MODEL VALIDATION OF MIXED-SIGNAL SYSTEMS

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Byong Chan Lim
December 2012

© 2012 by Byong Chan Lim. All Rights Reserved.
Re-distributed by Stanford University under license with the author.

This work is licensed under a Creative Commons Attribution-Noncommercial 3.0 United States License.
http://creativecommons.org/licenses/by-nc/3.0/us/

This dissertation is online at: http://purl.stanford.edu/xq068rv3398

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Mark Horowitz, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Boris Murmann, Co-Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Bruce Wooley

Approved for the Stanford University Committee on Graduate Studies.

Patricia J. Gumport, Vice Provost Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Today it is difficult to validate a mixed-signal System-on-Chip, i.e., one which contains analog and digital components. The problem is that the analog and digital subsystems are usually strongly intertwined so they must be validated together as a system, but the validation approach for analog and digital blocks are completely different. We address this problem by creating high-level functional models of the analog components that are compatible with top-level digital system validation, and then providing a method of formal checking to ensure that these functional models match the operation of the transistor level implementations of these blocks.

The formal checking of the functional analog models is enabled by observing that the result surface of an analog block is a smooth function of its analog inputs – that is what makes it an analog block. This smooth result surface means it is not difficult to "explore" the design space of an analog block. We use this insight to create an equivalence checker between two analog descriptions: a SPICE netlist and its Verilog model. This checker exploits the fact that most analog circuits have a linear intent.

Lastly we use this same insight to simplify statistical analysis of large mixed-signal systems to ensure that the system is robust to process variations. We describe a way to characterize the statistical behavior of circuits and to map the results to the functional model so one can estimate the parametric yield of the system by running system-level Monte Carlo simulations with analog models instead of circuit netlists.

Acknowledgements

Looking back on the days at Stanford, I have been so lucky to get help from many people. Without their help, this work would not have been possible. First and foremost, I would like to thank my lovely wife, Sangmee Kim, for her endless love, support, encouragement, and prayers.

I am fortunate to have had a dedicated advisor, Professor Mark Horowitz. Not too much to say that he is a great mentor, researcher, and teacher, he has spared no effort for me to focus on the research in various ways. I cannot express the depth of my gratitude to him in words. Besides my advisor, I would like to thank Professor Boris Murmann and Professor Bruce Wooley for their valuable knowledge, insightful comments and questions. I am also grateful to Professor Yoshio Nishi who served on my orals committee, and to my undergraduate and master degree advisor, Professor Oh-Kyong Kwon, for continual encouragement and support.

I would like to acknowledge the guidance and expertise of Professor Jaeha Kim who was like another advisor to me. Without his guidance, this thesis would not have seen the light. I am grateful to everyone in VLSI research group, former and present students of Mark — James Mao, Metha Jeeradit, and many others — for their friendship and helpful discussions. Especially, I would like to thank James Mao for his help and criticism on my thesis work. I would also like to thank wonderful administrative assistants, Teresa Lynn and Mary Jane Swenson, for always making things run smoothly.

Much of the research would not have been possible without the support of former and current supporters of the Stanford Rethinking Analog Design initiative — TI, ADI, Intel, Maxim, Mentor Graphics, Xilinx, TSMC, Toshiba, Rambus, LSI, and

NVIDIA.

Last but not least, I would like to thank my family, my parents and brothers, for their unconditional love, support, and patience.

Contents

List of Tables

List of Figures

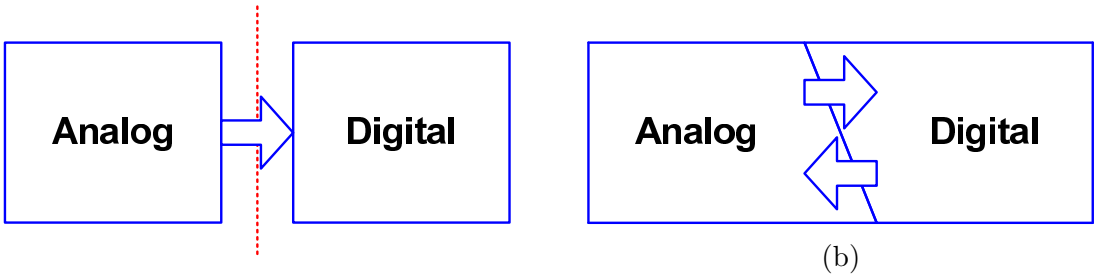
Chapter 1

Introduction

Most of today's VLSI systems are mixed-signal in nature; they have both analog and digital components. Microprocessors, for example, have multiple phase-locked loops for clock generation and high-speed serial interfaces for chip-to-chip communication [2–6]. To interface with the physical world, RF and data conversion circuits are integrated in various communication ICs [7–12]. The complexity of these mixed-signal systems is increasing as technology scales down, and thus the complete, system-level verification of the systems is becoming more important since any small functional bug can cause a re-spin which is time consuming and expensive. However, this mixed signal validation is difficult because validation methods and representations of analog/digital subsystems are not compatible. Separate validation of the subsystems is also not possible because they are strongly intertwined.

1.1 Challenges of Mixed-Signal System Validation

As technology scales, power supply voltages also scale down while the circuit performance requirements become more challenging. Moreover, many short-channel effects of nanometer CMOS devices, e.g., DIBL, velocity saturation, surface scattering, have a negative impact on the analog circuit performance which can counter balance the improved device cut-off frequency [13]. To mitigate these issues, analog designers have leveraged digital circuits by simplifying the analog circuit and then compensating for



(a) (b)

Figure 1.1: Mixed-signal design: (a) Analog/Digital system and (b) Mixed-signal system [1].

its non-ideal characteristics in digital domain. This strategy typically forms feedback loops between analog and digital blocks [10, 14].

Validating mixed-signal systems gets harder as these analog/digital interactions increase. If the interface between them is loosely coupled and the signal flow is unidirectional as depicted in Figure 1.1a, the verification would be easy. One only needs to check each subsystem separately and ensure that the interface is correct by running co-simulation for a short period of time. Unfortunately, when two subsystems are tightly coupled as shown in Figure 1.1b, it is necessary to check if the feedback loops function correctly and are stable over the environmental variability, e.g., process, temperature, and supply voltage. Therefore, the two subsystems should be validated together as a single system.

Mixed-signal validation is time consuming mostly because of the different levels of abstractions between analog and digital subsystems. A common practice for validation is to run co-simulation, e.g., a hardware description language (HDL) simulator for a digital system and a circuit simulator for an analog system. In this case, simulating analog components limits the simulation speed. A circuit simulator executes many numerical iterations on each time step to maintain the simulation accuracy, which makes the simulation slow. On the other hand, the simulator for digital systems is event driven, i.e., no time integration, making its simulation speed much faster.

