8]. The full models for both mixers and amplifiers are described in Appendix A.

Due to the models implemented, the current version of VHSIM has limitations. The case of a strong input signal to an amplifier or a mixer is not modeled accurately. This may occur in receivers driven to compression by a strong blocker, or in transmitters using highly nonlinear power amplifiers. Since VHSIM models can be easily modified to include higher order Volterra transfer functions, including functions up to fifth or seventh order should be sufficient for most practical cases. The easiest way to determine those higher order functions is to use high-order polynomial fitting and consider the functions to be equal to the constant coefficients of the polynomial (frequency independent). A small modification of the algorithm determining the output frequency combinations is needed to include frequency combinations due to the higher orders. Furthermore, the equation

[8] Typically this is sufficient for most mixers

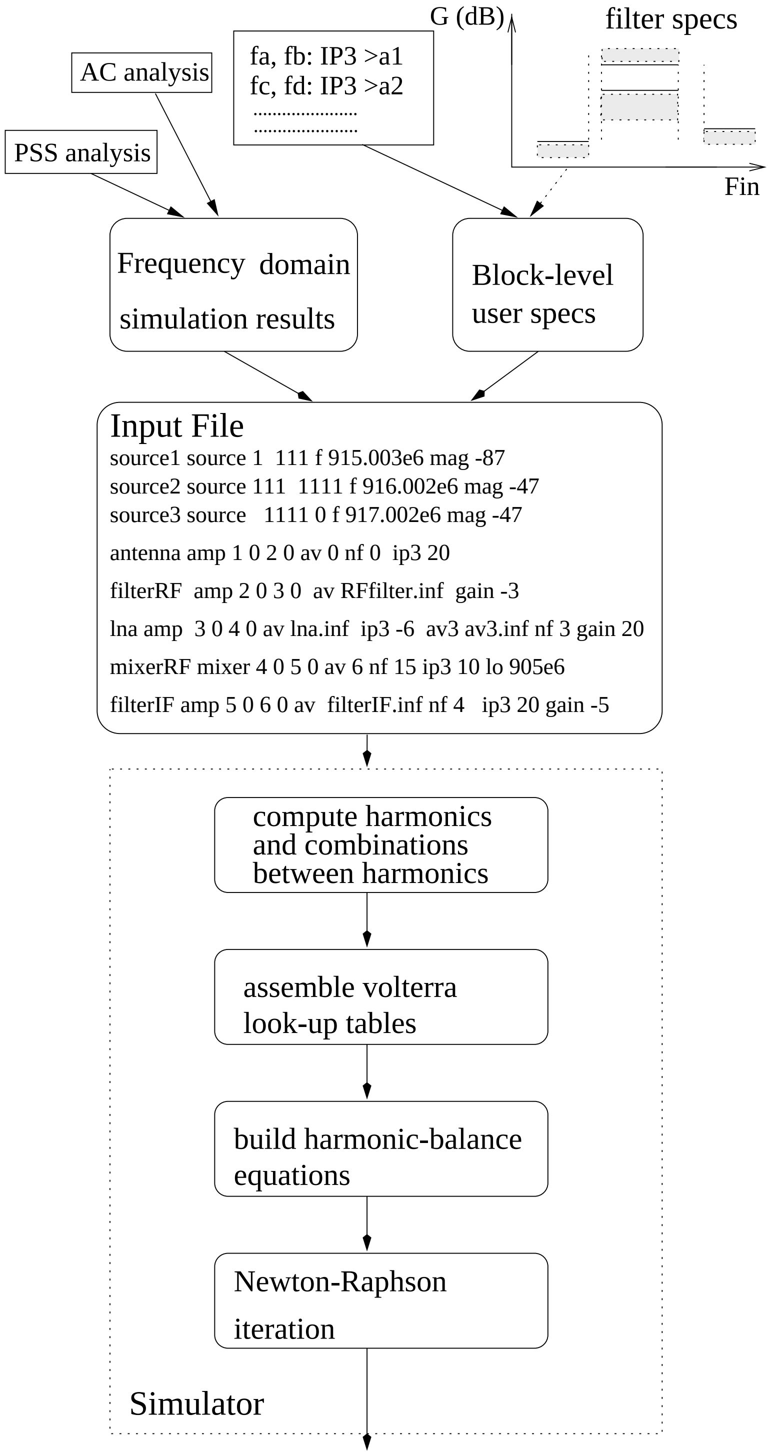



Figure 4.12: Behavioral Simulator Flow Diagram

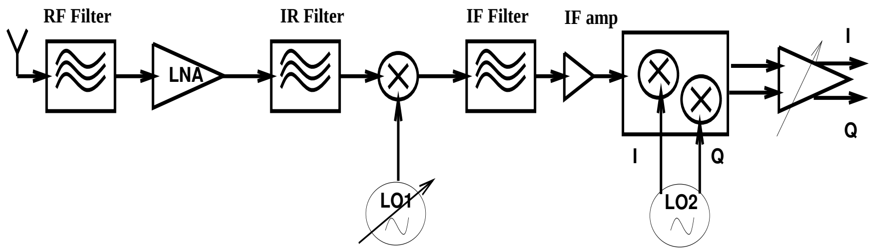



Figure 4.13: Super-Heterodyne Receiver

formulation mechanism described in 4.6.3 should include higher order terms. It should be noted that including higher order terms may significantly increase the simulation time, since the number of harmonics depends on the order of nonlinearity. $J \propto K^n$, where $J$ is the number of output harmonics, $K$ the number of input harmonics and $n$ the maximum order of nonlinearities. The use of heuristics may significantly reduce the number of output harmonics, as shown in 4.6.2.

## 4.7 Examples

### 4.7.1 High-Level Simulation of Super-Heterodyne Receiver

The super-heterodyne receiver of Figure 4.13 was used to test the behavioral simulator VHSIM. It is intended to operate in the $902 - 928MHz$ band and selects a $30KHz$ channel. It has a sensitivity of $-90dBm$, intermodulation protection of $30dB$ and in-band blocking of -37dB meeting a $10^{-3}$ BER (10dB SNDR) specification. The results were compared to behavioral models in SPECTREHDL, the high-level language used with SPECTRERF [9].

In both simulators performance metrics such as filtering, second and third order intercept points ($IP_2$,$IP_3$), DC offsets, RF leakages, coupling and frequency translation were modeled. In SPECTREHDL, nonlinearities were modeled with polynomials and filters were mapped to 3-10th order Chebyscheff or Butterworth implementations. A typical block in SPECTREHDL was modeled as in Figure 4.4. Table 4.1 summarizes the types of models used. The approach of the Volterra I/O mapping table gives a unified representation for all blocks in the case of VHSIM.

Two types of analyses were used in SPECTRERF:

| | VHSIM | SPECTRERF |
|---|---|---|
| filters | Volterra I/O map | $\frac{N(s)}{D(s)}$ |
| $IP_2, IP_3, P^{-1dB}$ | Volterra I/O map | third order polynomials |
| mixing | Volterra I/O map | ideal multiplier |
| offsets,sources | multi-tone harmonic sources | multi-tone harmonic sources |

Table 4.1: Modeling of RF Blocks

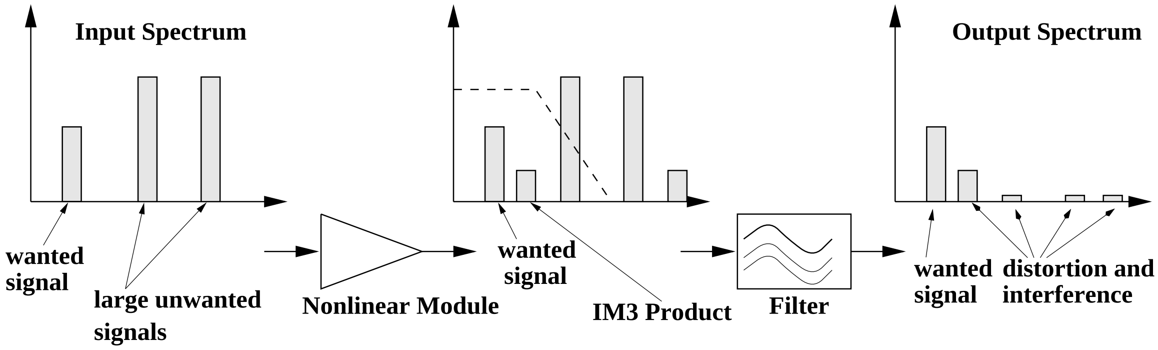



Figure 4.14: Third Order Intermodulation

- PSS, where all the harmonics are considered strong nonlinear excitations and therefore the minimum common period is used to find the periodically varying operating point; it is more accurate but takes a lot longer to complete, since for a narrow-band system the tones are closely spaced.

- PAC, where only the LOs are considered strong nonlinearities and input signals are considered linear; this analysis does not give accurate results for distortion but is a lot faster. For distortion analysis, the input can be consider as one large signal plus one small signal and PAC can be used to find the intermodulation.

In a narrow-band system like a super-heterodyne receiver, the filters used are in the order of 30 KHz bandwidth. To accurately simulate the effects of the filters in the presence of distortion, as in Figure 4.14, a small fundamental frequency is required. This is also desirable in order to accurately model the intermodulation of narrow-band adjacent channels. Using a small fundamental frequency, as explained in 4.4.5, may significantly degrade the performance of the periodic steady-state analysis in SPECTRERF.

The results of VHSIM were found to be consistent with the ones from SPECTRERF with a big improvement in simulation time, especially when a small fundamental is used.

| | | VHSIM | SPECTRERF | |
|---|---|---|---|---|
| | | | *PSS* | *PAC* |
| 2-tones | CPU (sec) | 7 | 2265 (fund = 10KHz) | 30.6 (fund = 1MHz) |
| 2-LOs | max error (dB) | 0.1 | 0 | 1.5 |
| 3-tones | CPU (sec) | 32.8 | 2265 (fund = 10KHz) | 30.6 (fund = 1MHz) 30.6 |
| 2-LOs | max error(dB) | 0.12 | 0 | 1.6 |

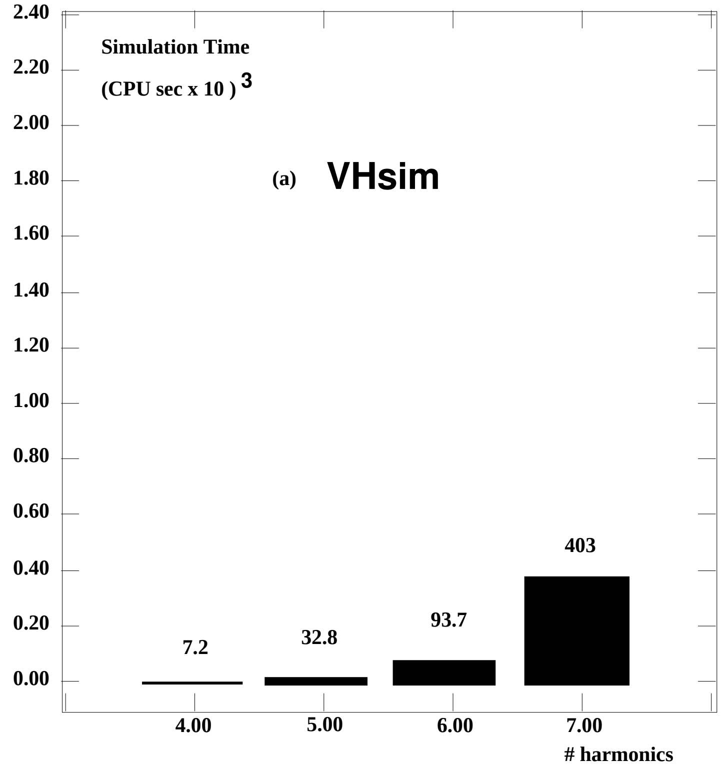
Table 4.2: Super-heterodyne Receiver Simulation Comparison

A three-tone test was used to check the signal-to-interference ratio in the presence of two intermodulating signals. The frequencies used were $LO_1 = 905MHz, LO_2 = 10MHz, F_1 = 915.01MHz, F_2 = 916.02MHz, F_3 = 917.02MHz$.

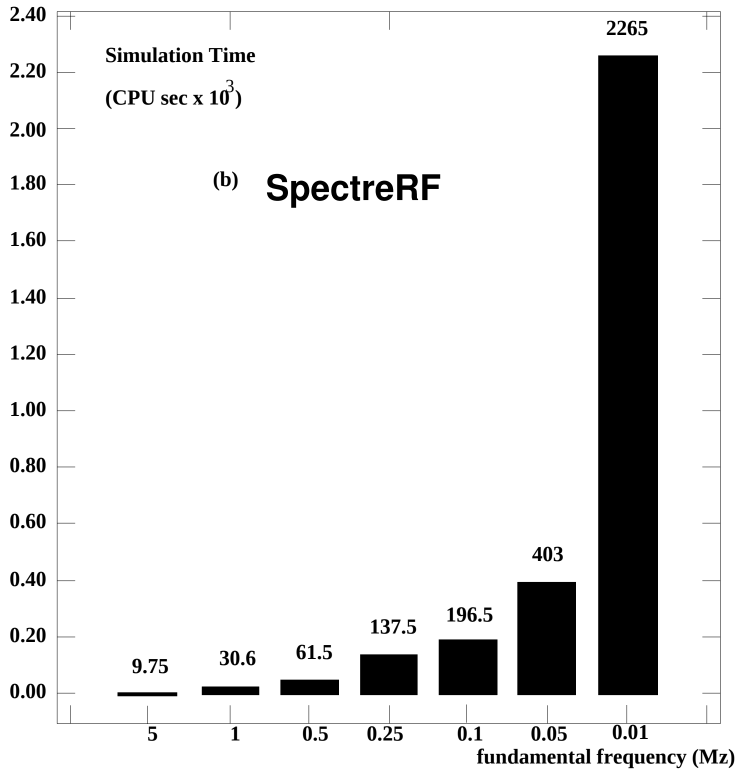
The performance of the simulators was evaluated for different numbers of input harmonics and different input frequency spacing. As expected, the simulation time for VHSIM depends only on the number of harmonics present in the system, while for SPECTRERF it depends on the input frequency spacing. Figure 4.15 shows the simulation times for the same problem for both simulators, as a function of the fundamental frequency and the number of input harmonics. It should be noted however that in SPECTRERF, high-order filters create convergence problems, thus lower order filters had to be used. This is not a problem for VHSIM. For simulations with many input harmonics VHSIM becomes, as expected, significantly slower.

The results are summarized in Table 4.2. VHSIM simulations were run on a DEC Alpha-Server 600MHz and all CPU times are normalized to that. For SPECTRERF simulations, an HP C240 was used. The maximum error refers to the third order intermodulation component which dominates the SNDR. The result of a periodic steady-state analysis is used for reference. For PSS analysis, a very small fundamental frequency was used since the system is narrow-band. The speed advantage of VHSIM is reduced when compared to PAC, but the accuracy is better.

A typical VHSIM output is shown in Figure 4.16. Figure 4.17 shows the simulation time in VHSIM for different number of input tones, while changing $f_{max}$ and therefore the size of the problem (Equation 4.49). Table 4.3 summarizes the performance of the simulator for 4 and 5 input tones. $N_1, N_2$, are the number of input and LO frequencies, $F_{max}$ the maximum upper frequency, $J_{TOT}$ the total number of frequency combinations, $J$ the number of output frequencies used, $\#N_z$ the percentage of non-zero Jacobian elements and N the total number of branches and nodes. When more nonlinear effects are included, the matrices




(a)




(b)

Figure 4.15: (a) VHSIM Simulation Speed (b) SPECTRERF Simulation Speed

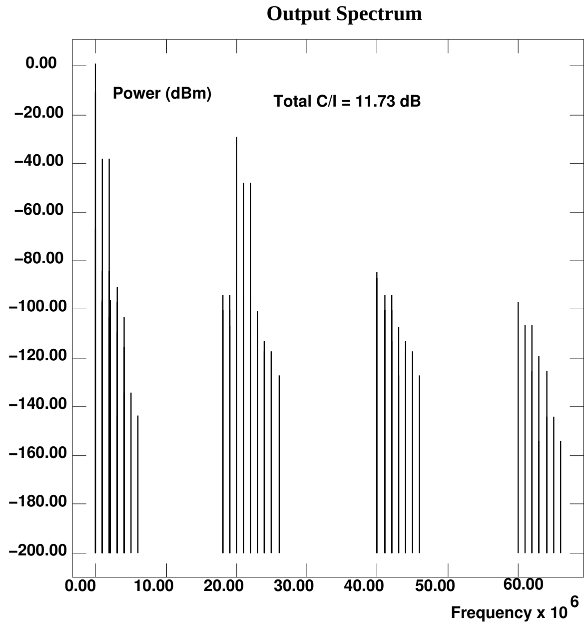



Figure 4.16: Simulator Output

| $N_1, N_2,$ | $F_{max}$ | $CPU$ (s) | $J_{TOT}$ | $J$ | $N$ | $\#N_z\%$ |
|---|---|---|---|---|---|---|
| 2,2 | 2.1e9 | 7.1 | 1241 | 168 | 8604 | 0.05 |
| 2,2 | 6e9 | 17 | 1241 | 253 | 12144 | 0.03 |
| 3,2 | 1.1e9 | 9.1 | 16807 | 217 | 11067 | 0.03 |
| 3,2 | 2.1e9 | 67 | 16807 | 428 | 19788 | 0.02 |

Table 4.3: VHSIM Performance Table

become less sparse and simulation time increases.

### 4.7.2 Wideband IF CMOS RF Receiver

The wideband IF receiver of Figure 4.18 [104] was also used to evaluate the performance of VHSIM. Briefly, the architecture works as follows:

1. The input signal is filtered by an RF filter to obtain the desired RF band.

2. After amplification from a low noise amplifier, the signal is down-converted to an IF, using a *fixed local* oscillator, as opposed to a channel-select variable oscillator that is used in a super-heterodyne receiver. As a result, the *image* RF band is also converted and an image-reject mixer architecture is used to cancel the image band. The second local oscillator is tuned to the desired channel and down-converts it to DC.