Moreover, long test vectors are needed to simulate the digital subsystems for the system to settle because the overall system generally requires a boot-up sequence to calibrate and configure the system. In addition, the number of test vectors to run

the system validation is usually very large since it should check all the functional modes created by digital systems. This can even be the case for a small, analog-intensive system because digital feedback loops have much lower bandwidth than signal bandwidth such that it takes a long time for the digital loop to settle. As a result, the length of the required simulation time can grow by orders of magnitude. Compounding the simulation speed issue is the need to run regression tests. Designers should validate the system functional modes whenever any part of a design is changed. Thus simulating analog systems is often the limiting factor in speeding up the overall system validation even though analog components only occupy a small portion of a system.

To speed up mixed-signal system validation, avoiding time integration in simulating analog components is necessary. This means models should be functional; they need to describe the overall function of the circuit and not the connection of the individual devices. Once this transformation is done, one possible way to enable fast system simulation is to use analog functional models written in an event-driven HDL, e.g., (System)Verilog.[1]

Since a digital system is designed with a Verilog HDL model, there are strong benefits to using an event-driven HDL, especially Verilog HDL, for analog model creation. In general, digital designers control chip validation because this validation is mostly for checking digital system functions. By writing analog models in HDLs, it is possible to seamlessly fit the analog subsystem into the verification flow of the digital subsystem. SystemVerilog also contains features to make the generation of validation tests easier. One is able to leverage many features, e.g., class, randomization, assertion, and coverage, of SystemVerilog test suites [15].

1.2 Mixed-Signal Design Errors

Creating an analog functional model does not completely solve the validation problem. Unfortunately, the model is perceived very differently by analog and digital designers. In modern digital system design, functional behavior is first written in a HDL such

[1]For the rest of this thesis, the term *Verilog* includes SystemVerilog if it is not explicitly addressed.

as Verilog or VHDL; this serves as the specification of the system. This model is then expanded and debugged, and a set of regression tests is used to ensure new errors are not introduced into the design. During this process, the HDL is automatically mapped to digital standard cells for the targeted technology, and then placed and routed to physical layout as well as gate-level netlist [16]. With the generated gate-level netlist, other electrical rule checks such as timing analysis and power analysis are also performed without relying on (fast-)circuit simulators. Since these physical designs are derived from and are validated against the functional model, digital designers trust the model.

However, for analog designers, the model is often viewed as an approximation of the actual circuit. They verify the circuits, draw layouts, and then run another circuit verification with parasitic extracted netlists. While they may write analog models for system validation, the circuits are neither generated from nor validated against the models. Thus the traditional design flow does not force designers to write an accurate analog model. The models are often provided without checking if their functional behavior matches that of the corresponding circuit implementations, which leads to mixed-signal design errors.

As mentioned previously, digital designers control system validation and they trust analog models because they believe the model is the specification. Thus, although the chip validation with the models has passed, the real chip could fail because of inconsistencies between analog circuits and their models. The cause of these errors is usually not a subtle analog issue such as nonlinearity and noise — these are found through circuit simulations. Rather, the problems are often trivial wiring mistakes: inconsistencies between circuits and models at the I/O boundary, which include missing connections, mislabeled pins, signal inversion, bus-order reversal, and bus-encoding mismatch. For example, the polarity of a signal might be inverted, e.g., active low vs. active high for the reset signal, and a bus might be connected via different encoding styles, e.g., big-endian vs. little-endian. Worse yet, these errors are often repeated, which is extremely wasteful.

These types of errors between a model and its circuit do not occur for digital systems since the implemented digital standard cells are simulated to ensure that they

match their functional models. We need a similar functional equivalence checking between an analog circuit and its model to ensure the analog model matches the circuit. The digital tools check the boolean function extracted from both the circuit and functional model; the next chapter explains how a linear model serves the same function for analog circuits, and uses this model to create equivalence checker.

1.3 Statistical Analysis of Mixed-Signal Systems

Another validation issue for mixed-signal systems is how to account for the effects of process variations, both die-to-die and within-die variation, on circuit performance. Ensuring the system is robust to the expected variation of process parameters is becoming more important since the process variations are becoming larger as devices scale down in size.

Currently, using circuit simulators to run process corner and Monte Carlo simulations is common practice for statistical analysis, but it is computationally expensive since a large number of samples are needed and the circuit simulation is slow. This becomes worse as the circuit size and the number of variation parameters increase. Moreover, while this can be done in most linear/nonlinear analog circuits, coupling them to complex digital systems makes the problem much harder because of the different levels of abstractions between the analog/digital subsystems as mentioned in Section 1.1.

The situation becomes even worse when a new process technology node is developed in parallel with circuit design. Tuning process parameters for better yield involves many iterations between process engineers and circuit designers. Thus it might be very expensive if each iteration is long, which is the case if one wants to run Monte Carlo simulations with circuit netlists. Of course these simulations could be run in parallel, but that is also expensive because of the cost of simulator licenses.

An alternative is to run Monte Carlo simulations with functional models instead of circuit netlists if the models are accurate enough. With functional models, performing Monte Carlo simulations at the system level runs much faster and since these are end-to-end tests, the associated test vectors and measurement scripts already exist. As

we will show in Chapter 5, one gets an additional performance boost since there are fewer parameters to vary in this case.

1.4 Organization

This thesis describes two key contributions on mixed-signal model validation: functional equivalence checking between an analog circuit and its model and a statistical simulation framework for large mixed-signal systems.

Chapter 2 provides our framework for creating and analyzing analog models. "Analog" means that the response surface is smooth, which enables the linearization of the surface for simpler analysis. The key challenge is to find the domain where the response is smooth. We also show how this framework can be extended to tunable circuits by decomposition into coupled linear systems.

Equivalence checking between two analog descriptions is described in Chapter 3. We explain how we can exploit the linear intent of analog circuits for functional equivalence checking and analog test vector generation. We first explore how to check the functional equivalence between different analog models and how to generate analog test vectors. Then, we describe the implementation of our equivalence checker. Chapter 4 uses the analog cells in a 40-way, time-interleaved Analog-to-Digital Converter (ADC) as examples of model equivalence checking.

Chapter 5 presents the statistical analysis of a large mixed-signal system to ensure that the system is robust to process variation. We first investigate ways to abstract the effect of the variation on circuit performance. From this abstraction, we create process variation models of mixed-signal circuits, and provide a methodology to estimate the parametric yield by running system-level Monte Carlo simulations with analog functional models instead of circuit netlists. A Phase-Locked Loop (PLL) example is shown to demonstrate the analysis methods.

Chapter 2

Basis for Model Validation

Over the past four decades, digital design methodology has changed dramatically. Rather than designing a digital system at circuit level, the system is described by a technology independent HDL model which is synthesized into a real implementation using a set of cells for standard digital functions. The function of the system is then verified by simulating the HDL model with functional simulators and many other rule checkers such as static timing analysis and power analysis tools are available for the chip-level verification.

A boolean abstraction for signal values and the synchronous nature of digital systems, which provides a framework for a mapping between an abstract functional model and its physical implementation, have enabled many of these tools. A similar kind of abstraction is necessary for analog model verification. This chapter proposes using a *linear abstraction* of analog circuits to serve the equivalent function for the circuits. We begin by briefly reviewing the previous works on analog model validation and then describe the background of our approach.

2.1 Inconsistency between Analog Models

As mentioned in Chapter 1, any mismatch between a circuit and a model may cause design errors and lead to a re-spin of the design. In many cases, this functional discrepancy occurs at the I/O boundary, not in the detail function [17].

A common mistake is polarity inversion of a digital control input since such an input can be either active high or low. For example, the model enters power down mode when the signal is asserted with a logic high signal but the circuit uses active high to activate in normal operating mode. Another mismatch in communicating digital signals between digital and analog blocks is reversal of bit ordering, i.e., big endian vs. little endian. For instance, a digital block transmits a signal DATA[7:0] while the receiving analog block assumes DATA[0:7]. It is also possible to miscommunicate encoding protocol of digital buses, e.g., binary code vs. thermometer code, when digitally controlling analog properties. Depending on the encoding protocol, the range of adjustable analog properties may vary such that the actual controllability of a circuit is not wide enough to meet the specifications.

Polarity inversion can occur in communicating analog signals as well. Differential inputs can be swapped, and the polarity of a current source/sink can be inverted. It is also possible that the sensitivity of analog response to an analog input is in the opposite direction between two analog models. The analog type of a signal such as voltage, current, and phase can also be confused. An analog signal in a model is usually represented as a real number, which means that the intended type of the signal is not checked. Thus, unlike digital signals which only have the "logic" type, an analog block could send a voltage signal while its receiving block took a current signal.

Another issue is that an analog functional model is often a simplified version of the circuit, and may not be pin accurate to its circuit netlist. This can lead to another class of errors since it is then hard to generate a correct, verified system-level netlist to run through layout versus schematic (LVS) at the transistor level.

These errors are trivial and obvious mistakes, and many are easily identified through manual checks. However, these errors still occur in practice since the documentation on the interface is often not updated and the analog model is not checked against its circuit implementation. Frequently, at least one of these bugs escapes detection in a complex system with an iterative design and verification process. Therefore, a tool for detecting these errors would greatly improve analog validation.

2.2 Previous Work on Analog Model Validation

Validating a model always raises a question about coverage. That is, how many input patterns should be tried to explore all states in a system. In digital logic, boolean abstraction answers this question easily as it assumes that each input is either '1' or '0'. The number of possible input patterns is finite and one can completely explore its outputs across all possible input patterns, at least for small number of inputs; this is how digital standard cells are verified. On the other hand, the answer has remained unclear for analog circuits since analog signals take on continuous values.

Recent efforts have taken different approaches to address this coverage question for analog circuits, yielding different types of checking tools. One approach is hybrid system verification [18–21]. The transfer function of a continuous-time analog circuit is transformed into discrete z-domain by discretizing voltage and time. The states of this new system form a finite grid and the circuit can be represented as a finite-state machine (FSM). One of the established digital verification methods and its coverage metric are then applied to the FSM. While this method enables formal checking of analog circuits, the number of states to be explored is often very large even for a small circuit, making this approach infeasible for large-scale circuits. In particular, it is difficult to map strongly nonlinear analog circuits in voltage domain (e.g., a phase locked loop and a class-D amplifier) to the framework.

Another approach is a top-down verification methodology, where a circuit component is validated in the system-level context using the same simulation setup for the model and circuit [22]. One of the important issues when using a higher-level model is validating how well it reflects the behavior of a real circuit. In the top-down verification methodology, one creates a number of testbenches and their test vectors, which leverage the transistor-level simulation tests, to compare the results of different models of the circuit to check fidelity. This method was created mostly to check Verilog-AMS models with their circuit implementations; checking fully functional models adds additional issues.

The first issue is that this methodology relies on the existing circuit test vectors. These are created manually, with no guarantee that these vectors will be complete to

ensure that the implementation matches the model. For example, the tests created to measure the transistor level performance of the circuit might not test its operation for all possible power modes or configurations. The second issue is that the correspondence between the real circuit and the Verilog model is often weak; the behavioral model may use representations of analog signals such as phase, frequency, or charge as well as voltage and current. Since real circuits can only represent signals in terms of voltage and current, some signal domain translation is necessary to compare the two models.

To address these limitations, we need to create a higher level of circuit abstraction for analog circuit behavior. With such an abstraction, it is possible to create a spanning set of test vectors and formally validate a design, enabling functional equivalence checking between the circuit representations at the lower, different levels of abstractions. In digital design, every digital standard cell is verified against its logic function using boolean abstraction. With the boolean abstraction, all inputs, states, and outputs only have two values, '1' and '0'. Thus the boolean output of a digital standard cell is compared to its logic function for all possible combinations of boolean inputs. The abstraction of analog circuits used to compare the function of two analog descriptions is discussed in the next section.

2.3 Formal Analog Model

"Analog" means that response surface is smooth. Since the smoothest function is linear, we generally analyze analog response through linearization at an operating point. This linearization is a powerful tool to simplify the analysis of analog response, and we will use it as the base of our abstraction for analog circuits.

Linear abstraction means that we map the behavior of all analog circuits to a *linear (dynamic) system* with corrections [1, 23–26]. The response of a linear system to its inputs is completely specified by the system's transfer function. Therefore, a transfer function can serve as a formal model of a linear circuit. In a matrix form, the system model is given by

$$\mathbf{Y} = \mathbf{A}\mathbf{x} \tag{2.1}$$

where $\mathbf{Y}$ is the system output vector, $\mathbf{x}$ is the system input vector, and $\mathbf{A}$ is called either *gain matrix* if the system is frequency independent, or *transfer matrix* if the system is frequency dependent. Particularly, the dynamical behavior of the system can be described by a linear filter model such as the time-domain impulse response or frequency-domain transfer function, which is often analyzed by small-signal AC analysis in circuit simulators.

This abstraction seems plausible since most characteristics of analog circuits that we care about are either the properties of a linear system such as gain and bandwidth, or the quantities that describe the deviation of the circuit from the linear model such as offset and distortion. In addition, most analog circuit specifications require that nonlinear properties should be minimized. For example, one wants to minimize signal distortion and input offset of an amplifier, the nonlinearity of a data converter (e.g., integral nonlinearity and differential nonlinearity), and signal distortion of a RF mixer when shifting the frequency of a carrier signal with a local oscillator.