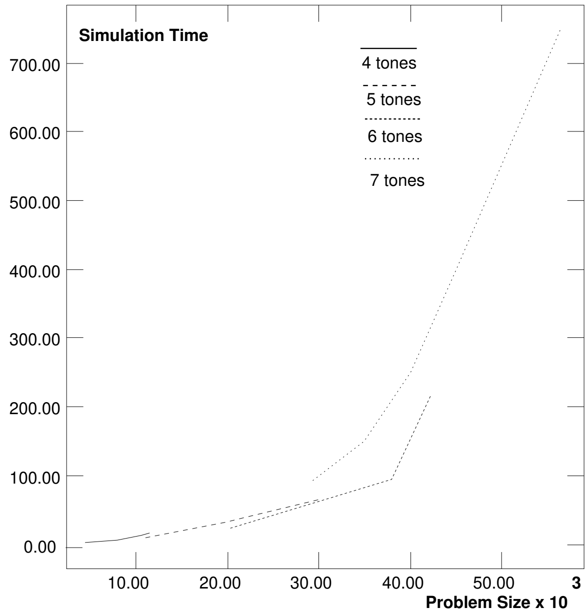



Figure 4.17: Simulation Speed

3. The baseband signal is amplified by a variable gain amplifier and finally filtered by a low-pass filter.

4. The filtered channel is digitized by an A/D converter. Additional filtering is being done digitally.

#### 4.7.2.1 Behavioral System-Level Simulation

The selection of the specifications for the individual components is typically done manually, using analytical calculations. VHSIM was used to perform intermodulation and blocking simulations using the parameters of Table 4.4 from [105]. Two or three input tones were used for the simulations. The frequencies and magnitudes of the tones used, are summarized in Table 4.5.

The results of the simulations are summarized in Table 4.6 and agree with the specifications for the overall receiver derived in [105] ($IP_3$ =-13.5 dBm). The output of the simulator for the two-tone, third order intermodulation test is shown in Figure 4.19. Figure 4.20 shows the output spectrum using three tones (one wanted signal and two unwanted large blockers). In both figures, the spectrum before and after the assumed digital filter is shown.

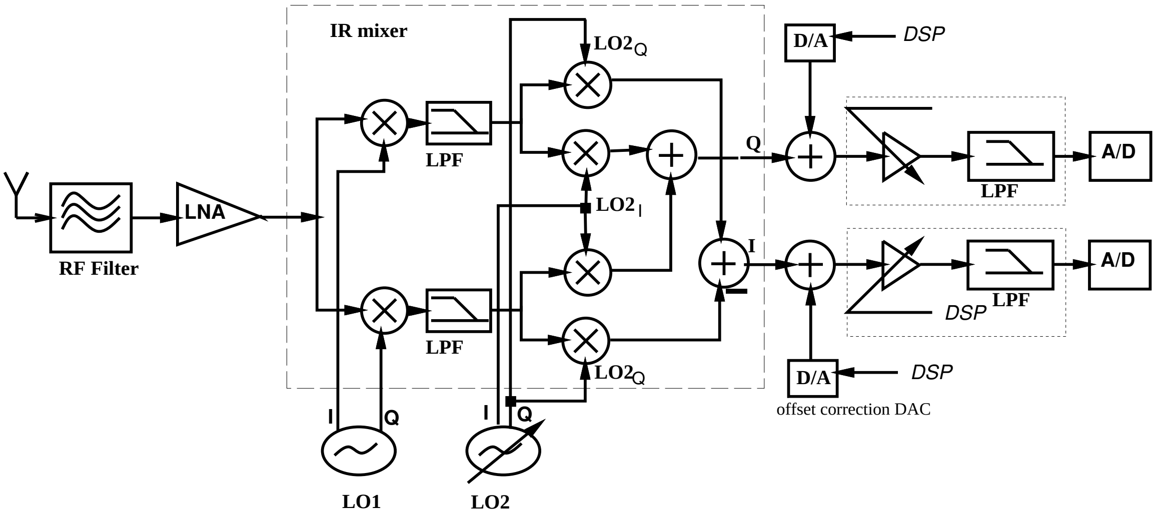
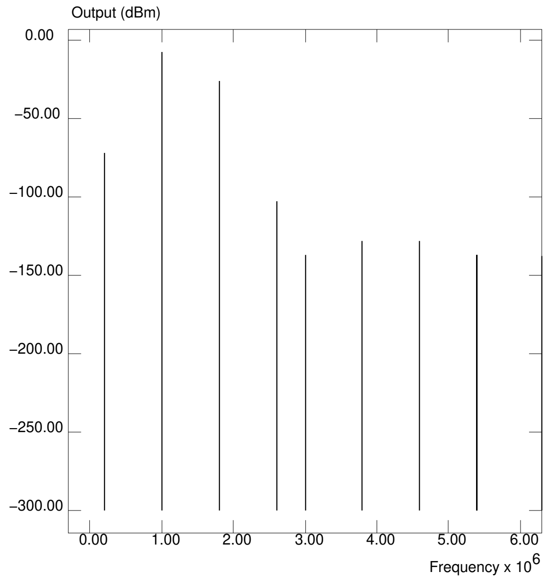
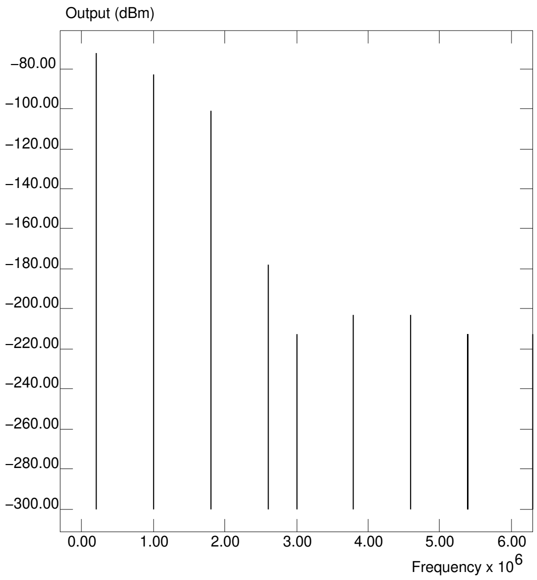



Figure 4.18: Wideband IF Receiver




(a) Without Digital Filter




(b) With Digital Filter

Figure 4.19: Simulator Output for Wideband IF Receiver - 2 tones

| *Block* | *Variable* | *Specification* |
|---|---|---|
| *Antenna* | Gain (dB) | 0 |
| | NF (dB) | 0 |
| *RF Filter/T-R Switch* | Gain | -4 |
| | NF (dB) | 0 |
| | Attenuation (dB) | 38 |
| *LNA* | max Gain(dB) | 22 |
| | min Gain (dB) | 0 |
| | NF (dBm) | 2.74 |
| | input $IP_3$ (dBm) | -7 |
| *Mixer 1* | max conv. Gain(dB) | 10 |
| | min conv. Gain(dB) | 0 |
| | NF (dB) | 18.6 |
| | input $IP_3$ (dBm) | 13 |
| *LO1* | phase noise (dBc/Hz) $_{3\ \mathrm{MHz}}$ | -137 |
| | channel spacing (KHZ) | fixed |
| | $F_{LO}$ (MHz) | 1470 |
| *Mixer 2* | conv. Gain(dB) | 8 |
| | NF (dB) | 26 |
| | input $IP_3$ (dBm) | 17.1 |
| *LO2* | phase noise (dBc/Hz) $_{3\ \mathrm{MHz}}$ | -137 |
| | channel spacing (KHZ) | 200 |
| | $F_{LO}$ (MHz) | 390 |
| *VGA Amp / Anti-Alias Filter* | max Gain(dB) | 12 |
| | min Gain (dB) | 3 |
| | NF | 32 |
| | input $IP_2$ (dBm) | 107 |
| | input $IP_3$ (dBm) | 29.9 |
| *Digital Channel Select Filter* | *frequency (Hz)* | *attenuation (dB)* |
| | 0-300e3 | 0 |
| | 400e3-600e3 | 15 |
| | 800e3- | 65 |

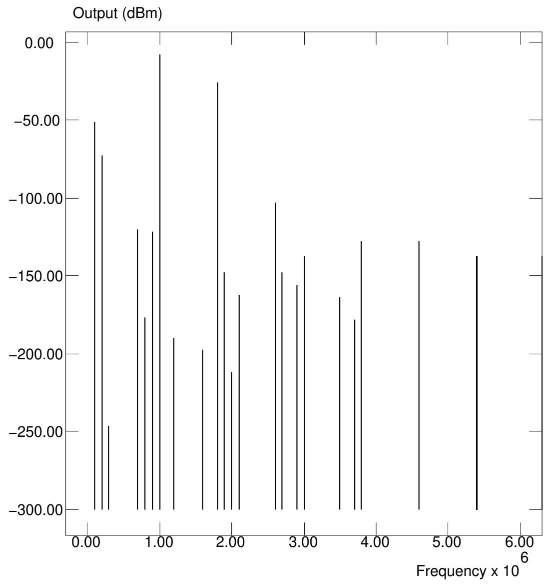
Table 4.4: Wideband IF Receiver Block-Level Parameters

| Test | *Frequency* (Hz) | Power (dBm) ) |
|---|---|---:|
| (2 tones) | 1861e6 | -49 |
| | 1861.8e6 | -49 |
| (3 tones) | 1861e6 | -49 |
| | 1861.8e6 | -49 |
| | 1860.1e6 | -99 |

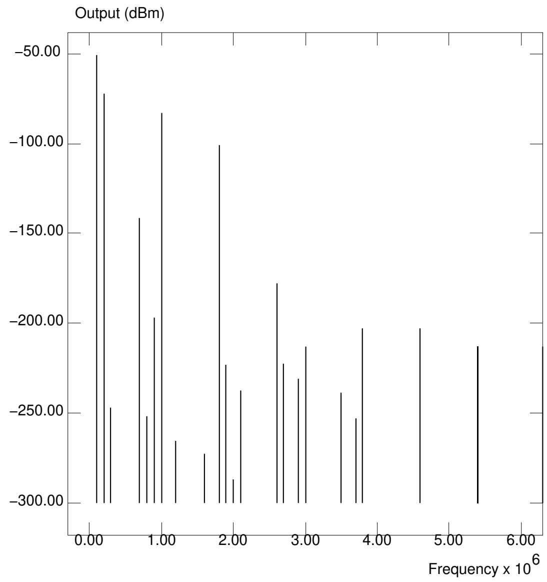
Table 4.5: VHSIM Wideband IF Simulation Inputs

| Test | *CPU Time* (s) | Problem Size | # harmonics | SNDR (dB) | $IP_{3TOTAL}$ |
|---|---|---|---|---|---|
| $IM_3$ (2 tones) | 4.6 | 5966 | 157 | - | -13dBm |
| $IM_3$ (3 tones) | 17.3 | 13612 | 332 | 21.3 | -13dBm |

Table 4.6: VHsim Wideband IF Simulation Results




(a) Without Digital Filter




(b) With Digital Filter

Figure 4.20: Simulator Output for Wideband IF Receiver - 3 tones

| Module | Parameter | *CPU Time* (s) |
|---|---|---|
| RF Mixer | $A_v{}^{\pm 1}(f)$, $A_v{}^{\pm 3}(f)$ | 31.9 |
| | $IM_3(f_{out})$ (4 points) | 4 x 151 |
| IF Mixer | $Av^{\pm 1}(f)$, $Av^{\pm 3}(f)$ | 22 |
| | $IM_3(f_{out})$ (4 points) | 4 x 51 |
| RC Filter | $A_v(f)$ | |
| VGA + Sallen-Key Filter | $A_v(f)$ | 5 |
| | $IM_3(f_{out})$ (4 points) | 4 x 74 |

Table 4.7: Wideband IF Receiver Extraction Results

#### 4.7.2.2 Bottom-Up Hierarchical Verification

The wideband-IF receiver described in 4.7.2 was used to test the performance of the VHsim in simulating an extracted transistor-level design. The designs extracted, were the ones used for the multi-standard, fully integrated CMOS RF transceiver [105], which is an evolution of [104]. Transistor-level extracted schematics were used to create Volterra look-up tables. SpectreRF PAC analysis was used to extract the linear sideband transfer functions, up to third order, of the two mixers (RF and IF). The intermodulation performance was extracted using periodic steady-state (PSS) analysis at different frequency mixes. The AC response and the intermodulation response of the VGA/baseband filter was also extracted using AC and PSS analysis with SpectreRF.

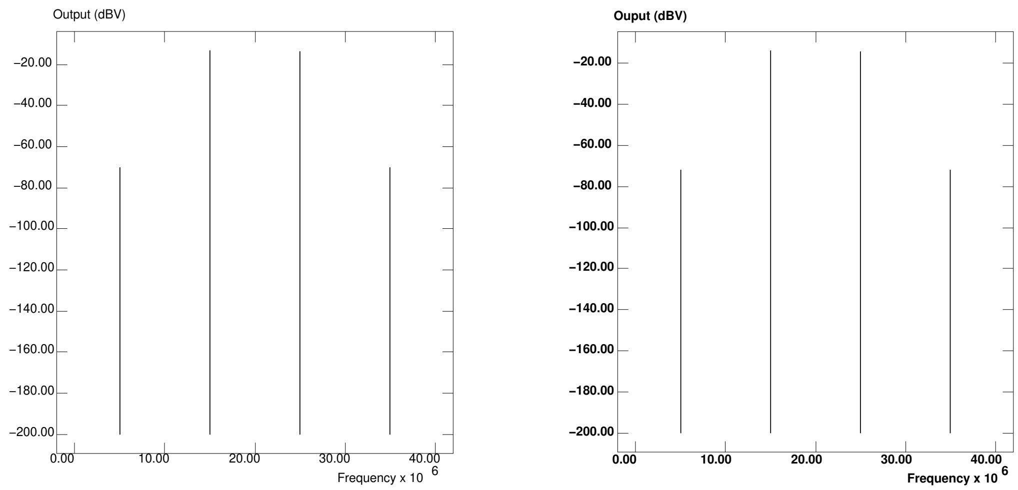
Table 4.7 summarizes the parameters that were extracted with the corresponding simulation times in SpectreRF.

**Cascade of Mixers**

To test the performance of the behavioral simulator, the cascade of the two mixers, RF and IF, were simulated with VHsim. The overall mixer structure was also simulated using SpectreRF. Table 4.8 summarizes the results. The total simulation time for VHsim, includes both the extraction with SpectreRF, plus the actual behavioral simulation time. A big improvement in overall simulation time is achieved, while the accuracy of the simulation is within 0.5 dB from full SpectreRF simulation. VHsim simulations were run on a DEC Alpha-Server 600MHz and all CPU times are normalized to that. Figure 4.21 shows the output spectra close to the intermodulation products, from both SpectreRF and VHsim. Simulations match accurately for the signals above the numerical noise floor.

| | *CPU Time* (s) | Overall input $IP_3$ |
|---|---|---|
| SPECTRERF (PSS) | 919 | -2.95 dBV |
| VHSIM | VHSIM: 3.6 | -3.5 dBV |
| | Extraction: 255 | |
| | Total: 258.6 | |

Table 4.8: VHSIM Two Mixer Cascade Simulation Comparison




(a) VHSIM (b) SPECTRERF

Figure 4.21: Cascade of Mixers $IM_3$ Simulation

The difference of 0.5 dB can be attributed to numerical errors of both simulators and the maximum frequency limit in VHSIM.

The tones used for the simulations were: $f_1 = 1875MHz$, $f_2 = 1885MHz$, $F_{LO_1} = 1470MHz$, $F_{LO_2} = 390MHz$. However, for VHSIM, using $f_1 = 1861.8MHz$, $f_2 = 1861MHz$, which are the input tones that are specified for the receiver to test the intermodulation performance, the overall input $IP_3$ is $V_{IP_3}$ = -3.9 dBV. At different frequency tones the $IP_3$ may be different. With SPECTRERF, this is very hard to verify for the cascade, since a very small fundamental frequency has to be used and the simulation time and required memory are prohibitive [9]. For the individual mixers, more accurate values can be extracted, using tones closer to the ones specified, and then VHSIM can be used for the system simulation.

[9] See section 4.4.5. For the Newton-Raphson iteration the whole transient waveform is saved, thus the memory requirements dramatically increase with the reduction of the step-size. High accuracy requires a very small step-size in the numerical integration.

The accuracy of the full SpectreRF simulation can also be compromised by the very small signals used, requiring very tight tolerances. If very small amplitude tones are used, the numerical noise becomes comparable to the intermodulation noise. For this reason, larger amplitudes are preferred. However, if a cascade of blocks is used, the amplitude of the signals exciting the subsequent stages may be too large, so that higher than third order nonlinearities cause the $IP_3$ calculation to be inaccurate. Thus, if the blocks are extracted separately using appropriate amplitudes, VHsim may give more accurate results. In Table 4.8, if the cascade formula for the calculation of intermodulation is used, $IP_3 = -3.45dBV$, which agrees with VHsim.

### Receiver Path

The overall intermodulation performance of the receiver was also tested using the extracted Volterra look-up tables with VHsim. For the RF Filter and the LNA that were not available, behavioral models were used from the specifications of Table 4.4. A comparison with a full SpectreRF simulation was not possible due to the complexity of the problem. The fundamental frequency for this simulation has to be very small requiring excessive computation time and memory.