There are many possible problems with using a linear abstraction for analog circuits. The first, and most obvious, issue is that some analog circuits are grossly nonlinear. These circuits, e.g., mixers, phase-locked loops (PLL), and delay-locked loops (DLL), have transfer functions that cannot be approximated by a linear or even slightly nonlinear function. We deal with these circuits by noticing that the intent of the circuits is still smooth and nearly linear, but in a domain different from voltage and time. For example, the role of a mixer is to shift and combine different frequency spectrum, which can be described by a linear operator in frequency space. A PLL may have square wave inputs and outputs, but the relationship between the phase of the input clock and the phase of the output clock is smooth and nearly linear. The use of domain translators to pull out the linear intent of circuits is discussed in more detail in Section 2.3.1. The second issue is that many analog circuits are configurable. For instance, the gain or bandwidth of an amplifier can be controlled by some control inputs. These circuits cannot be represented by a linear model since superposition does not hold; the cross product of inputs also significantly contributes to the output. To fit these circuits to our linear system framework, we introduce the concept of a coupled linear system by pointing out that the controlled property is generally slowly

time varying not to disturb the signal path of the system that is carrying information. This is discussed further in Section 2.3.2.

Linearization of analog responses does not prevent designers from creating models with nonlinear effects. While all analog circuits show some nonlinearity, we rely on the fact that the intent of the designer is for the circuit to be a linear system,[1] so the realistic behavior of the circuit at worst can be well captured by a weakly nonlinear system model [27–33], without having to resort to a more general, but poorly understood, strongly nonlinear system model [34–37]. Given that the circuit response is weakly nonlinear, possibly in a different variable domain, the residual error after linearization is small. Therefore, the error can be fitted by a linear summation of a few terms in a functional series expansion, i.e., a generalized linear model [38]. For example, the system response to inputs can be modeled by either Taylor series without memory effect, or Volterra series to capture memory effect [39, 40]. This approximation exactly coincides with how analog specifications are made. Often, the distortion due to nonlinearity of a circuit is analyzed with a few orders of harmonic distortion since most of the distortion is concentrated on the first few harmonics.

2.3.1 Domain Translation

When viewing voltage or current vs. time, many analog circuits are strongly nonlinear — mixers and ring-oscillators are easy examples.[2] For these nonlinear circuits, however, it is often easier to understand the circuit function in another domain where the circuits become linear such as phase and duty-cycle domain.

Through variable domain translation, the linear abstraction can be applied to circuits which appear strongly nonlinear in electrical domains. In fact, almost all of these circuits exhibit linear system behavior in some transformed variable domains [23]. A representative example is a PLL which consists of a set of digital blocks: phase detector, ring oscillator, clock buffers, and frequency divider. It takes a clock signal as input and produces another clock, possibly with a different frequency, as

[1] The circuit is linear or weakly nonlinear over a certain range of inputs. We deal with the case when the input range is too wide in greater detail in Section 3.2.3.

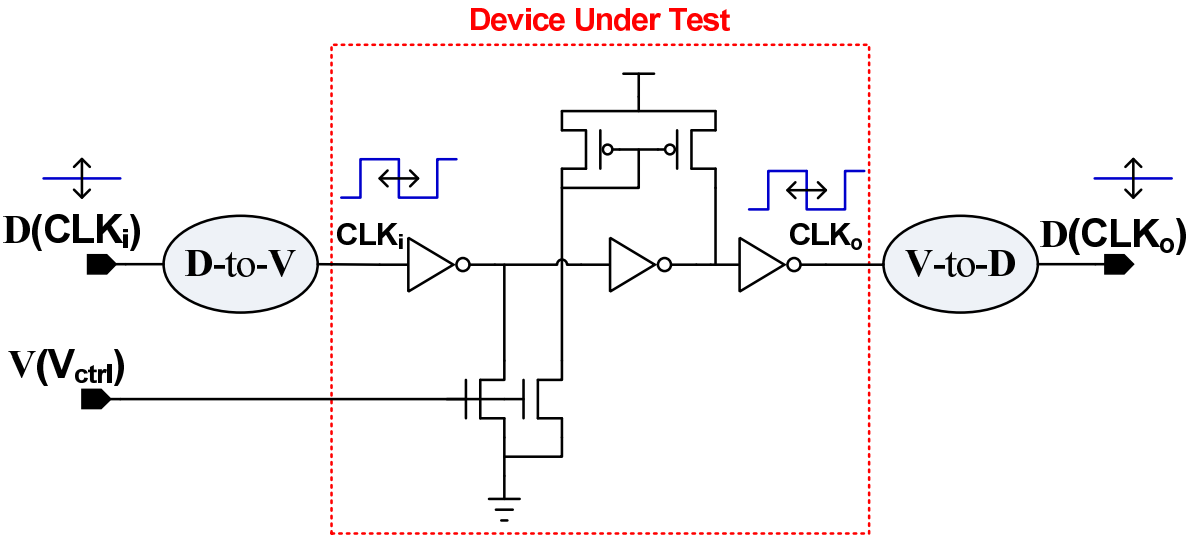
[2] Domains such as voltage, current, and time are referred to electrical domains in the thesis.

output, which shows strongly nonlinear behavior in the voltage domain. As its name implies, a PLL locks the output clock phase to the input clock phase. In the phase domain of the input and output clocks, the PLL is fairly linear when it is operating around its locking condition. Moreover, the frequency-domain transfer function of a PLL from the input phase to the output phase is the key characteristic of interest, which shows various properties of a PLL as a linear dynamic system, including loop bandwidth, damping factor, poles, and zeros.

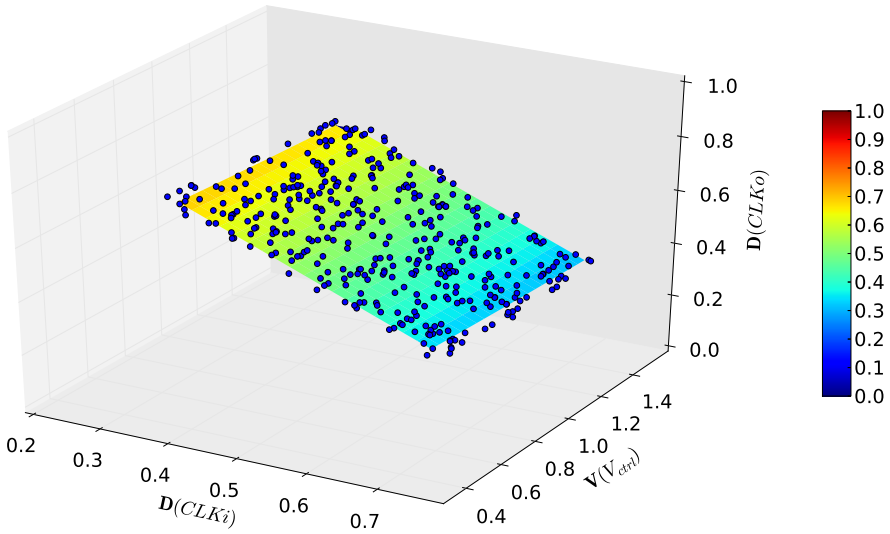
The transfer function of these strongly nonlinear circuits can be characterized by an extended AC analysis with variable domain translators [23]. In most RF circuit simulators, periodic AC (PAC) analysis is available to simulate the linear, periodically time varying (LPTV) system response of RF circuits such as mixers and oscillators, given that the circuit has periodic steady state (PSS) [41]. For instance, the linear system model of a mixer is found by analyzing the signal spectrum around the frequency shifted by the local oscillator clock frequency. This AC analysis technique can be extended to strongly nonlinear circuits such as PLL and DLL with variable domain translators since the resulting response surface can be smooth in the transformed domain and thus PAC analysis can be performed in that variable domain. For example, the transfer function of a PLL circuit is characterized with this extended PAC analysis in the phase domain [23]. In a similar way, one can perform the analysis of a DLL in the delay domain where its linear intent is exposed.

For various calibration or adaptation loops, the right domain is usually the quantity being calibrated, e.g., gain for the adaptive gain loop, duty cycle for duty-cycle correcting loop. The function of a circuit can be easily mapped to a linear system model when one finds an appropriate variable domain where the circuit response is smooth. For example, a duty-cycle adjuster and its response are shown in Figure 2.1. The circuit adjusts the duty cycle of the output clock $\mathrm{CLK_o}$ from the incoming clock $\mathrm{CLK_i}$ by controlling the voltage input $\mathrm{V_{ctrl}}$. While clearly nonlinear in voltage, the response surface is nearly hyperplane in duty-cycle domain as shown in Figure 2.1b through the domain translation with $\mathbf{D}$-to-$\mathbf{V}$ and $\mathbf{V}$-to-$\mathbf{D}$ translators.[3] Thus its

[3] $\mathbf{D}(\cdot)$ means a signal in duty-cycle domain, and $\mathbf{V}(\cdot)$ means a signal in voltage domain. SystemVerilog code examples of $\mathbf{D}$-to-$\mathbf{V}$ and $\mathbf{V}$-to-$\mathbf{D}$ translators are listed in Appendix A.



(a)



(b)

Figure 2.1: Duty-cycle adjuster: (a) circuit diagram with domain translators and (b) response surface in duty-cycle domain.

function is simply formulated as the following linear equation

$$\mathbf{D}(\mathrm{CLK_o}) = \alpha \cdot \mathbf{D}(\mathrm{CLK_i}) + \beta \cdot \mathbf{V}(\mathrm{V_{ctrl}}) + \gamma \tag{2.2}$$

where γ is the offset of the system, α and β are partial gains from $\mathbf{D}(\mathrm{CLK_i})$ and $\mathbf{V}(\mathrm{V_{ctrl}})$ inputs, respectively. This simplification greatly reduces the complexity of the circuit verification; one simply samples three responses to extract α, β, and γ.

Note that these domain translators should be used carefully in model validation. To avoid errors in domain translators causing chips to fail, all translators should contain inverse functions as well, e.g., a phase($\mathbf{\Phi}$) to voltage($\mathbf{V}$) translator would have a $\mathbf{V}$-to-$\mathbf{\Phi}$ translator. Prior to model validation, these translation functions are validated by assuring that the concatenation of these two functions is the identify function. Once the translators are checked, it is highly improbable that a translator exactly cancels errors in a functional model.

2.3.2 Coupled Linear System Model

There is another kind of circuit which appears strongly nonlinear: a tunable analog circuit. In many designs, some properties of a circuit are tunable to optimize the circuit performance over the environmental variation such as process, voltage, and temperature. The parameters are either adjusted by digitally stepping through quantized values or adjusted continuously with control inputs. For example, a variable gain amplifier (VGA) shown in Figure 2.2 is a voltage amplifier, where its voltage gain α (the controlled parameter) is adjusted by the analog bias current input $\mathrm{I_{BIAS}}$. This kind of circuit is strongly nonlinear at first glance. The gain control in the VGA means that the output is related to the product of $\mathrm{I_{BIAS}}$ and $\mathrm{V_{IN}}$ since the gain is also varying.

Most designers describe and reason about the circuit as the cascade of two *coupled linear systems*; one modifies a system parameter of the second. Generally speaking, tunable parameters of a circuit are controlled either statically, or varying over time with some control feedback loops. If the control is static, the circuit can be easily decoupled into coupled linear systems: a system along the path carrying information

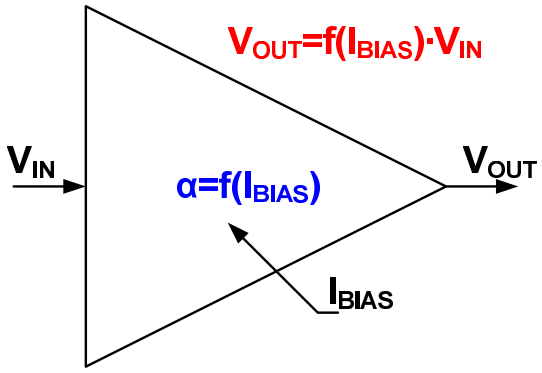
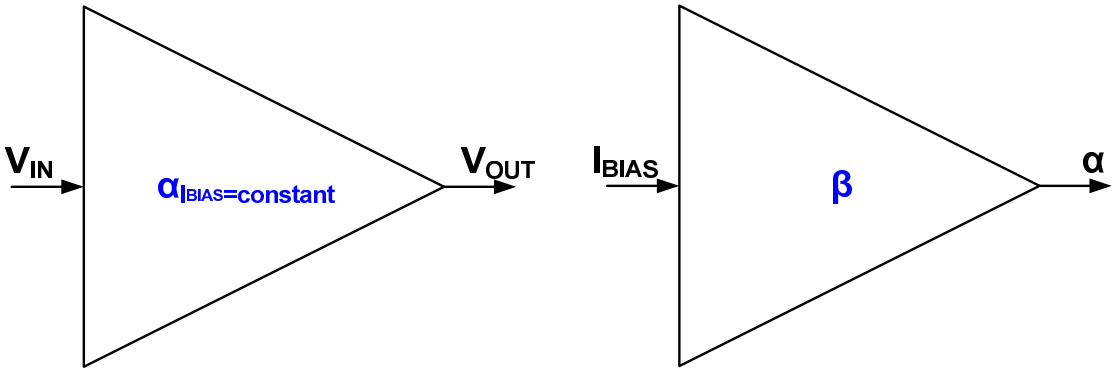


Figure 2.2: Block diagram of a variable gain amplifier.