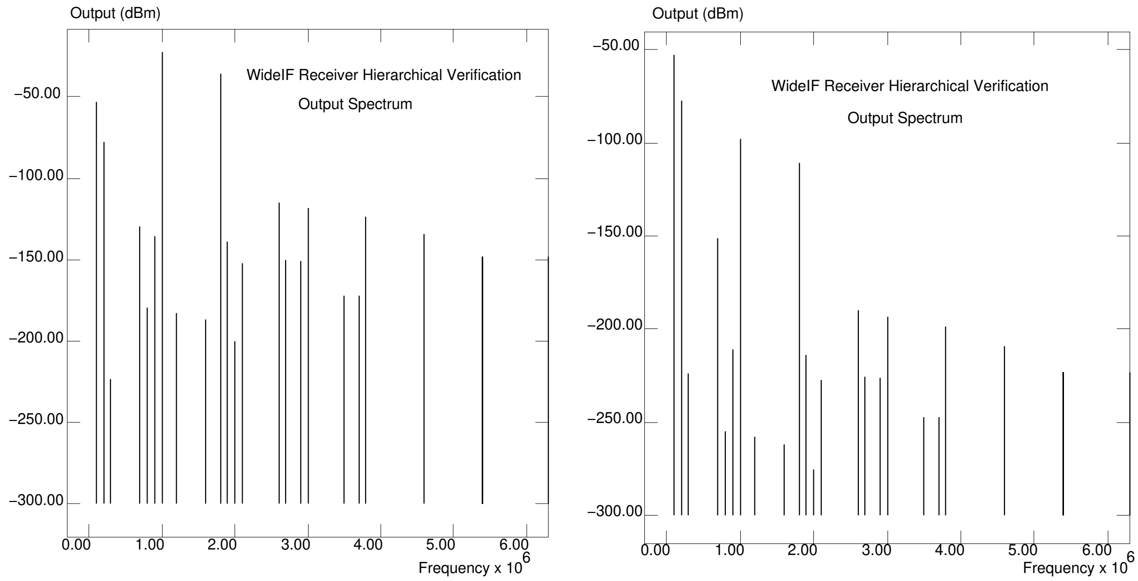
Table 4.9 summarizes the results of the simulation. Figure 4.22 shows the output spectrum with the same input signals as in Table 4.5. The results are within the specifications given. Since the actual module designs were more conservative than the specifications, the overall predicted nonlinearity is improved. Even though a full SpectreRF simulation was not possible, given the simulation time for the cascade of mixers that used a 5 MHz fundamental frequency, it is estimated that it could take over 20,000 CPU seconds, just by using a smaller fundamental frequency (0.2 MHz). Adding the extraction time to the VHsim simulation, this implies a speed-up by a factor of *thirty*.

It is easily seen that VHsim can simulate accurately and in reasonable time extremely complex problems that cannot be handled otherwise. Instead of having to simulate a large and stiff systems [10], smaller blocks can be simulated in reasonable time, extracting the necessary information needed for VHsim.

[10]frequencies differ by orders of magnitudes

| Test | *CPU Time* (s) | Problem Size | # harmonics | SNDR (dB) | $IP_{3TOTAL}$ |
|---|---|---|---|---|---|
| $IM_3$ (2 tones) | 29.3 | 6751 | 157 | - | -10.75dBm |
| $IM_3$ (3 tones) | 159.3 | 19274 | 419 | 24.5 | -10.75dBm |

Table 4.9: VHSIM Wideband IF Extracted Simulation Results




(a) Without Digital Filter (b) With Digital Filter

Figure 4.22: Simulator Output for Wideband IF Receiver (Extracted Modules) - 3 tones

## 4.8 Conclusions

A frequency-domain, "black box" representation of deterministic effects of RF systems has been described. The modeling is based on the Volterra theory of nonlinear systems. The behavioral harmonic balance simulator developed based on those models, is significantly faster than time-domain macro-modeling and can accurately simulate deterministic effects such as intermodulation and frequency translation.

Unlike previous work [84], values for the Volterra coefficients *are obtained directly from specifications or extracted performances.* Since no costly differential equation-based models have to be used, the computation time of the Jacobian for the *harmonic balance* system of equations is greatly reduced compared to standard harmonic-balance implementations. The Volterra coefficients are used purely at the *behavioral level*, not a the device level, which results in small nonlinear algebraic systems that can be solved very efficiently.

The simulator can be used both in system design to partition high-level specifications of RF systems to blocks, as well as in system verification given extracted high level, frequency domain performance parameters. This makes it suitable to use in the context of our proposed hierarchical methodology for RF systems. System-level verification results compare accurately with available behavioral simulation techniques and full transistor simulations, while great improvement in simulation time is observed. The results are also presented in [106].

# Chapter 5

# Design of A Voice Mail Pager System

## 5.1 Introduction

The continuing growth of demand for large integrated electronic systems requires more efficient system-level design methodologies and design flows, that exploit hierarchy, make efficient use of design automation technology to understand trade-offs at every level of the design hierarchy, and allow for re-use of existing software or hardware modules.

Most system-level design methodologies are focused purely on digital implementations. The hardware-software co-design methodology proposed in [107] addresses the design of complex hardware-software embedded systems.

However, in most modern wireless communication systems, the analog RF front end is the most challenging design task. Systematic consideration of the trade-offs of the analog portion of the system from the top-most level of the hierarchy, can significantly affect the optimization of the analog design, making the overall system implementation easier.

So far, the proposed hierarchical design methodology for analog and mixed-signal systems has been demonstrated for relatively simple analog blocks [2, 3] and medium complexity, application-specific mixed-signal systems (Chapter 3, [82]).

To facilitate the design of emerging mixed-signal systems, that combine programmable digital functions as well as analog interface circuits for wireless, high-frequency communication, it is desirable to use both the aforementioned methodologies. In this chapter, the design of a *voice-mail pager* system is used as a vehicle to develop a methodology for the efficient design of complex wireless embedded systems. The overview of the overall methodology is briefly discussed and a complete system design of the RF portion of the pager system is presented.

The methodology can be considered as the system-level expansion of both the hardware-software co-design methodology and the analog top-down, constraint-driven design methodology. It is also perfectly compatible with the goals of the Virtual Socket Interface Alliance ($\mathrm{VSI}_{TM}$) [108] since it focuses on re-usability of existing modules and provides a hierarchical interface by which those modules can be included.

In 5.2 the POLIS co-design methodology and the goals of the $\mathrm{VSI}_{TM}$ Alliance are briefly discussed to provide the necessary background. Section 5.4 gives an overview of the methodology. Section 5.5 briefly discusses the higher levels of the design hierarchy, from functional specifications to communication protocol design and selection of the physical layer parameters. Finally, 5.6 presents in detail the methodology for the RF portion of the system.

## 5.2 Background

### 5.2.1 Hardware/Software Co-design Methodology

To facilitate the design of hardware-software systems, the POLIS design methodology and system has been developed [107]. The system is based on a formal model of designs that is implementation independent and allows high-quality optimization in subsequent phases of the design process. The design process itself is formalized as a "self-replicating" process of function and architecture selection followed by mapping of one into the other. The system has been tested on control dominated applications, such as engine control in automotive systems and on mixed data-control applications such as Digital Video Broadcasting.

### 5.2.2 VSI Design Paradigm

The basic goal of the VSI $\mathrm{Alliance}_{TM}$ [108] is to provide an appropriate interface for the re-use of intellectual property modules. The modules can be included in various levels of the design hierarchy, from an abstract VHDL description to a functional layout.

| *Specification* | *Value* |
|---|---|
| system description | 1-way, voice + text |
| operating radius | 1 km |
| # users | 1000 |
| messages/day/user (voice) | 10 |
| messages/day/user (text) | 2 |
| message length (voice) | [5-120 sec] |
| peak period | 8 hours |
| # lost messages | $\leq 1\%$ |
| service time | $\leq 2$ sec |
| voice quality | "telephone" |

Table 5.1: Voice Mail Pager Specifications

There is also an ongoing effort to include analog and mixed-signal modules. The presented voice-pager system provides a perfect opportunity to test the functionality of such interfaces in a realistic design and explore possibilities of analog and mixed-signal system interfaces.

## 5.3 System Description

The system is a one-way voice-mail pager system, intended to operate at industrial sites and serve about 1000 users at a radius of about 1 km. There is only one base station transmitting simultaneously to all users. Users can receive both voice and text messages. Table 5.1 gives an overview of the system specifications from a user perspective. The specifications represent typical user requirements for paging traffic during working hours for a medium size industrial cite. To simplify the problem, there is no two-way paging, so the acceptable number of lost pages has to be quite low.

## 5.4 Methodology Overview

The layered structure of a typical wireless communication system is shown in Figure 5.1. The Application layer (APP) handles the input data and compresses/decompresses the voice. The Media Access Control (MAC) layer packs the bits into frames and super-frames and uses the appropriate control for channel hopping and addressing. The Physical (PHY) layer modulates/demodulates bits into symbols. Finally the RF layer is responsible for the analog /digital conversion, and transmission/reception of the modulated signal.

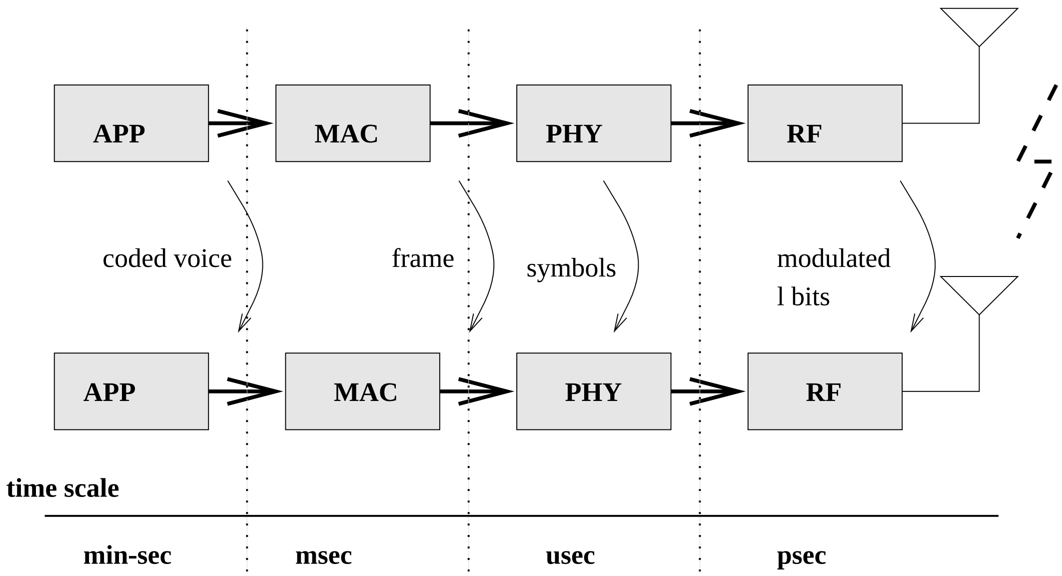



Figure 5.1: System Functional Diagram

The functionality of each layer can be mapped onto software or hardware. Typically, the physical "Channel" follows the Physical layer of the system, but in this diagram the RF link is being shown explicitly since it is part of the system design. Once the functionality of all the layers has been specified, a hierarchical design methodology can be followed for the implementation.

However, there are numerous trade-offs involved in the partitioning of the functionality between the layers. A lot of design variables, specified for example at the MAC protocol layer, can be translated to both functionality specifications for a specific hardware/software block, and constraints for the RF layer. For instance, the frame error rate (FER) decided at the APP layer serves as a constraint for the RF layer, the bits/slot/frame is a constraint for the MAC layer, whereas the QCELP compression algorithm, also specified at the same layer, can be directly mapped to a hardware or software implementation.

A typical IC design flow is shown in Figure 5.2. The layers described above are part of the functional requirements. Since there is also hierarchy in the process of specifying the requirements for each layer, the overall design flow for the voice mail (VM) pager system can be represented as in Figure 5.3. Each layer hierarchically generates both constraints for the subsequent layers, and implementation requirements.

Taking into account the hierarchy within the functional description of the system,

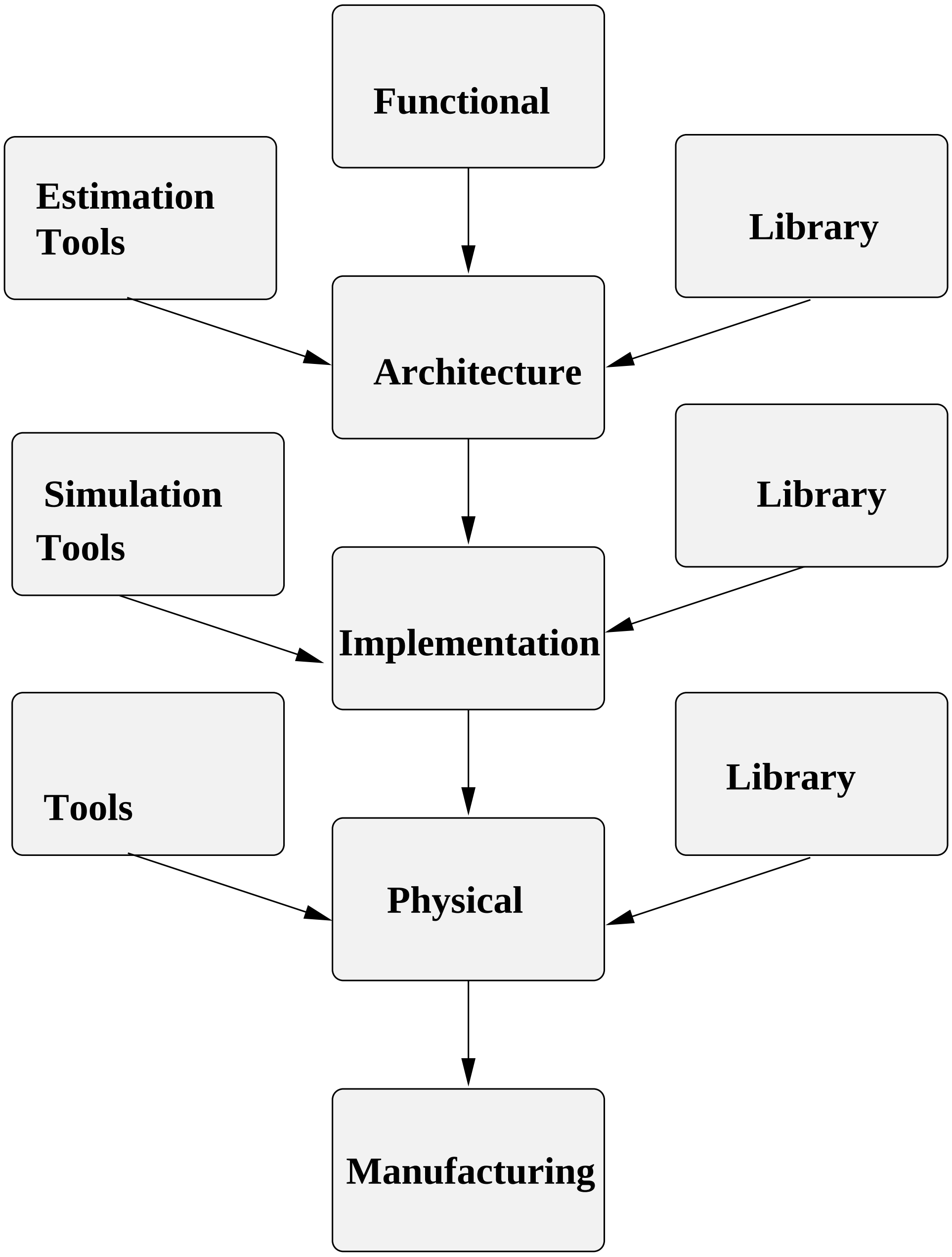



Figure 5.2: Typical Design Flow

there are roughly five behavioral levels in the pager design process: at the APP layer, given the user constraints and additional FCC constraints, behavioral simulation in C-code is used to specify a specific speech compression and decompression algorithm, a user slot length per frame, and an acceptable frame error rate (FER). Given the requirements of the APP layer and the implementation constraints, mainly FCC and environmental constraints, behavioral simulation is used, to define in more detail the protocol at the MAC layer. FCC, environmental and other constraints generated at the APP, MAC layers, are used to define in more detail the modulation scheme, detection scheme and acceptable carrier to interference ratio ($C/I$) at the PHY layer. Finally, an architecture and a specific implementation of the architecture are defined at the "Architecture", "Implementation" levels of the hierarchy. The final implementation can contain software, digital hardware and analog hardware blocks. The top-down analog design methodology can be used for the analog portion, and the POLIS methodology for the digital portion of the system. The above process is depicted schematically in Figure 5.3.

### 5.4.1 Design Goals

It is clear that different solutions can be selected at each level of the hierarchy depending on the design goals that are set. For the pager design, those goals are:

1. "ease" of implementation - faster time-to-market
2. possibility of reuse of designed modules
3. power minimization

It is important to weight appropriately those goals and be consistent with the overall objective at every stage of the design. For example it does not make sense to optimize at the highest level for reuse and immediately after for power, resulting in a system that is neither consistent with power minimization nor reuse maximization.