(a) (b)

Figure 2.3: Coupled linear systems of the VGA in Figure 2.2: (a) linear system from V_{IN} to V_{OUT} and (b) linear system from I_{BIAS} to α of the system (a).

and the other system from a control input to a controlled parameter. Even if the circuit parameter is adjusted by a control feedback loop, the update rate of the parameter is generally much slower than the information signal bandwidth in most systems. Otherwise, the loop becomes unstable. This is because the feedback loop needs to measure the average of the system response before updating the parameters. This means that validating these systems as two coupled systems should not introduce any holes for errors to slip in.

Going back to the VGA example in Figure 2.2, the nonlinear circuit can be decoupled into two coupled linear systems as shown in Figure 2.3: the system along the path from the voltage input V_{IN} to the voltage output V_{OUT} which is carrying

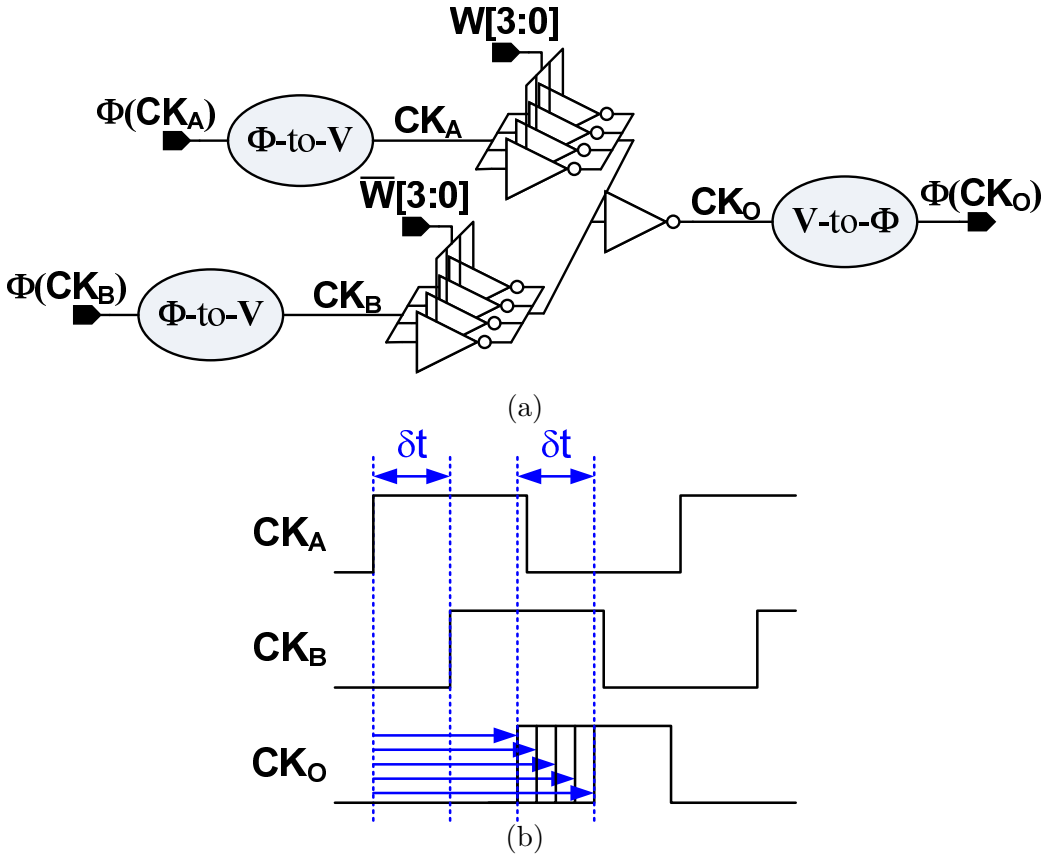


Figure 2.4: Phase interpolator: (a) circuit diagram and (b) timing diagram.

information and the other system from the bias current control input I_{BIAS} to the controlled parameter of the first system, α.

Another example shown in Figure 2.4 is a digital phase interpolator. It takes two incoming clocks, CK_A and CK_B, and produces the phase-interpolated clock CK_O. The amount of phase interpolation is adjusted digitally with the digital input W. Its function in phase domain, or equivalently in delay domain, is described as two simple linear equations, which are given by

$$\mathbf{\Phi}(\mathrm{CK_O}) = G \cdot (\mathbf{\Phi}(\mathrm{CK_B}) - \mathbf{\Phi}(\mathrm{CK_A})) + \phi_0 + \mathbf{\Phi}(\mathrm{CK_A}) \tag{2.3}$$

$$G = 1 - \sum_{k=0}^{3} \beta_k \cdot \mathrm{W_k} \tag{2.4}$$

Listing 2.1: Verilog model example for a variable gain amplifier

```
1  // Variable Amplifier with resistive load
2  // bias current is tunable
3  module VGA (
4    output real outp, outn,
5    input real inp, inn, IBIAS);
6
7  parameter real vdd = 1.8; // supply voltage
8  parameter real R_load=1e3; // resisitve load
9  parameter real vin_os = 0.0; // input dc offset voltage
10
11 real VIN; // input differential voltage
12 real VOUT;  // output differential voltage
13 real VOUT_CM;  // output common-mode voltage
14 real gain;  // variable gain of the amplifier
15
16 // VGA functional description
17 assign VIN = inp - inn;
18 assign gain = IBIAS*R_load; // Coupled Linear System 2
19 assign VOUT = gain*tanh(VIN-vin_os);  // Coupled Linear System 1
20 assign VOUT_CM = vdd - gain/2.0;
21 assign outp = VOUT_CM + VOUT/2.0;
22 assign outn = VOUT_CM - VOUT/2.0;
23
24 endmodule
```

where ϕ_0 is the phase (delay) offset of the system, G is the amount of phase interpolation, and $\mathrm{W_k}$ is k-th bit of digital control input W. From the equations, there exist two coupled linear systems: the linear system from $\mathbf{\Phi}(\mathrm{CK_A})$ and $\mathbf{\Phi}(\mathrm{CK_B})$ to $\mathbf{\Phi}(\mathrm{CK_O})$ with a static G by pinning W to a constant value and the other system from W to G. Once the system is decoupled, the formula for the second system remains unchanged while the first system is formulated as follows.

$$\mathbf{\Phi}(\mathrm{CK_O}) = G_{\mathrm{W=constant}} \cdot (\mathbf{\Phi}(\mathrm{CK_B}) - \mathbf{\Phi}(\mathrm{CK_A})) + \phi_0 + \mathbf{\Phi}(\mathrm{CK_A}) \tag{2.5}$$

It is noteworthy that decoupling into coupled linear systems is exactly how we write a model of a tunable circuit. For instance, the functional Verilog model of the VGA example can be written as shown in Listing 2.1. It shows that the second system, gain vs. $\mathrm{I_{BIAS}}$, is separately described first in line 18, and coupled to the first

system in line 19.