### 5.4.2 Constraint Translation

At each level of the design hierarchy, a set of performance constraints passed from the higher level and a set of "local" implementation constraints, are being translated onto variable values that become constraints for the subsequent level of the hierarchy. The

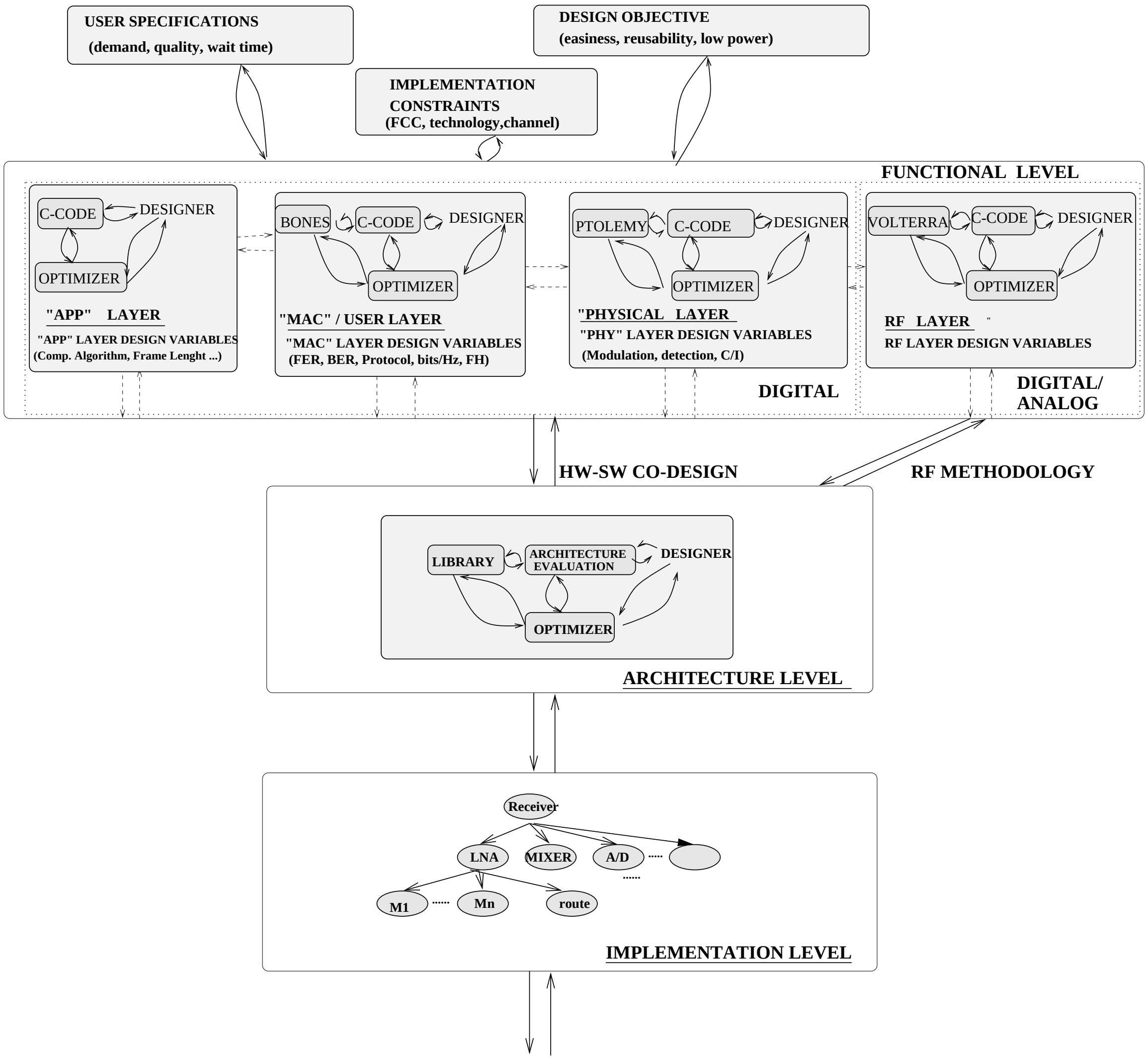



Figure 5.3: Methodology “Tree”

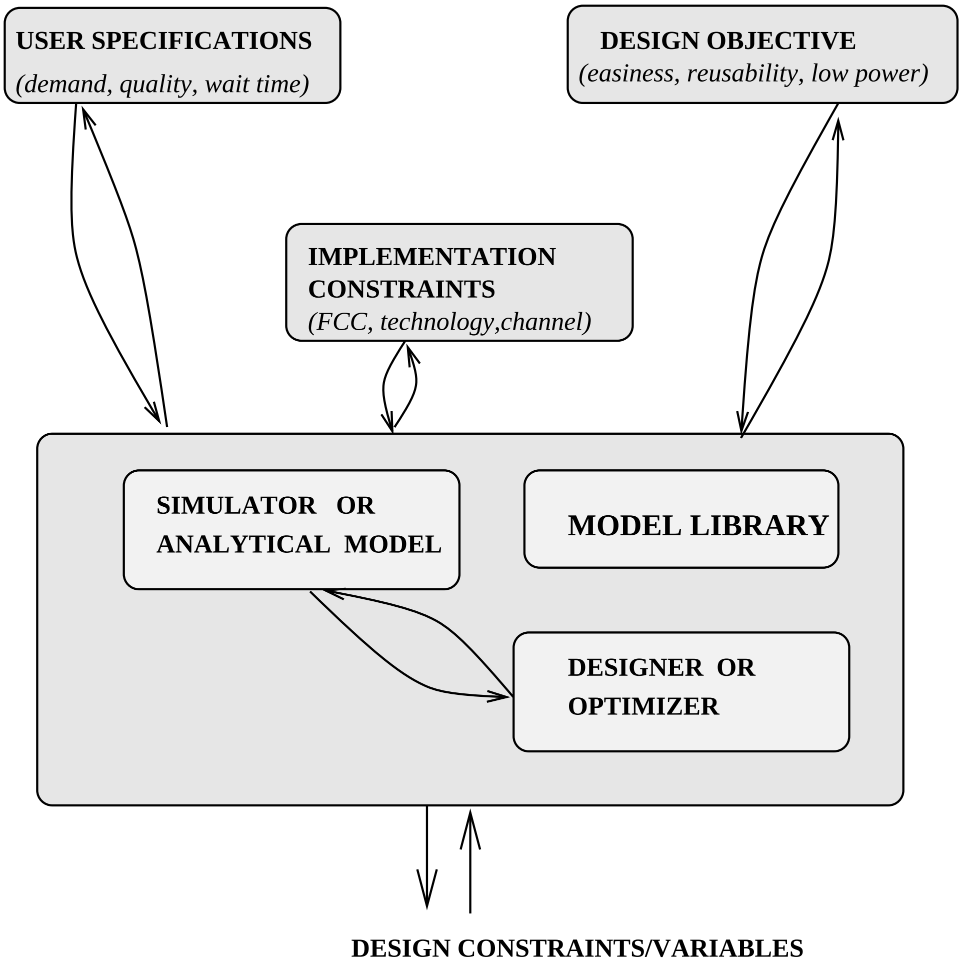



Figure 5.4: Constraint Translation

"constraint" translation mechanism is being done by some kind of performance evaluation that could be obtained analytically or by simulation. Since many solutions may exist for a constraint translation problem, a design objective is needed at every level; such objectives can be heuristic at the higher levels, such as "ease" of implementation, re-usability, or more accurate such as area and power at the transistor level. Such a heuristic objective could be lumped at a "flexibility" function (Section 3.4.4). The heuristic high-level objective is needed for a fast high-level architectural evaluation and partitioning of constraints.

Naturally, overall optimality cannot be guaranteed, but a feasible solution can be found in an inexpensive way, without having to go into implementation details. Given the desired objective, a set of parameters can be selected either automatically by using optimization, as in 3.4.5, or manually using "designer's judgment". In both cases, adequate high level simulation can guarantee that the solution is feasible. It is therefore critical to identify the variables ("constraints") to be used at each level, so that the hierarchy is clearly defined, and also understand the effect of those high level design variables to our design constraints and objectives. For instance, to consider the effect of the choice of a specific modulation scheme such as GMSK, we need not only adequate system-level simulation tools, but also a way to quantify the effect of this choice to the RF subsystem. Such effects can be either qualitative, such as making the RF implementation "harder" from a designer's standpoint and more likely to consume more power, or quantitative, requiring properties such as specific Noise Figure (NF) and third order intermodulation performance ($IM_3$). To consider such effects we need to show how to link existing tools in a specific design flow, show what variables are used, and quantify high-level design trade-offs. Possibly new tools and techniques could be developed to assist the design flow, for example a way to "link" communication, system-level simulation such as PTOLEMY, with RF simulation such as SPECTRERF.

## 5.5 VM Pager Hierarchical Design

### 5.5.1 Functional Layers

#### 5.5.1.1 Application Layer (APP)

At the highest level of the design hierarchy, the user specifications of Table 5.1 and additional environmental constraints, have to be mapped onto high-level design variables. Due to FCC restrictions, the system can only operate in one of the unlicensed bands. For all design objectives, ease of implementation, re-usability and power minimization, it makes more sense to operate at the lowest unlicensed band, the 902 MHz - 928 MHz ISM band. Hence, we consider the FCC constraints of this band as our top-level "environmental" constraints. The two specified ways to access the ISM band is either by frequency hopping (FH) or by code division multiple access (CDMA). Frequency hopping is selected, since the data have to be processed at a much lower rate than CDMA. Processing data at the chip rate would complicate the transceiver design and increase the power consumption.

A summary of the constraints due to the FCC regulations for FH is given in Table 5.2. Notice that a constraint of interference of other users using this band is added.

| *Constraint* | *Value* |
|---|---|
| hop channels | $\geq 50$ |
| BW per channel | $\leq 500\ kHz$ |
| peak power | $\leq 1\ W$ |
| hop time | $\leq 0.4\ sec/channel/20sec$ |
| out-of-band rad. | $\leq 210\ uV/m$ |
| in-band interference | $\geq yyy$ |

Table 5.2: 902-928 Band FCC Constraints

| *Variable* | *Value* | *Objective* |
|---|---|---|
| comp. algorithm | QCELP 9600 bits/sec | easiness, power |
| FER | 0.02 | easiness, power |
| BER | 0.001 | easiness, power |
| bits/frame/user | 600 | easiness, power |

Table 5.3: "APP" Variables

This constraint is mostly unknown and has to be determined either by measurements or by making some general assumptions.

At this layer the high-level variables to be decided are frame length, algorithms to be used for voice coding and decoding and acceptable error rates. A C-model of the *QCELP* compression algorithm is used in order to decide the acceptable frame error rate and frame length. The variables that have to be decided so that the constraints of Table 5.1 and Table 5.2 are satisfied, are summarized in Table 5.3.

In the variable selection process, the objective is to maximize the ease of the design, keeping the power at a reasonable level. When the parameter value does not affect the ease of the design, power minimization is selected as the objective. The "cost" of the specific parameter set can either be determined heuristically by the designer or models according to 5.4.1 can be used.

#### 5.5.1.2 Media Access Control Layer (MAC)

The voice and control information are transmitted in frames. Each frame carries a number of user message slots and a number of frames and several frames are organized in a super-frame. The variables that have to be decided, so that the constraints of Table 5.1 and Table 5.2 are satisfied, are summarized in Table 5.4. Statistical models in BONeS [109] are

| *Variable* | *Value* | *Objective* |
|---|---|---|
| # hop channels | 50 | easiness, power |
| comp. algorithm | QCELP<br>9600 bits/sec | easiness, power |
| FER | 0.02 | easiness, power |
| BER | 0.001 | easiness, power |
| slots/frame | 8 | easiness |
| bits/ slot | 600 | easiness, power |
| frames/super-frame | 32 | power |
| modulation efficiency | 1 bit/sec/Hz | |
| hops/frame | 1 | easiness, power |
| channel bit rate | 240 kbps | (dependent) |
| user channel access | FDMA | power |
| user bit rate | 30 kbps | (dependent) |
| hop rate | 20 msec | (dependent) |

Table 5.4: “MAC” Variables

used to simulate the service time, the overflow of messages and the number of lost messages. The parameter values determined are summarized in Table 5.4, together with the objective that was used for the selection of the specific parameter.

Figure 5.5 briefly describes the organization of the MAC protocol. The paging traffic, taking into account the protocol overhead, requires an aggregate bit rate of 250 Kbit/sec. There are 32 frames per super-frame. At the beginning of each super-frame, control information is given about paging messages (address, channel). Each frame contains control information and message slots of 600 bits each. Each message slot is at a different frequency so a receiver only needs to operate at the message bit rate, not at the aggregate bit rate. The frequency hopping rate is made equal to the frame rate to reduce the protocol implementation complexity and is kept low to minimize power consumption.

Typically, more than one parameter configurations are evaluated. For example, first we can evaluate the “cost” of an optimal configuration of protocol-related variables that uses TDMA channel access for the user and then the cost of a configuration that uses FDMA. The “cost” of the specific configuration can either be determined heuristically by the designer or according to 5.4.1.

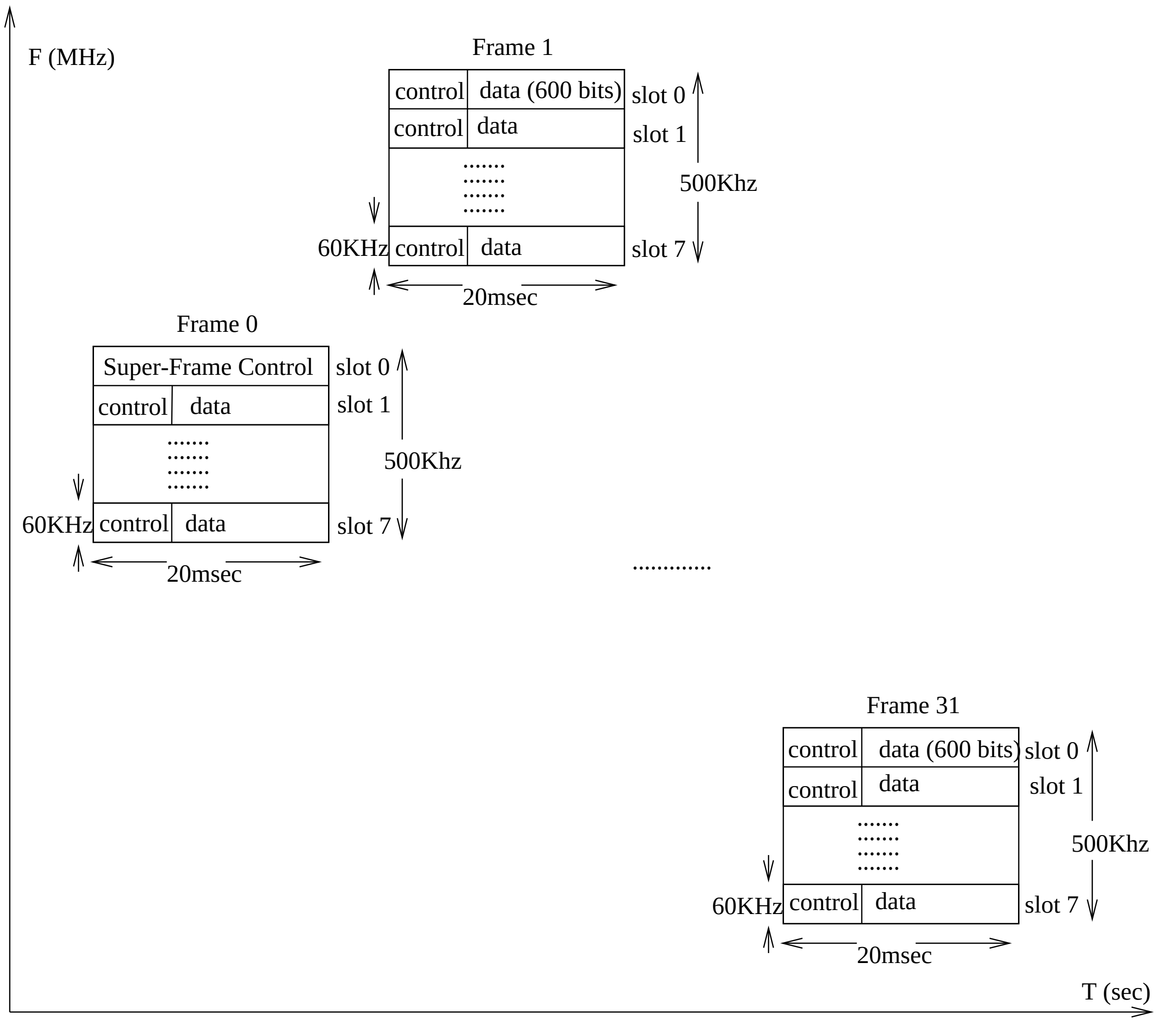



Figure 5.5: Pager Protocol

| *Variable* | *Value* | *Objective* |
|---|---|---|
| modulation | GMSK | re-usability |
| detection | incoherent | re-usability, power, easiness |
| "channel noise"(C/I) | 10dB | power |

Table 5.5: "PHY" Variables

#### 5.5.1.3 Physical Layer (PHY)

At this layer the specific modulation and detection scheme are determined. Also a more detailed "channel" model, taking into account interference is used, in order to derive specifications for the performance of the RF front-end transmitter of the system, such as an overall $C/I$ requirement so that the BER requirement is satisfied. The system-level simulation of the communication system including the channel interference effects can be done with PTOLEMY and again a heuristic "cost function" can be used according to 5.4.1. A summary of the variables determined at this layer is given in Table 5.5. As "constraints" or design constants, the variables of the previous layers are used, together with the FCC/environmental constraints of Table 5.2.

#### 5.5.1.4 RF Layer

At the cost function of the APP, MAC, PHY layer high level parameters, the following trade-offs related to the the analog RF front-end implementation should be considered:

**compression algorithm:** low bits/sec so overall bandwidth is minimized and simpler modulation techniques can be used to simplify the RF requirements. Requires more DSP and better bit error rate.

**hops/frame:** as low as possible for relaxing the frequency synthesizer requirements and reducing lost bits due to synchronization. The limit is FCC regulations and interference tolerance.

**access scheme/slot:** TDMA would be simpler, since no narrow-band channel selection has to be made. However, FDMA is used to minimize the processing rate of the user and, as a consequence, the user power dissipation.

**slots/frame:** low, so that if FDMA is used, channels are not very close and filtering is easier. If slots/frame is too low, the average delivery time for a page may be unacceptable.

**frames/super-frame:** high, so power is minimized (RF is only active once per super-frame when not receiving). The limit is the delivery delay requirement.

**modulation efficiency:** low, so constant envelope schemes such as FSK or GMSK can be used to simplify the RF implementation. GMSK rather than FSK is used for re-usability though.

The choice of the above variables limits the possible RF architectures and puts significant limitations in RF implementation. The design methodology for the RF layer is being discussed in 5.6.

### 5.5.2 Architecture/Implementation Layer

After the details of the functionality have been determined at the functional layers, a specific hardware/software architecture and implementation have to be defined. Determining the trade-offs between different architectures requires high-level evaluation of the specific implementation of the architecture. For this reason those layers are considered simultaneously.

At first, a rough architecture has to be chosen; it may consist for example of a micro-processor, a DSP core, a bus, a memory, an FPGA for more specialized hardware operations, and an RF path as in Figure 5.6. The "cost" of the architecture is evaluated by taking into consideration a combination of:

- availability of blocks,
- estimated power consumption
- estimated re-usability and ease of design for the blocks that are not already available.

However, to be able to evaluate the functionality of the architecture, some block-level implementation details are needed, such as type of logic used in the FPGAs, type of software running in the $\mu P$ and the DSP core and type of RF block architecture used.

Once the architecture has been decided, the next step is to specify details about the implementation; for example what specific block architecture will be used for the RF portion, or how the DSP will be programmed and be interfaced with the FPGA to performed the specified functional tasks within the timing and power constraints given.

For the digital part of the implementation, the Polis hardware-software co-design methodology can be used or the commercially available VCC [110].

For the analog portion of the architecture, different tools have to be used. However it is crucial to be able to evaluate trade-offs between the analog and the digital domain; for example, the analog RF front end architecture can have a digital final demodulation stage, so the cost of the speed/power/difficulty of the digital implementation has to be considered for the specific analog architecture. For this reason the two sub-systems, the analog and the digital have to be considered simultaneously as implied in Figure 5.3.

The analog sub-system has to pass to Polis the following information:

- A/D #bits
- sample rate
- "channel" noise model due to analog system
- other control information for the DSP, for example DSP-based offset correction requirements, or DSP-based gain control requirements

At the *implementation* layer, a significant difference between analog and digital constraints is that the former are basically second order effects, such as noise and distortion, whereas the latter are functional constraints. The second order effects of the analog subsystem are modeled as "noise" at the higher layer of the hierarchy, the *physical* layer.

## 5.6 RF Design Methodology

To define an RF architecture the following process can be followed:

1. define accurately a model for the environmental "interference" ("Channel model");
2. define specific performance constraints such as BER, power
3. define objectives, such as re-usability, potential multi-standard capability, power minimization;

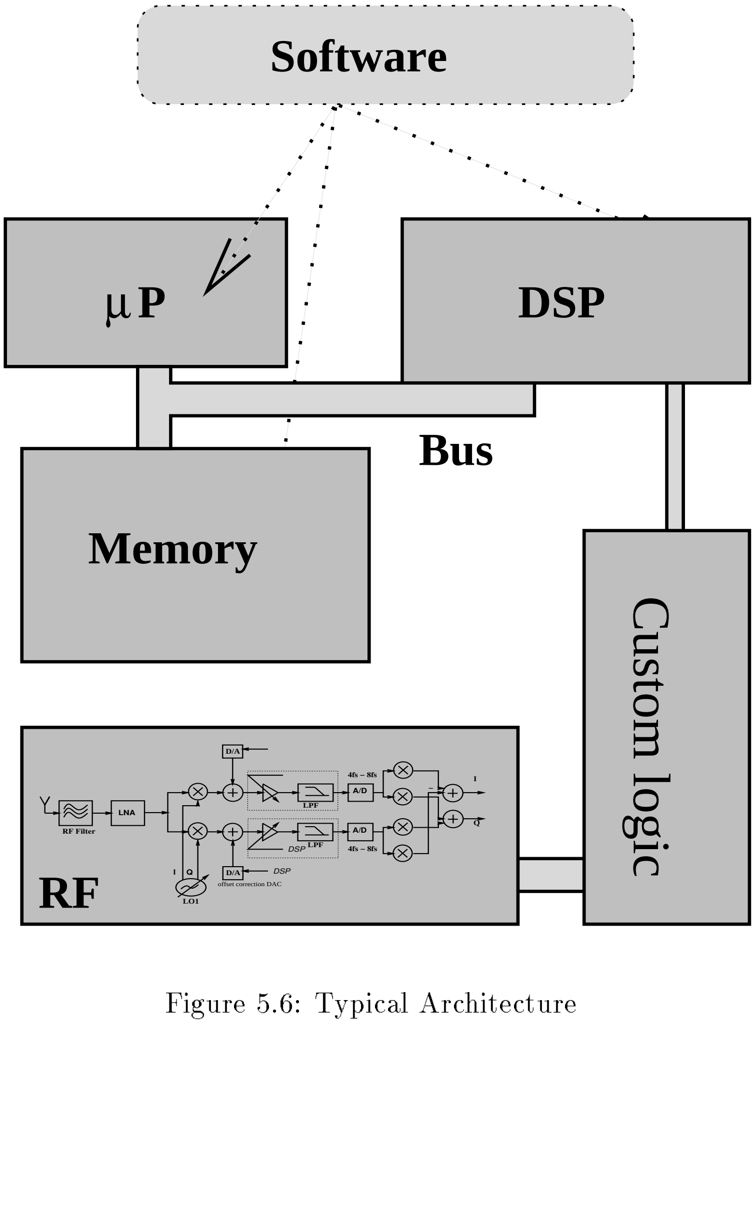



Figure 5.6: Typical Architecture

4. from 1,2, define the dynamic range, distortion and noise performance requirements for the RF transmitter;
5. select an architecture of components;
6. partition the requirement of step 4 onto performance requirements for the blocks of the selected architecture;
7. recursively partition each architecture as defined by top-down analog design methodology.

In order to select an "optimal" architecture, the criteria of 5.4.1 can be used. The methodology for the RF receiver design is shown in Figure 4.1. First, high-level simulation (or evaluation) is used to select the architecture with the minimum "cost" or highest "flexibility". Using high-level simulation, the constraints for the RF receiver are mapped onto constraints for its building blocks. The same process is followed for each one of the blocks of the selected architecture, until the hierarchy is exhausted. The hierarchical/recursive process stops until a library module is found meeting the specified constraints, or until it is designed/synthesized.

The performance of an RF receiver is typically specified by the parameters described in 4.2. Various methods can be used to partition those high-level requirements onto RF building block constraints, as described in 4.3. Figure 4.1 shows the constraint decomposition process for an RF receiver. At the lower levels of the hierarchy the process has been described in Chapter 3.

The approach used for this design is analytical approximation, as described in 4.3.1. In this approach, the interaction with the other tools is rather small: the total required SNDR , computed at the "Physical" layer, plus the FCC limitations, are used as constraints for this layer. There is not really a co-simulation with the digital part of the design. The digital simulator uses the DSP requirements of the RF architecture and the A/D resolution and rate, as constraints for the digital implementation. The total SNDR computed, can also be used as "channel noise" in the Physical layer. Such an approximation could be inadequate at the bottom-up verification phase where the various noise and interference contributions can be modeled more accurately.

The information needed from the "digital" simulation to accurately predict the performance of the RF portion are:

- performance of the digital channel filters, and digital demodulation DSP
- DSP offset correction information
- other DSP related information such as gain control steps

### 5.6.1 Design Objectives - High Level Optimization

In 5.4.2, the problem of high-level constraint propagation was discussed qualitatively. To quantify the high-level design trade-offs, the concept of "flexibility functions"( [1], Chapter 3.4.4) can be used. However, since in RF systems power minimization is a crucial goal, a rough estimate of power consumption, given the high-level block parameters, is needed.

Unlike digital blocks, the optimization of power consumption of analog blocks, given performance specifications, is often a hard task and can have different solutions depending on the specific topology and type of the block. Moreover, since analog performance parameters depend on many second order effects, simple analytical formulas are only useful to predict a trend rather than an accurate power performance.

Solutions have been proposed for pipelined A/D converters [111], $\Sigma - \Delta$ A/D converters [112], CMOS switched capacitor baseband filters [113], LC-tuned low noise amplifiers [114] and CMOS Gilbert-cell based mixers [115].

Based on the above, simple formulas are given for common analog blocks. These formulas depend on constants that are process-dependent and can be extracted from the performance of equivalent blocks for a given IC process. The following equations give measure of the power of common analog blocks given high-level parameters:

$$
\begin{aligned}
P_{GILBERT\_CELL} &= \frac{K_1 \cdot V_{IIP_3}}{NF-1} \cdot \left(1 + \frac{K_2}{C_G}\right) && (5.1)\\
P_{LC\_TUNED\_LNA} &= \frac{K_{LNA} \cdot V_{IIP_3}}{NF - 1^{1/2}} && (5.2)\\
P_{VCO} &= \left[\frac{f_0}{\Delta f}\right]^2 \cdot \frac{K_{LO}}{P_{NOISE}} && (5.3)\\
P_{A\_D} &= MAX(K_A N_{bits}, K_B {N_{bits}}^2 f_s, K_C N_{bits} 2^{2N_{bits}} f_s) && (5.4)\\
P_{FILTER} &= K_{FILTER} \cdot f_s \cdot N_{POLES} && (5.5)\\
& && (5.6)
\end{aligned}
$$

where $K_1$, $K_2$, $C_G$, $K_{LNA}$, $K_{LO}$, $K_A$, $K_B$, $K_C$, $K_{FILTER}$ are constants, $V_{IIP_3}$ is the input-referred third order intercept point, NF the Noise Factor, $f_s$ the sampling frequency, $N_{bits}$ the number of bits, $P_{NOISE}$ the phase noise, and $N_{POLES}$ the number of poles. The constants in the above equations can be approximated by fitting measured data of a design of the same architecture in the available process. Having defined the objective function, the high-level optimization problem can be expressed as:

$$
\begin{aligned}
min \quad & \sum_{i=1}^{M} P_i \\
s.t \quad & SNDR_{CASE_1} \geq S_1 \\
\vdots \quad & \\
& SNDR_{CASE_K} \geq S_k \\
& NF \leq N_1 \\
& V_{IIP_3} \geq IP_1 \\
\vdots \quad &
\end{aligned}
\tag{5.7}
$$

where the objective is calculated as in (5.1) and the constraints are evaluated using one of the simulation techniques mentioned in Chapter 4.

### 5.6.2 RF Architecture for VM Pager

The requirements for the RF architectures are:

- meet the constraints set by the previous layers, mainly overall SNDR, given a set of environmental constraints and received signal conditions;
- re-usability, so that the architecture or RF building blocks can be used for a different product;
- programmability, for the same reason as above;
- low power consumption;
- possibility of full integration;
- ease of design;

| *Test* | *Variable* | *Value* |
|---|---|---|
| Sensitivity | min wanted signal (dBm) | -90 |
| Saturation | max wanted signal (dBm) | -20 |
| in-band Blocking | max in-band blocker (dBm) | -37 |
| IM3 | max in-band blockers for $IM_3$ (dBm) | -47 |
| out-of-band Blocking | max out-of-band blocker (dBm) | 0 |

Table 5.6: Environmental Conditions

An evaluation of possible architectures was done, using the above optimization strategy and the analytical method described in 4.3.1. The most straight-forward architecture that can meet the performance constraints is the traditional super-heterodyne receiver of Figure 5.7. However, since it requires many external components and hence high power and low possibility of integration a low-digital IF architecture was also considered.

To set the exact specifications for the receiver, more detailed models of interference are needed. Since the system operates at the unlicensed ISM band the exact conditions are largely unknown so some rough approximations have to be made which are summarized in Table 5.6. The values used for the interferers are "borrowed" from the IEEE 802.11 wireless LAN standard, operating at the 2.4-2.5 GHZ ISM band. The system should maintain a BER = $10^{-3}$ in all cases. Once a prototype is manufactured, a more accurate interference model can be built.

#### 5.6.2.1 Super-Heterodyne Architecture

The traditional super-heterodyne architecture was first considered, since it is the most well-understood and typically has the best selectivity. Even though it may not be the optimal architecture in terms of re-usability or power consumption, it can be the easiest to implement.

The architecture is shown in Figure 5.7. The RF signal is filtered through an RF filter and then down-converted to a fixed 30 MHz IF frequency using an RF local oscillator, tuned at the desired 60 KHz channel. Then, the IF signal is filtered by an external, narrow-band IF filter centered at 30 MHz, amplified by an IF amplifier and finally down-converted to DC using a fixed second local oscillator. Finally the baseband signal passes though a variable gain amplifier and is digitized by an A/D converter.

The optimization problem for selecting the parameters of the super-heterodyne

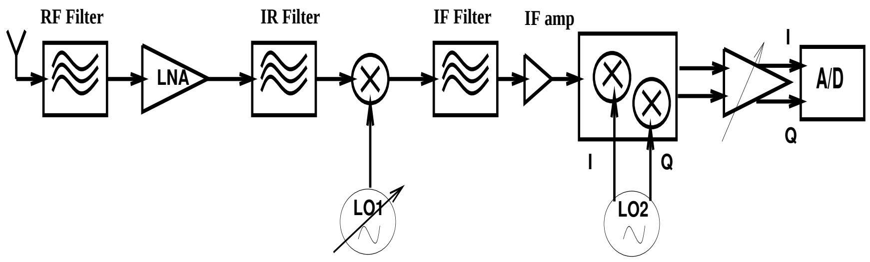



Figure 5.7: Super-Heterodyne Architecture

receiver is:

$$
\begin{aligned}
min \quad & P_{LNA} + P_{MIXER} + \ldots + P_{A/D} \\
s.t \quad & SNDR_{Sensitivity} \geq 10\ dB \\
& SNDR_{Saturation} \geq 10\ dB \\
& SNDR_{Blocking1} \geq 10\ dB \\
& SNDR_{Blocking2} \geq 10\ dB \\
& SNDR_{IM_3} \geq 10\ dB \\
& NF \leq 9 \\
& P_{IIP_3} \geq -15\ dBm \\
& Pin_{max_{LNA}} \leq 0\ \ dBm \\
& \vdots \\
& Pin_{max_{A/D}} \leq 13\ \ dBm
\end{aligned}
\tag{5.8}
$$

The goal of the optimization is to minimize a measure of estimated power consumption maintaining a 10 dB SNDR at various environmental constraints as derived from Table 5.6. To make sure that the analog components do not saturate, additional constraints are given for the maximum input signal that a block can have. Finally, to have a re-usable design, a constraint was also given for the overall noise figure, since from the sensitivity test the noise figure required is pretty relaxed.

To evaluate the constraints, a C++ program was developed implementing the analytical techniques described in 4.3.1. The MINOS [71] nonlinear optimization package was used to solve the problem. Optimization time was negligible, since analytical formulas

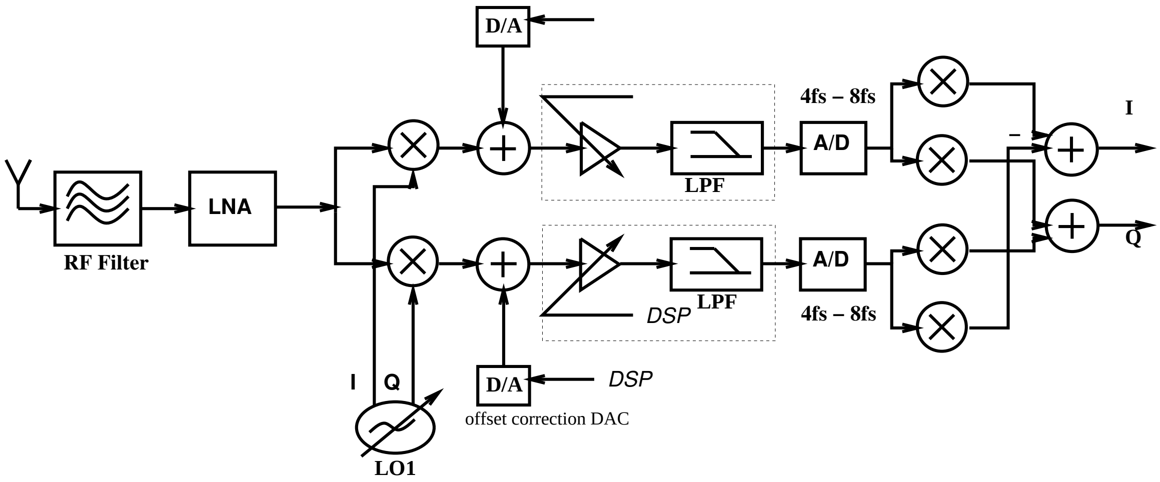



Figure 5.8: Digital IF Architecture

were explicitly given and no behavioral simulation was used inside an optimization loop.

Table 5.7 summarizes the selected parametrized architecture. For off-chip components such as filters, fixed parameters were used matching the performance of commercially available parts. Other RF block parameters were fixed, when they did not affect significantly the optimization problem in order to reduce the complexity. Table 5.8 summarizes the estimated SNDR receiver performance at different cases.