In summary, we have shown a way to expose the underlying linear intent of tunable analog circuits. It turns out that the tuned parameters of a main linear system form additional linear systems with their controlling inputs, which are loosely coupled to the main linear system. We describe these multiple linear systems as coupled linear systems, and show that linear abstraction still holds for these circuits. One remaining question is how to explicitly expose the controlled parameters to validate the models since the parameters are not the direct outputs of the circuits. The answer to this question is explained in Section 3.3 when defining the port intent in our linear system context.

2.4 Summary

In this chapter, we showed that a linear model captures the designer's intent of analog circuits and can serve as a formal model. The linear system model can be applied to many kinds of circuits by transforming the variable domain of I/Os, and describing a circuit as coupled linear systems. Also, the linear system model can be easily extended to build an accurate analog model by describing the circuit as a weakly nonlinear system.

In the next chapter, we discuss how functional equivalence checking of analog circuits works within a linear abstraction, and the implemented prototype checker tool.

Chapter 3

Validating Analog Functional Models

The linear abstraction explained in Chapter 2 enables functional equivalence checking between two analog representations: circuit vs. model.[1] First, we address how to formally check the equivalence of analog models by leveraging linear abstraction. This checking requires the generation of test vectors for both models, so Section 3.3 describes how we classify I/O ports in mixed-signal circuits to guide test vector generation. Section 3.4 and Section 3.5 explain how the prototype checker tool works. Chapter 4 demonstrates the utility of our checker tool with the circuits from a high-speed A/D converter.

3.1 Functional Equivalence of Analog Models

Functional equivalence checking of analog models is straightforward since the linear abstraction formally defines the functional behavior of an analog circuit; the checking is performed by extracting linear system models from both the circuit netlist and the Verilog model, and comparing the extracted models. As explained in Section 2.3,

[1]The checking is not necessarily between a circuit netlist and a Verilog model. Any comparison between two models is possible. For example, one can compare a Verilog-D model with a Verilog-AMS model

Model	Gain matrix $\mathbf{G}$	Relative error in [%] to Model 1
Model 1 (Circuit netlist)	$\begin{pmatrix} -1.0 & -0.026 \end{pmatrix}$	N/A
Model 2 (Correct)	$\begin{pmatrix} -1.0 & -0.028 \end{pmatrix}$	$\begin{pmatrix} 0 & 8 \end{pmatrix}$
Model 3 (CLK_o is inverted)	$\begin{pmatrix} 1.0 & 0.028 \end{pmatrix}$	$\begin{pmatrix} -200 & -207 \end{pmatrix}$
Model 4 (V_{ctrl} is inverted)	$\begin{pmatrix} -1.0 & 0.028 \end{pmatrix}$	$\begin{pmatrix} 0 & -207 \end{pmatrix}$

Table 3.1: Equivalence checking results of the duty-cycle adjuster shown in Figure 2.1 — gain matrices of various models.

the transfer (gain) matrix in Equation 2.1 completely specifies a linear system; it describes how the system response (output) changes with the inputs. Once a circuit and a Verilog model are mapped onto linear system models, their equivalence is formally checked by comparing two gain matrices:

$$\mathbf{G_{circuit}} \stackrel{?}{=} \mathbf{G_{Verilog}} \tag{3.1}$$

where $\mathbf{G_{circuit}}$ is the gain matrix of a circuit netlist and $\mathbf{G_{Verilog}}$ is the matrix of a Verilog model.

Most I/O port inconsistencies discussed in Section 2.1 can be detected by pairwise comparison of the two gain matrices to see if either the signs or relative magnitudes are different. A sign discrepancy shows that either a port is improperly connected to a circuit or the signal polarity is inverted. Similarly, comparing the relative magnitudes can detect whether a control bus is connected in a reversed order, or assumes a different encoding style. Since these errors usually produce a large discrepancy in value, they are easily detected with coarse comparisons.

Table 3.1 shows the extracted gain matrices of the duty-cycle adjuster shown in Figure 2.1. It lists the matrices of a circuit, a correct Verilog model, and two broken Verilog models when the system is mapped to the linear model in duty-cycle domain as described in Equation 2.2. As summarized in the table, comparing partial gains shows any discrepancy. When the netlist (Model 1) and the Verilog model (Model 2) are matched, the difference in partial gain values is relatively small. However, there is

a clear discrepancy when they are not matched; the relative error is large. Therefore, designers can easily recognize errors. Moreover, the cause of errors can be found by observing the partial gain from each input to an output. When compared to Model 1, Model 4 has polarity inversion in the partial gain G_2 from V_{ctrl}, but no inversion in the partial gain G_1.[2] In contrast, the signs in both G_1 and G_2 are inverted in Model 3. Thus one is able to identify that there is polarity inversion from CLK_i to CLK_o in Model 3 as well as the inversion from V_{ctrl}.

On the other hand, as opposed to boolean comparisons, comparing gain matrices is a bit vague since partial gains in a matrix are real numbers and any two real numbers are not the same. It might be necessary to define the tolerance bounds for assuring that two numbers are the same, i.e., two models are equivalent, so that the checking is more complete. To avoid defining the right tolerance, we take a different approach.

Rather than defining the tolerance bounds, we divide this model validation problem into two problems: one is to check any discrepancy between models at their I/O boundary by comparing their extracted linear models, and the other is to characterize the linear system model of a circuit netlist and force the linear system model of the parameterized Verilog model to the circuit netlist. As shown in Table 3.1, coarse comparison of the extracted models is sufficient to check if two models have any inconsistency at their I/O boundary. Once this checking is complete, one is able to plug the extracted gain matrix from the circuit netlist into the Verilog model. Then, the updated Verilog model is used for the system-level verification.

3.2 Analog Test Vector Generation

Without an abstract model as a guide, it is impossible to know if a set of test vectors for identifying a system is complete. Abstraction defines the space of possible outputs. For digital combinational logic, since a signal takes only either '1' or '0', it requires 2^N input vectors for N inputs to generate all possible combinations of inputs.

[2] G_1 and G_2 are partial gains of $\mathbf{G}$ in Table 3.1; $\mathbf{G}=\begin{pmatrix} G_1 & G_2 \end{pmatrix}$.

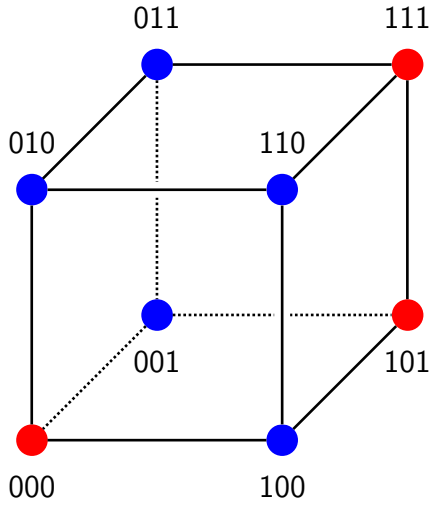


Figure 3.1: Digital response surface.