#### 5.6.2.2 Low Digital IF Architecture

A promising architecture that combines programmability, high-integration potential in CMOS, low-power and potential for re-use, is the low digital-IF shown in Figure 5.8. An RF filter is used at the front-end to eliminate out-of-ISM-band interference. After amplification from a low-noise amplifier (LNA), the 500kHz band containing all the 8 transmitted user channels is converted to a low IF of 500KHz, using I-Q mixing and an RF local oscillator (LO) with 500kHz step, centered at RF. The I-Q mixing is necessary at this stage, since the converted band contains all user channels around DC, so image rejection will be necessary at the digital stage. The 500kHz band is filtered by an anti-alias filter and then digitized by a high dynamic range A/D converter. The digitized IF signal is digitally mixed to DC using an image rejection scheme, and the desired user channel is selected with a digital filter. Since the modulated signal is GMSK, there is significant content at DC, so

| *Block* | *Variable* | *Final Value* | *Initial Value* | *Range* |
|---|---|---|---|---|
| *Antenna* | Gain (dB) | 0 | 0 | - |
| | NF (dB) | 0 | 0 | - |
| | input $IP_3$ (dBm) | 13 | 13 | |
| *RF Filter* | Gain | -3 | -3 | - |
| | NF (dB) | 2 | 2 | - |
| | Attenuation (dB) | 30 | 30 | - |
| | input $IP_3$ (dBm) | 10 | 10 | - |
| *LNA* | max Gain(dB) | 20 | 20 | $5 \ldots 20$ |
| | min Gain (dB) | 0 | 0 | - |
| | NF (dBm) | 3.5 | 3 | $2 \ldots 6$ |
| | input $IP_3$ (dBm) | -12.2 | -6 | $-20 \ldots 0$ |
| *Mixer 1* | max conv. Gain(dB) | 6 | 6 | $-6 \ldots 6$ |
| | min conv. Gain(dB) | 0 | 0 | - |
| | NF (dB) | 20 | 10 | $5 \ldots 20$ |
| | input $IP_3$ (dBm) | 10.5 | 20 | $-20 \ldots 20$ |
| *LO1* | phase noise (dBc/Hz) $_{100\ \mathrm{KHz}}$ | -89 | -95 | $-120 \ldots -80$ |
| | channel spacing (KHZ) | 60 | - | - |
| | $F_{LO}$ (MHz) | 902-928 | - | - |
| *IF Filter* | Gain (dB) | -3 | -3 | - |
| | $\sqrt{v_n{}^2}(V)$ | $2e^{-6}$ | $2e^{-6}$ | - |
| | Attenuation (dB) | 60 | 60 | - |
| | input $IP_3$ (dBm) | 15 | 15 | - |
| *IF Amp* | max Gain(dB) | 20 | 20 | - |
| | min Gain (dB) | 20 | 20 | - |
| | NF (dBm) | 19 | 19 | - |
| | input $IP_3$ (dBm) | 10 | | |
| *Mixer 2* | max conv. Gain(dB) | 0 | 0 | - |
| | min conv. Gain(dB) | 0 | 0 | - |
| | NF (dB) | 30 | 30 | - |
| | input $IP_3$ (dBm) | 20 | 20 | - |
| *LO2* | phase noise (dBc/Hz) $_{100\ \mathrm{KHz}}$ | -80 | -80 | - |
| | channel spacing (KHZ) | fixed | - | - |
| | $F_{LO}$ (MHz) | 30 | - | - |
| *VGA Amp* | max Gain(dB) | 30 | 30 | - |
| | min Gain (dB) | 0 | 0 | - |
| | $\sqrt{v_n{}^2}(V)$ | $2e^{-5}$ | $2e^{-5}$ | - |
| | input $IP_3$ (dBm) | 30 | 30 | - |
| *A/D* | Gain | 0 | - | - |
| | # bits | 10 | 12 | $4 \ldots 12$ |
| | Nyquist rate | 0.25 Ms/s | - | - |

Table 5.7: Super-heterodyne Architecture RF Block-Level Parameters

| *Test* | SNDR (dB) |
|---|---|
| Sensitivity | 27 |
| Saturation | 49 |
| in-band Blocking | 10.5 |
| IM3 | 10 |
| out-of-band Blocking | 17.4 |
| *Performance* | *Value* |
| Total Noise Figure | 8.85 dB |
| Total input $IP_3$ | -8.55 dBm |
| Total "Power" (final) | 0.035 |
| Total "Power"(initial) | 0.143 |
| Max Gain | 77 dB |
| Min Gain | 11 dB |

Table 5.8: Super-heterodyne Receiver Performance

offset and 1/f noise can cause interference at the user channels around DC. For this reason a decision-based offset removal scheme is used, controlled by the DSP.

The most significant drawbacks of this architecture are the typical disadvantages associated with direct conversion: LO leakage, time varying offsets, 1/f noise and programmability of the LO at the RF frequency.

The optimization problem for selecting the parameters is similar to the one for the super-heterodyne receiver in (5.8).

Table 5.7 summarizes the values for the components selected. For some blocks that are off-chip fixed parameters were used matching the performance of commercially available parts. Other RF block parameters were fixed when they did not affect significantly the optimization problem to reduce the optimization complexity. Table 5.8 summarizes the estimated SNDR receiver performance at different cases and the value of the objective function.

## 5.7 Conclusions

The general guidelines for the design methodology of a pager system were presented, as a means to develop a more general methodology for wireless embedded systems. Eventually, the goal is to merge the analog top-down design methodology with the Polis hardware software co-design methodology.

| *Block* | *Variable* | *Final Value* | *Initial Value* | *Range* |
|---|---|---|---|---|
| *Antenna* | Gain (dB) | 0 | 0 | - |
| | NF (dB) | 0 | 0 | - |
| | input $IP_3$ (dBm) | 13 | 13 | - |
| *RF Filter* | Gain | -3 | -3 | - |
| | NF (dB) | 2 | 2 | - |
| | Attenuation (dB) | 30 | 30 | - |
| | input $IP_3$ (dBm) | 10 | 10 | - |
| *LNA* | max Gain(dB) | 20 | 20 | $10 \dots 20$ |
| | min Gain (dB) | 0 | 0 | - |
| | NF (dBm) | 2.1 | 3 | $2 \dots 6$ |
| | input $IP_3$ (dBm) | -14 | -6 | $-20 \dots 0$ |
| *Mixer* | max conv. Gain(dB) | 6 | 0 | $-6 \dots 6$ |
| | min conv. Gain(dB) | 0 | 0 | - |
| | NF (dB) | 16.5 | 12 | $6 \dots 20$ |
| | Image Rejection (dB) | 30 | 30 | - |
| | input $IP_3$ (dBm) | 9.8 | 0 | $-20 \dots 20$ |
| *LO1* | phase noise (dBc/Hz) $_{100\ \mathrm{KHz}}$ | -88 | -100 | $-120 \dots -80$ |
| | channel spacing (KHZ) | 500 | - | - |
| | $F_{LO}$ (MHz) | 902-928 | - | - |
| *AA Filter* | max Gain (dB) | 60 | 50 | $20 \dots 60$ |
| | min Gain (dB) | 30 | - | - |
| | $\sqrt{{v_n}^2}(V)$ | $10e^{-5}$ | $5e^{-5}$ | $1e^{-5} \dots 15e^{-5}$ |
| | Attenuation (dB) | 60 | - | - |
| | input $IP_3$ (dBm) | 20 | 20 | - |
| *A/D* | Gain | 0 | 0 | - |
| | # bits | 10 | 8 | $4 \dots 12$ |
| | Nyquist rate | 1.5 Ms/s | - | - |

Table 5.9: Low IF Architecture RF Block-Level Parameters

| *Test* | SNDR |
|---|---|
| Sensitivity | 27 |
| Saturation | 59 |
| in-band Blocking | 10 |
| IM3 | 10 |
| out-of-band Blocking | 22.1 |
| *Performance* | *Value* |
| Total Noise Figure | 9 dB |
| Total input $IP_3$ | -12.3 dBm |
| Total "Power" (final) | 0.045 |
| Total "Power" (initial) | 0.343 |
| Max Gain | 82 |
| Min Gain | 26 |

Table 5.10: Low Digital IF Receiver Performance

The design of the pager system is done in a hierarchical way. High-level simulation is used for the optimal selection of the design parameters at every level. The high-level variables used and the corresponding design trade-offs were discussed. The Polis hardware-software co-design methodology will be used for the selection of an architecture for the digital part of the design. The analog RF portion of the overall system is also considered at every layer of the hierarchy. A high level simulation and optimization environment for RF systems was used to evaluate specific RF architectures and system specifications were generated for two possible RF implementations.

For the completion of the project the following steps are needed: (1) implementation of analog/digital co-simulation environment, possibly in VCC, so that analog/digital/software trade-offs are evaluated and analog subsystem is co-simulated with the gain control and offset removal algorithms; (2) full architectural mapping of digital/software part; (3) possible design of standard cell/ analog blocks and fabrication.

# Chapter 6

# Conclusions and Future Work

## 6.1 Conclusions

The goal of the research described in this dissertation was to develop and demonstrate an efficient methodology for the design of mixed-signal and analog RF systems. The foundation for this work was the top-down, constraint-driven design methodology for analog circuits [1].

Based on that, we developed behavioral simulation and optimization techniques to apply the methodology in a complex mixed-signal design such as a video driver system with on-chip programmable clock generator. In our design example, we were able to efficiently use a variety of constraint-driven analog CAD tools that were developed in the context of the methodology [4]. Such tools included behavioral simulation, automatic D/A synthesis from specifications, non-Monte-Carlo noise simulation and constraint-driven analog layout synthesis. We were able to successfully account for effects such as timing jitter, early in the design cycle, by using behavioral simulation techniques. The overall result was encouraging, demonstrating a significant reduction in the overall design and fabrication time, and producing working fabricated parts from the first run. We showed that our methodological approach to analog and mixed-signal CAD, can have an impact in the design of large mixed-signal systems, as opposed to being able to apply it only in small modules such as operational amplifiers and D/A converters.

We put the basis to apply our methodology into the cutting-edge area of RF systems by developing a new system-level simulation technique, based on Volterra-Series and harmonic balance. The simulation and modeling is done purely in the frequency domain, as typical specifications are given for RF systems. Unlike time-domain macro-models, it relies on a "black-box" representation that does not depend on implementations details. This makes it suitable to use in the context of our proposed hierarchical methodology; from RF system level specifications in the frequency domain, models are build and simulation can be used to partition top-level specifications onto block-level constraints; at the verification phase, frequency-domain information in the form of look-up tables can be extracted for individual RF blocks and the overall system can be simulated using our approach. The results presented validate the approach in both phases of the design, top-down design and bottom-up verification. At the top-down phase, results compare accurately with time-domain macro-models, while demonstrating a significant improvement in simulation time. At the verification phase, results compare accurately with full transistor simulation, while the simulation time is improved by at least an order of magnitude.

Finally, the design of a voice-mail pager system was initiated as a vehicle to develop a more general methodology for wireless-embedded systems. The goal is to merge the POLIS hardware software co-design methodology, intended for control dominated digital embedded systems, with our analog-RF design methodology. The RF implementation should be taken into account as early as the protocol design phase. We have demonstrated an initial design flow from specifications to protocol design, physical layer design and RF specification generation. We applied our design methodology to the system design of the RF part of the voice-mail pager.

## 6.2 Future Work

The research in an all-encompassing methodology for wireless mixed-signal embedded systems is a very wide area, with many open questions. First of all, the methodology needs to be validated with yet more complex examples, including RF transceivers. Examples can be the drivers to develop and test the necessary tools to support the methodology and also demonstrate the need for additional work in algorithms and tools.

Second, accurate simulation of all effects in RF systems, including noise, is still needed. The Volterra-series based behavioral simulation tool presented here can be enriched with recent developments in RF noise simulation, including cyclo-stationary noise [13] and phase noise [116, 14, 16]. Behavioral models for various types of noise sources can be built, to provide for a full characterization of RF blocks.

Merging the POLIS hardware-software, co-design methodology with our mixed-signal and analog-RF design methodology is also an important direction for future research. To provide better system optimality and explore more architectural options, the partition between hardware/software and analog/digital has to be treated in a unified way, before any decision is made about the actual implementation. Research in the area of wireless protocols that take into account the RF implementation, is necessary for that. The voice-mail pager design example provides a vehicle to investigate those open questions.

To be able to partition a system between analog/digital hardware and software, a more efficient characterization and modeling of analog, continuous time systems within the POLIS - PTOLEMY framework is also needed. Hybrid simulation techniques, that combine event-driven or signal data flow with continuous time equations are also desirable.

Finally, tools and methodologies are widely accepted when they provide a good environment for communication with the user. A unified framework under which most of the tools supporting the methodology can inter-operate, supported by libraries at various levels of the design hierarchy, can have a significant impact in accelerating the design of complex wireless embedded systems.

# Appendix A

# VHSim Input File Format

A brief description of the input file format of VHSim is given in this Appendix. Each line of the input file is the description of an RF block or a control command. RF blocks are characterized as two-ports. The "amp" and "mixer" two-ports are currently implemented. A detailed description of the input file syntax follows:

**RF block models:** models of common RF building blocks such as mixers, amplifiers, sources, impedences:

**source:** $<\mathit{name}>$ source $<p>$ $<n>$ $<\mathit{parameterList}>$

$<\mathit{name}>$**:** name of the source

**source:** keyword showing that the line is a description of a source

$<p>$ $<n>$**:** integers identifying the +, - terminals of the source

$<\mathit{parameterList}>$**:** $<\mathit{parameterName}>$ $<\mathit{value}>$ $<\mathit{parameterList}>$

where $<\mathit{parameterName}>$ is one of the following:

**f:** frequency of the sinusoidal source

**mag:** magnitude of the sinusoidal source

**phase:** phase of the sinusoidal source

**amp:** $<\mathit{name}>$ amp $<p_1>$ $<n_1>$ $<p_2>$ $<n_2>$ $<\mathit{parameterList}>$

$<\mathit{name}>$**:** name of the "amplifier" module

**amp:** keyword showing that the line is a description of an amplifier

$<p_1>$ $<n_1>$ $<p_2>$ $<n_2>$**:** integers identifying the input and output port terminals

<*parameterList*>: <*parameterName*> <*value*> <*parameterList*>

where <*parameterName*> is one of the following:

**av** voltage gain in dB. If <*value*> is a name of a file, the file is read and is assumes to contain gain as a function of frequency.

**gain** voltage gain in dB. In case where a file is given in **av**, **gain** is the passband gain.

**P1dB** 1dB compression point in dBm.

**ip3** input-referred third order intercept point in dBm.

**av3** the name of a file containing values of a third order volterra transfer function at different input frequency combinations.

**ip2** input-referred second order intercept point in dBm.

**av2** the name of a file containing values of a third order volterra transfer function at different input frequency combinations.

**offset** DC offset in Volts

**nf** Noise Figure in dB

**rin** input impedence in Ohms. Can also be the name of a file containing Rin(f).

**rout** output impedence in Ohms or filename as above.

**rref** reference impedence for two-port in Ohms

**ar** reverse *current* gain. Can also be a filename as above.

**mixer:** <*name*> amp <$p_1$> <$n_1$> <$p_2$> <$n_2$> <*parameterList*>

<*name*>: name of the "mixer" module

**mixer:** keyword showing that the line is a description of a mixer

<$p_1$> <$n_1$> <$p_2$> <$n_2$>: integers identifying the input and output port terminals

<*parameterList*>: <*parameterName*> <*value*> <*parameterList*>

where <*parameterName*> is any of the parameters specified for the module "amp" with the following additional parameters/differences:

**lo** the local oscillator frequency for the mixer in Hz

**mainSideBand** +1 for upconversion or -1 for downconversion

**av** the conversion gain in dB, or a filename

**avi** the image gain in dB or a filename

**rfleak** leakage term from input to output (zero-order side-band transfer function) or filename

**spurious2** second order spurious response at $mainSideBand \cdot 2$ in dB

**spurious3** third order spurious response at $mainSideBand \cdot 3$ in dB

**av2i** filename for second order volterra-transfer function at the image band

**av3i** filename for third order volterra-transfer function at the image band

**imp:** <*name*> imp <*p*> <*n*> <*parameterList*>

<*name*>**:** name of the impedence

**source:** keyword showing that the line is a description of an impedence

<*p*> <*n*>**:** integers identifying the +, - terminals of the impedence

<*parameterList*>**:** <*parameterName*> <*value*> <*parameterList*>
where <*parameterName*> is one of the following:

**mag, r:** magnitude of the impedence (ohms) or name of file containing values of Z(f) at different frequencies

**phase:** phase of the impedence

**options command:** options <*optionList*>

**options:** keyword showing that the line is an options line

<*optionList*> <*optionName*> <*value*> <*optionList*>
where <*optionName*> is one of the following:

**fwanted** frequency of wanted output harmonic for computation of SNDR

**fmax** maximum frequency allowed in the harmonic balance algorithm

**abstol** absolute tolerance for convergenve of Newton-Raphson iteration

**reltol** relative tolerance for convergenve of Newton-Raphson iteration

**maxorder** integer for maximum order of nonlinearity alllowed

**maxside** integer for maximum sideband frequency number allowed

**output command:** output $< node_1 1 > ... < node_M >$
where $node_i$ are the nodes for which an output solution file for the node voltage E(f) will be produced

# Appendix B

# SpectreHDL Macro-Models

```
///////////////////////////////////////////////////////////////////////////
//   Mixer model                                                          //
//                                                                        //
//   models Rin, Rout, offsets, ip2, ip3, conversion gain, noise figure //
//         models frequency dependant behavior at IF by specifying       //
//         filter as in "filter.def"                                      //
//                                                                        //
//  Author: Iason F. Vassiliou 9/19/96                                    //
///////////////////////////////////////////////////////////////////////////

#define fkt 1.66e-20
#define tnorm 300
#define Kb 1.3806226e-23
#define pi 3.14159
#define no_value 10000
#define undefined -1

#include  "filter.def"
#include "nlamp.def"
#include "basic_mixer.def"
#include "impedence.def"
module vsource;
module resistor;
module capacitor;
module inductor;
module svcvs;
module vcvs;

module mixer (ina, lo_node, commona, outa)
```

```
               (conversion_gain, f_if1, f_if2, f_if3, f_if4,
                f_ifappr,  f_iftype, f_ifripple, f_ifattn,
                f_iforder, Rin,Cin,Lin, Rout,Cout,Lout , Rnorm,
                ip2, ip3, comp1db, vsat, vos, vdc,
                noise_figure,limit,rel_poly1_2)

node[V, I] ina, lo_node, commona, outa;
parameter real conversion_gain = 20;
                                    // conversion gain   (in dB)

parameter real f_if1= 100e9;
                    // f_if1, f_if2, f_if3 , f_if4  corner frequencies
                    //for passband filter after mixer
parameter real f_if2= undefined;
                    // defaults -1,-1,-1,-1 (no filter), f1,f2 = -1 lowpass
parameter real f_if3= undefined;     // f3, f4 = undefined highpass
parameter real f_if4= undefined;     // filter generation program is used
                    // filter types supported: same as in filter module

   // type of filtering at IF
parameter enum {butterworth,chebyschev,ichebyschev,elliptic} f_ifappr = butterworth;
parameter enum {lowpass, highpass, bandpass, bandstop} f_iftype = lowpass;
parameter real f_ifripple = 0.1;
parameter real f_ifattn = 40;
parameter integer f_iforder = 5;
parameter real Rin = undefined;  // input resistence (Ohm) undefined for "infinite"
parameter real Cin = undefined;  // input capacitance (F), undefined for 0 default
parameter real Lin = undefined;  // input inductance (H), undefined for 0 default


parameter real Rout = 0.0  ;     // output resistence (Ohm)
parameter real Cout = undefined; // output capacitance (F), undefined for 0 default
parameter real Lout = undefined; // output inductance (H), undefined for 0 default

parameter real Rnorm = 50.0;     // normalizing reference resistence for NF


parameter real ip2 = no_value;
//second order intercept point in dBV (output referred)

parameter real ip3 = no_value;
//third order intercept point in dBV (output referred)

parameter real comp1db=no_value;
```

```
//-1dB compression point in dBV    (output referred)

parameter real vsat = 5.0;       // saturation voltage
parameter real vos = 0.0;        // input referred offset (in V)
parameter real vdc = 0.0;        // dc output
parameter real noise_figure = 0.0;
                    // noise figure in dB ; refer to 50ohm input source impedence


parameter real limit = 1;
// point of dB drop from ideal value until polynomial model is
// valid default = 1: model valid until -1dB compression point

parameter real rel_poly1_2 = 0.8;
 // if both ip3, comp-1dB are given rel_poly1_2 defines region
 // of polynomiial giving ip3 relative to the -1dB compression point
 // by default set to 0.8 vin_-1dB

{
 node[V, I] inxa,inya,outya,outka,outla,outza;
 real  vnoise;
 real nvec[4];
 real vdc_total=0.0;

 vdc_total = vdc + vos * pow(10, 0.05* conversion_gain);
vnoise= fkt*Rnorm * (pow(10.0,0.1*noise_figure) - 1.0) ;
nvec[0] = 1.0;
nvec[1] = vnoise ;
nvec[2] = 1000e9;
nvec[3] = vnoise;

 vsource vn(inxa, inya)  (dc=0.0, noisevec=nvec);  //input referred noise

resistor rdum(ina, inxa)  (r=1e-9);
impedence zin(inya,commona) (R=Rin,C=Cin,L=Lin);

vcvs forward(outya, commona, inya, commona) (gain=1.0);

nonlinear_module nl1(outya,outza) (gain = conversion_gain, ip2=ip2, ip3=ip3,
                                    comp1db = comp1db, vsat =vsat, vos =0.0,
                                     vos =0.0 , vdc = 0.0, limit=limit,
                                     rel_poly1_2 = rel_poly1_2);
                                    // nonlinearities and clipping
```

```
basic_mixer mixer1(outza, lo_node, outka) (gain = 1.0, offset=vdc_total);
filter filter_if(outka, outla, commona)  (f1 = f_if1, f2 = f_if2,
                                          f3 = f_if3, f4 = f_if4,
                                          attn=f_ifattn, ripple=f_ifripple,
                                          filtertype=f_ifappr, type =f_iftype,
                                          order = f_iforder);

impedence zout(outla, outa) (R=Rout,C=Cout,L=Lout);


}



//////////////////////////////////////////////////////////////////////
//   nonlinear amplifier model                                      //
//                                                                  //
//   models ip2, ip3, P-1dB, Voffset, clipping; auxiliary model     //
//  to be used with complete amplifier or mixer model               //
//                                                                  //
//  Author: Iason F. Vassiliou 9/19/96                              //
//////////////////////////////////////////////////////////////////////

#define no_value 10000
                      // vout = a1*v+a2*v^2+a3*v^3
                      //includes clipping and offset
                      //model valid to -1 db compression, another polynomial
                      //valid from -1db compression to saturation

module  nonlinear_module(in,out) (gain,ip2,ip3, comp1db, vsat,
                        vos,vdc,limit,rel_poly1_2)
node[V, I] in, out;
parameter real gain = 20;
parameter real ip2=no_value;
// second order intercept point in dBV (output referred)

parameter real ip3=no_value;
// third  order intercept point in dBV  (output referred)

parameter real comp1db =no_value;
// -1dB compression point in dBV    (input referred)

parameter real vsat=5;    // saturation voltage
parameter real vos=0;     // input referred offset
```

```
parameter real vdc=0;     // dc output
parameter real limit = 1;
// point of dB drop from ideal value until polynomial model is valid
// default = 1: model valid until -1dB compression point

parameter real rel_poly1_2 = 0.8;
// if both ip3, comp-1dB are given rel_poly1_2 defines region
// of polynomiial giving ip3 relative to the -1dB compression point
// by default set to 0.8 vin_-1dB

{
 real c1 = 1.0;  // coefficients of nonlinear polynomial
 real c2 = 0.0;
 real c3 = 0.0;
 real c3a = 0.0;   // different 3rd order coefficient when both
                   // ip3, -1dB point are defined
 real c0 = 0.0;

 real d1  = 1.0;
 real d3 = 0.0;
 real d0 = 0.0; // coefficients of nonlinear polynomial at compression

 real v_poly_limit, v_poly_limit_ip3;
 real vin_1db,v_limit,vin_1db_ip3 ;
 real v_limit_1;
 real vout,vin,vmax,vin1;
 real vin_eff;
 integer linear = 0;
 integer both_defined = 0;

initial {


                    // now calcul;ate polynomial coefficients:
                    // from meyers notes
                    // IM2 = second order/1st order = c2x^2/c1x
                    //        IM3 = 3d/1st = 3/4 c3x^3/c1x
                    // for ip im.n = 1, x = pow(10.0, 0.05*ip)

c1 = pow(10.0, 0.05 * gain);

 if((comp1db == no_value) && (ip3 ==  no_value))
{
  if(ip2 < no_value)
```

```
  {

                         // model includes only 2nd order disto

   c2 = 1.0*c1*c1/pow(10.0, 0.05*ip2);
   $warning("2nd order nonlinearity");
       // region of validity:  define the point where 2nd order
       // term = 50% of first order term (not v-1db)

   v_poly_limit = 0.5*c1/c2;
   }

  else             // purely linear model
  {
   vmax = vsat/c1;  // not -1db point, just saturation point
   $warning("linear");
   linear = 1;
  }


}

else if((comp1db == no_value) && (ip3 < no_value))

            // define 3rd order model based on ip3
{
 if(ip2 < no_value)
  c2 = 1.0*c1*c1/pow(10.0, 0.05*ip2);
 c3 = -1.0*c1*c1*c1*4.0/(3.0*pow(10.0, 0.1*ip3));
 $warning("defined from ip3");
  // now define region of validity: conventionally up to -1dB
  //compression assuming c1, c3 having opposite signs

 v_poly_limit = sqrt(-4.0/3.0*c1/c3*(1.0 - pow(10.0, -limit*0.05 )));
 vin_1db = sqrt(-4.0/3.0*c1/c3*(1.0 - pow(10.0, -0.05 )));
}

else if((comp1db < no_value) && (ip3 == no_value))

 // define 3rd order model based on -1dB point
{
  $warning("defined from comp 1db");
 vin_1db = pow(10.0, 0.05* comp1db );
 if(ip2 < no_value)
```

```
  c2 = 1.0*c1*c1/pow(10.0, 0.05*ip2);
 c3 = -4/3*c1/vin_1db/vin_1db*(1.0 - pow(10,-0.05));
 v_poly_limit = sqrt(-4.0/3.0*c1/c3*(1.0 - pow(10.0, -limit*0.05 ) ));

}

else       // ip3, -1db point both defined  use from 0 to vin_1db/2
           // use  c3 defined by ip3 and from vin_1db/2 to vin_1db
           // use c3 defined by -1dB. use co for cuntinuity.

{
 both_defined = 1;
 $warning("both ip3 and comp-1 dB point defined \n using 2 different polynomials");
  vin_1db = pow(10.0, 0.05* comp1db );
$strobe("vin_1db =  %f", vin_1db);
 if(ip2 < no_value)
  c2 = 1.0*c1*c1/pow(10.0, 0.05*ip2);

 c3 = -4/3*c1/vin_1db/vin_1db*(1.0 - pow(10,-0.05));
 c3a = -1.0*c1*c1*c1*4.0/(3.0*pow(10.0, 0.1*ip3));

                           // compare the 2 limits and chose the smaller

 v_poly_limit_ip3 = sqrt(-4.0/3.0*c1/c3a*(1.0 - pow(10.0, -limit*0.05 ) ));
 v_poly_limit  = sqrt(-4.0/3.0*c1/c3*(1.0 - pow(10.0, -limit*0.05 ) ));
 if( v_poly_limit_ip3 < v_poly_limit)
   v_poly_limit =  v_poly_limit_ip3;
 vin_1db_ip3 = sqrt(-4.0/3.0*c1/c3a*(1.0 - pow(10.0, -0.05 ) ));
 if(vin_1db_ip3 < vin_1db) vin_1db = vin_1db_ip3;
 v_limit_1 = vin_1db * rel_poly1_2;
 c0 = (c3a-c3) * v_limit_1 * v_limit_1 *v_limit_1;
}

                      // now define model close to saturation
if(linear == 0)
{

              // model is vout = d0 + d1 vin' + d3 vin'^3
              // vin' = vin - vin_1db
              // conditions : continuity and 1st derivative continuity at
              // vin_1db and vmax, where vmax gives vout = vsat
              // 4 equations and 4 unknowns (d0,d1,d3,vmax)

  v_limit = v_poly_limit;
```

```
  d0 =v_limit * (c1 + v_limit * (c2 +v_limit *  c3))+c0;
  d1 = c1 + 2.0*c2*v_limit + 3.0*c3*v_limit*v_limit;
  vmax = 1.5*(vsat-d0)/d1;
  d3 = -d1/(3.0*vmax*vmax);
}

  $strobe("vin_1db =  %f c1 = %f c2= %f c3= %f c0= %f c3a= %f",
           vin_1db,c1,c2,c3,c0,c3a);
  $strobe("d0= %f d1 = %f d3= %f vlimit= %f  vmax= %f v_limit_1 = %f",
           d0,d1,d3,v_limit,vmax,v_limit_1);
}

analog {
vin_eff = val(in) + vos;
if(vin_eff < 0)
   vin = -1.0*vin_eff;
 else
   vin = vin_eff;

if(linear == 1)          // only linear model with saturation
{


 if(vin <= vmax)
   vout = vin * c1;
 else
  vout = vsat;
}

else if(both_defined == 1)
// 4 regions: for ip3, comp -1dB, before saturation, saturation


{

if(vin <= v_limit_1)
  vout = vin * (c1 + vin * (c2 +vin *  c3a));
else if(vin <= v_poly_limit)
  vout = vin * (c1 + vin * (c2 +vin *  c3))+c0 ;
else if(vin <= vmax+v_poly_limit)

{

 vin1 = vin - v_poly_limit;
```

```
  vout = d0 + d1*vin1 + d3*vin1*vin1*vin1;
}
else
  vout = vsat;
}
else   // only one of ip3, -1db point defined
       // 3 regions: small disto, before saturation, saturation


{

if(vin <=v_poly_limit  ) {
 vout = vin * (c1 + vin * (c2 +vin *  c3));
 if(vout > vsat) vout = vsat;
 }
else if ((vin > v_poly_limit) && (vin <= (vmax+ v_poly_limit )))

 {
  vin1 = vin - v_poly_limit;
  vout = vin1 * (d1 + vin1 * vin1 *  d3)+d0;

 }

else
 vout = vsat;

}

if(vin_eff > 0)
 val(out) <- vout+vdc;
else
  val(out) <- -1.0*vout+vdc;


}

}
```

# Bibliography

[1] H. Chang, A. Sangiovanni-Vincentelli, F. Balarin, E. Charbon, U. Choudhury, G. Jusuf, E. Liu, E. Malavasi, R. Neff and P. Gray, "A Top-down, Constraint-Driven Design Methodology for Analog Integrated Circuits", in *Proc. IEEE CICC*, pp. 841–846, May 1992.

[2] H. Chang, E. Liu, R. Neff, E. Felt, E. Malavasi, E. Charbon, A. Sangiovanni-Vincentelli and P. R. Gray, "Top-Down, Constraint-Driven Methodology Based Generation of n-bit Interpolative Current Source D/A Converters", in *Proc. IEEE CICC*, pp. 369–372, May 1994.

[3] H. Chang, E. Felt and A. Sangiovanni-Vincentelli, "Top-Down, Constraint-Driven Design Methodology Based Generation of a Second Order $\Sigma$-$\Delta$ A/D Converter", in *Proc. IEEE CICC*, pp. 533–536, May 1995.

[4] H. Chang, E. Charbon, U. Choudhury, A. Demir, E. Felt, E. Liu, E. Malavasi, A. Sangiovanni-Vincentelli and I. Vassiliou, *A Top-Down, Constraint-Driven Design Methodology for Analog Integrated Circuits*, Kluwer Academic Publishers, 1997.

[5] L. Nagel, "SPICE2: A computer Program to Simulate Semiconductor Circuits", Ph.D. Thesis UCB/ERL M75/520, University of California at Berkeley, May 1975.

[6] W. J. Cunnigham, *Introduction to Nonlinear Analysis*, McGraw-Hill, New York, 1958.

[7] A. Ushida and L. O. Chua, "Frequency-Domain Analysis of Nonlinear Circuits Driven by Multi-Tone Signals", *IEEE Trans. on Circuits and Systems*, vol. CAS-31, pp. 766–778, September 1984.

[8] R. Telichevesky, K.S. Kundert and J. White, "Efficient Steady-State Analysis Based on Matrix-Free Krylov-Subspace Methods", in *Proc. Design Automation Conference*, June 1995.

[9] R. Telichevesky, K.S. Kundert and J. White, "Efficient AC and Noise Analysis of Two-Tone RF Circuits", in *Proc. Design Automation Conference*, June 1996.

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

[11] P. Feldmann and J.S. Roychowdhury, "Computation of Circuit Waveform Envelopes Using an Efficient, Matrix-Decomposed Harmonic Balance Algorithm", in *Proc. IEEE/ACM International Conference on CAD*, November 1996.

[12] A. Demir, E. Liu and A. Sangiovanni-Vincentelli, "Time-Domain non-Monte Carlo Noise Simulation for Nonlinear Dynamic Circuits with Arbitrary Excitations", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, pp. 493–505, May 1996.

[13] J. Roychowdhury, D. Long and P. Feldman, "Cyclostationary Noise Analysis of Large RF Circuits with Multitone Excitations", *IEEE Journal of Solid State Circuits*, vol. 33, n. 3, pp. 324–336, March 1998.

[14] Alper Demir, Amit Mehrotra and Jaijeet Roychowdhury, "Phase Noise in Oscillators: A Unifying Theory and Numerical Methods for Characterization", in *Proc. IEEE/ACM DAC*, pp. 26–31, 1998.

[15] A. Hajimiri and T. Lee, "A General Theory of Phase Noise in Electrical Oscillators", *IEEE Journal of Solid State Circuits*, vol. 33, n. 2, pp. 179–94, February 1998.

[16] Amit Mehrotra and Alberto Sangiovanni-Vincentelli, "Simulation Techniques for Noise in Non-Autonomous Radio Frequency Circuits, submitted to Design Automation Conference 1999".

[17] K. Kundert, "Simulation Methods for RF Integrated Circuits", in *Proc. IEEE/ACM International Conference on CAD*, November 1997.

[18] G. Casinovi, *Macromodeling for the Simulation of Large Scale Analog Integrated Circuits*, PhD thesis, University of California, Berkeley, 1988.

[19] S.C. Fang, Y.P. Tsividis and O. Wing, "SWITCAP: A Switched Capacitor Network Analysis Program", *IEEE Circuits and Systems Magazine*, vol. 5, n. 3, September 1983.

[20] L.A. Williams III and B.A. Wooley, "MIDAS-A Functional Simulator for Mixed Digital and Analog Sampled Data Systems", in *Proc. IEEE International Symposium on Circuits & Systems*, May 1992.

[21] U. C. Berkeley, *The Almagest - Ptolemy 0.7 User's Manual*, 1997.

[22] P. E. Allen and E. R. Macaluso, "AIDE2: An Automated Analog IC Design System", in *Proc. IEEE CICC*, pp. 498–501, May 1985.

[23] M. Degrauwe et al., "IDAC: An Interactive Design Tool for Analog CMOS Circuits", *IEEE Journal of Solid State Circuits*, vol. SC-22, No.6, pp. 1106–1116, December 1987.

[24] J. Rijmenants et al., "ILAC: An Automated Layout Tool for Analog CMOS Circuits", in *Proc. IEEE CICC*, pp. 761–764, May 1988.

[25] H. Y. Koh, C. H. Séquin and P. R. Gray, "OPASYN: A Compiler for CMOS Operational Amplifiers", *IEEE Trans. on CAD*, vol. 9, n. 2, pp. 113–126, February 1990.

[26] G. Jusuf, P. R. Gray and A. Sangiovanni-Vincentelli, "CADICS - Cyclic Analog-To-Digital Converter Synthesis", in *Proc. IEEE/ACM International Conference on CAD*, pp. 286–289, November 1990.

[27] R. Neff, *A Module Generator for High Speed CMOS Current Output Digital/Analog Converters*, PhD thesis, University of California, Berkeley, 1995.

[28] C. Makris and C. Toumazou, "Qualitative Reasoning in Analog IC Design Automation", in *Proc. IEEE CICC*, pp. 831–834, May 1992.

[29] C. Toumazou and C. A. Makris, "Analog IC Design: Part I - Automated Circuit Generation: New Concepts and Methods", *IEEE Trans. on CAD*, pp. 218–238, February 1995.

[30] C. A. Makris and C. Toumazou, "Analog IC Design: Part II - Automated Circuit Correction by Qualitative Reasoning", *IEEE Trans. on CAD*, pp. 239–254, February 1995.

[31] E. Ochotta, L. R. Carley and R. A. Rutenbar, "Analog Circuit Synthesis for Large, Realistic Cells: Designing a Pipelined A/D Converter with ASTRX/OBLX", in *Proc. IEEE CICC*, pp. 365–368, May 1994.

[32] E. Ochotta, L. R. Carley and R. A. Rutenbar, "Synthesis of High-Performance Analog Circuits in ASTRX/OBLX", *IEEE Trans. on CAD*, vol. 15, n. 3, pp. 273–294, March 1996.

[33] G. Gielen, G. Debyser, P. Wambacq, K. Swings and W. Sansen, "Use of Symbolic Analysis in Analog Circuit Synthesis", in *Proc. IEEE Int. Symposium on Circuits and Systems*, pp. 2205–2208, May 1995.

[34] S. Donnay, K. Swings, G. Gielen and W. Sansen, "A Methodology for Analog High-Level Synthesis", in *Proc. IEEE CICC*, pp. 373–376, May 1994.

[35] J. Crols, S. Donnay, M. Steyaert and G. Gielen, "A High-Level Design Optimization Tool for Analog RF Receiver Front-Ends", in *Proc. IEEE/ACM International Conference on CAD*, pp. 550–553, November 1995.

[36] J. Rael, Rofougaran A. and A. Abidi, "A. Design Methodology Used in a Single-Chip CMOS 900 MHz Spread-Spectrum Wireless Transceiver", in *Proc. IEEE/ACM DAC*, pp. 44–49, 1998.

[37] Cadence Design Systems, *Signal Processing Workstation Datasheet*, 1998.

[38] L. Moult and J.E. Chen, "The K-model: RF IC Modelling for Communication Systems Simulation", in *IEE Colloquium Analog Signal Processing*, pp. 11/1–8, 1998.

[39] C. Teuscher, *Low Power Receiver Design for Portable RF Applications: Design and Implementation of an Adaptive Multiuser Detector for an Indoor, Wideband CDMA Application*, PhD thesis, University of California, Berkeley, 1998.

[40] U. Choudhury and A. Sangiovanni-Vincentelli, "Use of Performance Sensitivities in Routing of Analog Circuits", in *Proc. IEEE Int. Symposium on Circuits and Systems*, pp. 348–351, May 1990.

[41] E. Charbon, E. Malavasi and A. Sangiovanni-Vincentelli, "Generalized Constraint Generation for Analog Circuit Design", in *Proc. IEEE/ACM International Conference on CAD*, pp. 408–414, November 1993.

[42] E. Charbon, *Constraint-Driven Analysis and Synthesis of High-Performance Analog IC Layout*, PhD thesis, University of California, Berkeley, 1995.

[43] E. Charbon, E. Malavasi, U. Choudhury, A. Casotto and A. Sangiovanni-Vincentelli, "A Constraint-Driven Placement Methodology for Analog Integrated Circuits", in *Proc. IEEE CICC*, pp. 2821–2824, May 1992.

[44] E. Charbon, E. Malavasi, D. Pandini and A. Sangiovanni-Vincentelli, "Simultaneous Placement and Module Optimization of Analog IC's", in *Proc. IEEE/ACM DAC*, pp. 31–35, June 1994.

[45] E. Malavasi and A. Sangiovanni-Vincentelli, "Area Routing for Analog Layout", *IEEE Trans. on CAD*, vol. 12, n. 8, pp. 1186–1197, August 1993.

[46] E. Charbon, G. Holmlund, B. Donecker and A. Sangiovanni-Vincentelli, "A Performance-Driven Router for RF and Microwave Analog Circuit Design", in *Proc. IEEE CICC*, pp. 383–386, May 1995.

[47] E. Felt, E. Malavasi, E. Charbon, R. Totaro and A. Sangiovanni-Vincentelli, "Performance-Driven Compaction for Analog Integrated Circuits", in *Proc. IEEE CICC*, pp. 1731–1735, May 1993.

[48] P. Miliozzi, I. Vassiliou, E. Charbon, E. Malavasi and A. L. Sangiovanni-Vincentelli, "Use of Sensitivities and Generalized Substrate Models in Mixed-Signal IC Design", in *Proc. IEEE/ACM DAC*, pp. 227–232, June 1996.

[49] R. Gharpurey and R. G. Meyer, "Modeling and Analysis of Substrate Coupling in ICs", in *Proc. IEEE CICC*, pp. 125–128, May 1995.

[50] E. Liu, *Analog Behavioral Simulation and Modeling*, PhD thesis, University of California, Berkeley, 1993.

[51] E. Liu, G. Gielen, H. Chang and A. Sangiovanni-Vincentelli, "Behavioral Modeling and Simulation of Data Converters", in *Proc. IEEE Int. Symposium on Circuits and Systems*, pp. 2144–2147, May 1992.

[52] E. Liu and A. Sangiovanni-Vincentelli, "Behavioral Simulation for Noise in Mixed-Mode Sampled-Data Systems", in *Proc. IEEE/ACM International Conference on CAD*, pp. 322–326, November 1992.

[53] A. Demir, E. Liu, A. Sangiovanni-Vincentelli and I. Vassiliou, "Behavioral Simulation Techniques for Phase/Delay-Locked Systems", in *Proc. IEEE CICC*, pp. 453–456, May 1994.

[54] E. Felt, and A. Sangiovanni-Vincentelli, "Testing of Analog Systems Using Behavioral Models and Optimal Experimental Design Techniques", in *Proc. IEEE/ACM International Conference on CAD*, pp. 672–678, November 1994.

[55] E. Felt, S. Zanella, C. Guardiani and A. Sangiovanni-Vincentelli, "Hierarchical Statistical Characterization of Mixed-Signal Circuits Using Behavioral Modeling", in *Proc. IEEE/ACM International Conference on CAD*, pp. 374–380, November 1996.

[56] A. Demir, E. Liu and A. Sangiovanni-Vincentelli, "Time-Domain non-Monte Carlo Noise Simulation for Nonlinear Dynamic Circuits with Arbitrary Excitations", in *Proc. IEEE/ACM International Conference on CAD*, pp. 598–603, November 1994.

[57] Brooktree Corporation, *Brooktree Graphics and Imaging Product Databook*, 1993.

[58] G. Work, G. Talbot, A. Ferris, N. Henderson and P. McGuiness, "A 20 nS color lookup table for faster scan displays", in *Proc. IEEE International Solid-State Circuits Conference*, pp. 310–311, February 1985.

[59] K. Chi et al., "A CMOS triple 100-Mbit/s video DA converter with shift register and color map", *IEEE Journal of Solid State Circuits*, vol. SC-21, n. 6, pp. 989–996, December 1986.

[60] J. Leonard, N. Weste, L. Bodony, S. Harston and R. Meaney, "A 66-MHz DSP Augmented RAMDAC for Smooth-Shaded Graphic Applications", *IEEE Journal of Solid State Circuits*, vol. SC-26, n. 3, pp. 989–996, March 1991.

[61] D. Reynolds, "A 320MHz CMOS Triple 8b DAC with On-Chip PLL and Hardware Cursor", in *Proc. IEEE International Solid-State Circuits Conference*, pp. 50–51, February 1994.

[62] Chrontel, Inc., San Jose, CA, *CH8398 16-bit Interface ChronDAC$^{TM}$ True-Color RAMDAC + Dual Clocks*, 1994.

[63] R. Neff, P. Gray and A. Sangiovanni-Vincentelli, "A Module Generator for High Speed CMOS Current Output Digital/Analog Converters", in *Proc. IEEE CICC*, pp. 481–484, May 1995.

[64] E. M. Sentovich, K. J. Singh, H. Savoj C. Moon, R. K. Brayton and A. L. Sangiovanni-Vincentelli, "Sequential Circuit Design Using Synthesis and Optimization", in *Proc. Intl. Conf. on Computer Design*, pp. 328–333, October 1992.

[65] I. A. Young, J. K. Greason and K. L. Wong, "A PLL Clock Generator with 5 to 110 MHz of Lock Range for Microprocessors", *JSSC*, vol. 27, n. 11, pp. 1599–1607, November 1992.

[66] D. Jeong, G. Borrielo, D. Hodges and R. Katz, "Design of PLL-Based Clock Generation Circuits", *IEEE Journal of Solid State Circuits*, vol. SC-22, n. 2, pp. 255–261, February 1987.

[67] F. M. Gardner, *Phaselock Techniques, 2nd ed.*, J. Wiley & Sons, New York, 1979.

[68] F. M. Gardner, "Charge-Pump Phase-Lock Loops", *IEEE Trans. on Communications*, vol. COM-28, n. 11, pp. 1849–1858, November 1980.

[69] A. Demir, *Analysis and Simulation of Noise in Nonlinear Electronic Circuits and Systems*, PhD thesis, University of California, Berkeley, 1997.

[70] B. Kim, T. C. Weigandt and P. R. Gray, "PLL/DLL System Noise Analysis for Low Jitter Clock Synthesizer Design", in *Proc. IEEE Int. Symposium on Circuits and Systems*, May 1994.

[71] B. A. Murtagh and M. A. Saunders, "MINOS 5.1 User's Guide", Technical Report Rep. SOL 83-20R, Dept. of Operations Research, Stanford University, Stanford, CA, January 1987.

[72] R. Harjani, R. A. Rutenbar and L. R. Carley, "OASYS: A Framework for Analog Circuit Synthesis", *IEEE Trans. on CAD*, vol. 8, n. 12, pp. 1247–1266, December 1989.

[73] P. R. Gray and R. G. Meyer, "Future Directions in Silicon ICs for RF Personal Communications", in *Proc. IEEE CICC*, May 1995.

[74] B. Kim, D. Helman and P. R. Gray, "A 30-MHz Hybrid Analog/Digital Clock Recovery Circuit in 2-$\mu m$ CMOS", *IEEE Journal of Solid State Circuits*, vol. 25, n. 3, pp. 1385–1394, December 1993.

[75] T. C. Weigandt, B. Kim and P. R. Gray, "Analysis of Timing Jitter in CMOS Ring Oscillators", in *Proc. IEEE Int. Symposium on Circuits and Systems*, May 1994.

[76] D. Harrison, P. Moore, R. L. Spickelmier and A. R. Newton, "Data Management and Graphics Editing in the Berkeley Design Environment", in *Proc. IEEE/ACM International Conference on CAD*, pp. 24–27, November 1986.

[77] C. A. Laber, C. F. Rahim, S. F. Dreyer, G. T. Uehara, P. T. Kwok and P. R. Gray, "A Family of Differential NMOS Analog Circuits for a PCM Codec Filter Chip", *IEEE Journal of Solid State Circuits*, vol. SC-22, N.2, pp. 181–189, April 1987.

[78] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint Generation for Routing Analog Circuits", in *Proc. IEEE/ACM DAC*, pp. 561–566, June 1990.

[79] J. Burns, A. Casotto, M. Igusa, F. Marron, F. Romeo, A. Sangiovanni-Vincentelli, C. Sechen, H. Shin, G. Srinath and H. Yaghutiel, "MOSAICO: An integrated Macro-cell Layout System", in *VLSI '87, Vancouver, Canada*, August 1987.

[80] R. Neff, P. R. Gray and A. Sangiovanni-Vincentelli, "A Module Generator for High-Speed CMOS Current Output Digital/Analog Converters", *JSSC*, vol. 31, n. 3, pp. 448–451, March 1996.

[81] Stanley Brown, "Top-Down Redesign of a PLL for DECT", Eecs 219 class project, UCB, December 1998.

[82] I. Vassiliou, H. Chang, A. Demir, E. Charbon, P. Miliozzi and A. Sangiovanni-Vincentelli, "A Video Driver System Designed Using a Top-Down, Constraint-Driven Methodology", in *Proc. IEEE/ACM International Conference on CAD*, pp. 463–468, November 1996.

[83] T. H. Lee, *The Design of CMOS Radio-Frequency Integrated Circuits*, Cambridge University Press, 1998.

[84] D.D. Weiner and J.F. Spina, *Sinusoidal Analysis and Modeling of Weakly Nonlinear Cicuits*, Van Nostrand Reinhold Company, 1980.

[85] Avant! Corporation, *HSPICE User Manual*, 1997.

[86] Cadence Design Systems, *Spectre$^{TM}$ HDL Reference Manual, Version 4.4*, 1995.

[87] V. Volterra, *Theory of Functionals and of Integral and Integro-Differential Equations*, Dover Publications, Inc., New York, 1959.

[88] R. H. Flake, "Volterra Series Representation of Time-Varying Non-linear Systems", in *Proc. Second Int. Congress of IFAC on Automatic Control (Basel, Switzerland)*, pp. 91–99, 1963.

[89] J. Vlach and K. Singhal, *Computer Methods for Circuit Analysis and Design*, Van Nostrand-Reinhold, 1983.

[90] J.S. Roychowdhury, "SPICE3 Distortion Analysis", Master's thesis, University of California, Berkeley, 1989.

[91] S. A. Maas, "Two-tone Intermodulation in Diode Mixers", *IEEE Trans. on Microwave Theory and Techniques*, vol. MTT-35, pp. 307–314, March 1987.

[92] S. Maas, *Nonlinear Microwave Circuits*, Artech House, Norwood, MA., 1988.

[93] M. Schetzen, *The Volterra and Wiener Theories of Nonlinear Systems*, John Wiley & Sons, 1980.

[94] L.A. Zadeh, "Frequency Analysis of Variable Networks", *Proceedings of the I.R.E.*, March 1950.

[95] A. Papoulis, *Probability, random variables, and stochastic processes*, New York: McGraw-Hill, second Edition, 1984.

[96] W.A. Gardner, *Introduction to Random Processes with Applications to Signals & Systems*, McGraw-Hill, second Edition, 1990.

[97] K. S. Kundert, J. K. White and A. Sangiovanni-Vincentelli, *Steady-State Methods for Simulating Analog and Microwave Circuits*, Kluwer Academic Publ., Boston, MA, 1990.

[98] M. S. Nakhla and J. Vlach, "A Piecewise Harmonic Balance Technique for Determination of the Periodic Response of Nonlinear Systems", *IEEE Trans. on Circuits and Systems*, vol. CAS-23, pp. 85–91, February 1976.

[99] R. G. Hicks and P. J. Khan, "Numerical Analysis of Subharmonic Mixers Using Accurate and Approximate Models", *IEEE Trans. on Microwave Theory and Techniques*, vol. MTT-30, pp. 2113–2120, December 1982.

[100] Verbeyst F and V. Bossche, "Analysis of Intermodulation Noise in Frequency Converters by Volterra Series", *IEEE Trans. on Microwave Theory and Techniques*, pp. 2531–5, December 1994.

[101] S. O. Rice, "Volterra systems with more than one input port - Distortion in a frequency converter", *The Bell System Technical Journal*, vol. 52, n. 8, pp. 1255–1270, October 1973.

[102] R. B. Swerdlow, "Analysis of Intermodulation Noise in Frequency Converters by Volterra Series", *IEEE Trans. on Microwave Theory and Techniques*, pp. 305–313, April 1978.

[103] K.S. Kundert, "Sparse Matrix Techniques", in A.E. Ruehli, editor, *Circuit Analysis, Simulation and Design*, pp. 281–324, Elsevier Science Publishers B.V. (North-Holland), 1986.

[104] J. Rudell, JJ. Ou, T. Cho, G. Chien, F. Brianti, J. Weldon and P. Gray, "A 1.9GHz Wide-Band IF Double Conversion CMOS Integrated Receiver for Cordless Telephone Applications", in *Proc. IEEE International Solid-State Circuits Conference*, pp. 304–305, February 1997.

[105] J. Rudell, J. Weldon, JJ. Ou, L. Lin and P. Gray, "An Integrated GSM/DECT Receiver: Design Specifications", Memorandum UCB/ERL M97/82, University of California at Berkeley, November 1997.

[106] Iason Vassiliou and Alberto Sangiovanni-Vincentelli, "A Frequency-Domain, Volterra Series-Based Behavioral Simulation Tool for RF Systems, to appear at CICC, San Diego CA, May 1999".

[107] F. Balarin, M. Chiodo, P. Giusto, H. Hsieh, A. Jurecska, L. Lavagno, C. Passerone, A. Sangiovanni-Vincentelli, E. Sentovich, K. Suzuki and B. Tabbara, *Hardware-Software Co-Design of Embedded Systems: The POLIS Approach*, Kluwer Academic Publishers, Boston/Dordrecht/London, 1997.

[108] VSI Alliance$_{TM}$, *VSI Alliance$_{TM}$ Architecture Document - Version 1.0*, 1997.

[109] Cadence Design Systems, *BONeS Designer Datasheet*, 1997.

[110] Cadence Design Systems, *Alta$^{TM}$ VCC User Guide, Version 1.0 Beta*, September 1998.

[111] T. B. Cho, *Low-Power Low-Voltage Analog-to-Digital Conversion Techniques Using Pipelined Architectures*, PhD thesis, University of California, Berkeley, 1995.

[112] A. R. Feldman, *High-Speed, Low-Power Sigma-Delta Modulators for RF Baseband Channel Applications*, PhD thesis, University of California, Berkeley, 1997.

[113] T.B. Cho, Chien G., F. Brianti and P.R Gray, "A power-optimized CMOS baseband channel filter and ADC for cordless applications", in *Proc. 1996 Symposium on VLSI Circuits8th Design Automation Workshop*, pp. 64–5, 1996.

[114] J. Ou, "Lecture Notes EECS 247
University of California Berkeley 1996".

[115] J. C. Rudell, "Qualifying Examination Slides
Univ. of California Berkeley 1996".

[116] A. Demir and A. Sangiovanni-Vincentelli, "Simulation and Modeling of Phase Noise in Open-Loop Oscillators", in *Proc. IEEE Custom Integrated Circuits Conference*, May 1996.