At first glance, it seems that generating analog test vectors for extracting the abstract model might be more difficult than digital vector generation since inputs can take more than two values. However, the smooth nature of the analog response surface makes the problem very simple. The unknown response to some inputs can be inferred by interpolating the known responses.

The following subsections begin by describing the way to generate analog test vectors for extracting a completely linear model, and then we generalize the method to a weakly nonlinear system model and a system model with strongly nonlinear, but still smooth response.

3.2.1 Extracting a Completely Linear Model

Linear abstraction simplifies the vector generation since it gets rid of any interaction between inputs. In a linear system, superposition holds; inputs contribute to outputs independently. The response of the system can be calculated by considering each input independently and summing the individual responses to each input. Like boolean abstraction for digital circuits, linear abstraction greatly reduces the possible model complexity of analog circuits, but in an orthogonal way. While boolean abstraction collapses the value space into discrete binary values, it does not reduce the

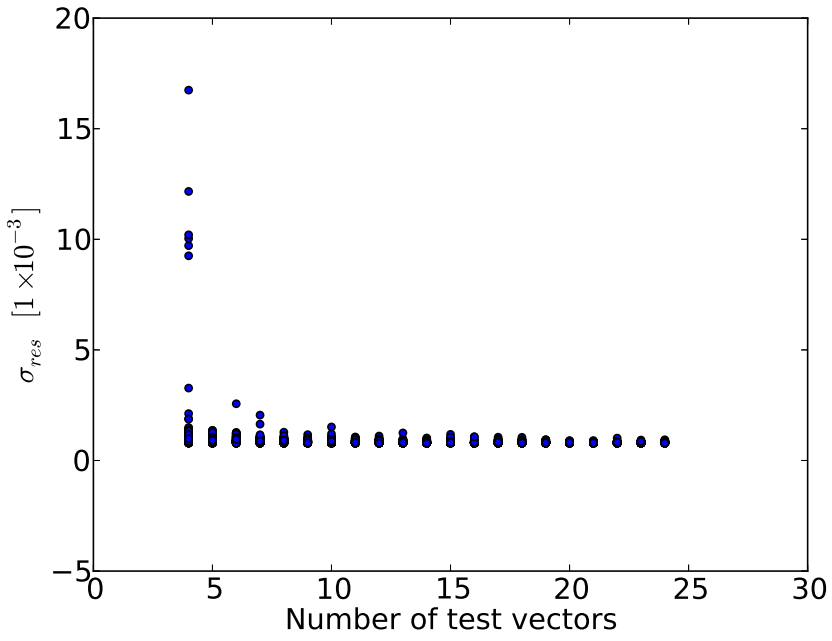


Figure 3.2: Standard deviation of residual errors, σ_{res}, of completely-linear models of the duty-cycle adjuster circuit in Figure 2.1 vs. the number of test vectors used in the model fitting.

dimensionality of the space. Since digital response is not smooth as shown in Figure 3.1, an output value generally cannot be predicted from another known outputs such that all possible input combinations should be enumerated. In fact, the key problem in digital verification is to efficiently explore a large number of states, which grows exponentially with the number of inputs or internal registers.

A linear abstraction does not have this problem of a exponentially-growing state space. As all inputs contribute to an output independently, one only needs to characterize the effect of each input on the output to fully explore an analog circuit. For instance, completely-linear system models of the duty-cycle adjuster circuit shown in Figure 2.1 are extracted with increasing numbers of test vectors. The standard deviation of the residual errors, σ_{res}, is calculated for 400 samples of each model. This calculation is repeated one hundred times. Figure 3.2 shows how quickly the collected residual errors decrease as the number of test vectors increases. As shown in

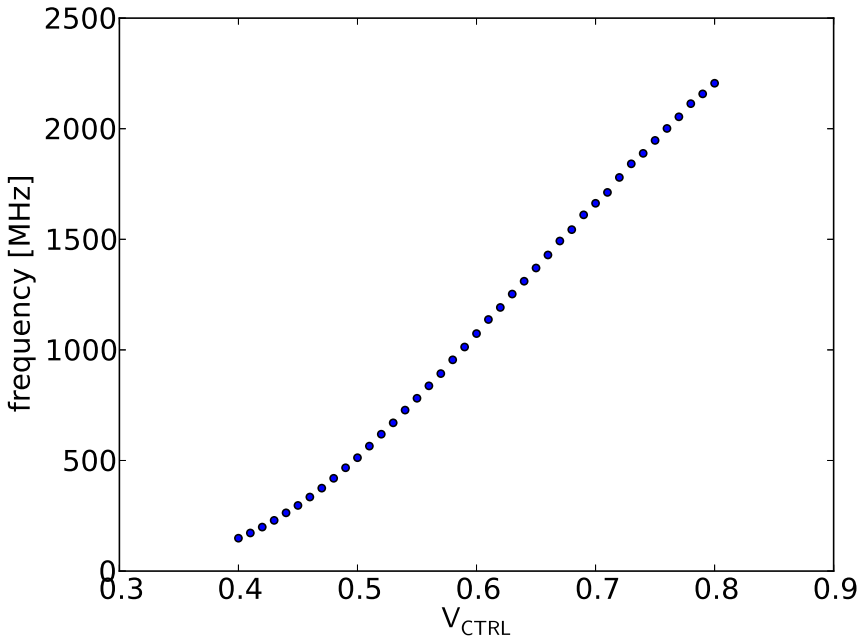
the figure, the maximum value of σ_{res} decreases dramatically as the number of vectors increases, which means that the linear system model of the duty-cycle adjuster can be extracted with a few test vectors and the extracted model is very accurate. The superposition principle also applies in time, such that the output is a linear combination of not only the current inputs but also the inputs at previous times. This leads to a well-known result that a linear system can be completely characterized by its response to an impulse arriving at each of inputs [42]. In many practical circuits, these impulse responses can be compactly expressed as a sum of a few exponentials, rather than as a general continuous-time function. It is this concise representation of a system that makes linear abstraction so powerful.

While extracting a completely linear model seems to require only $N + 1$ random test vectors for N analog inputs, we need to sample the response with more than $N + 1$ test vectors to ensure that the response actually fits well to a completely linear system, i.e., the residual error of the fitted model is small. There are two main reasons why the extracted linear model may be invalid: the circuit-under-test is inherently nonlinear in any of variable domains or the test setup is wrong. To ensure that the output response to any interaction between inputs is insignificant with a few test vectors, we adopted an orthogonal array (OA) for the vector generation, which is discussed in more detail in Section 3.4.1.

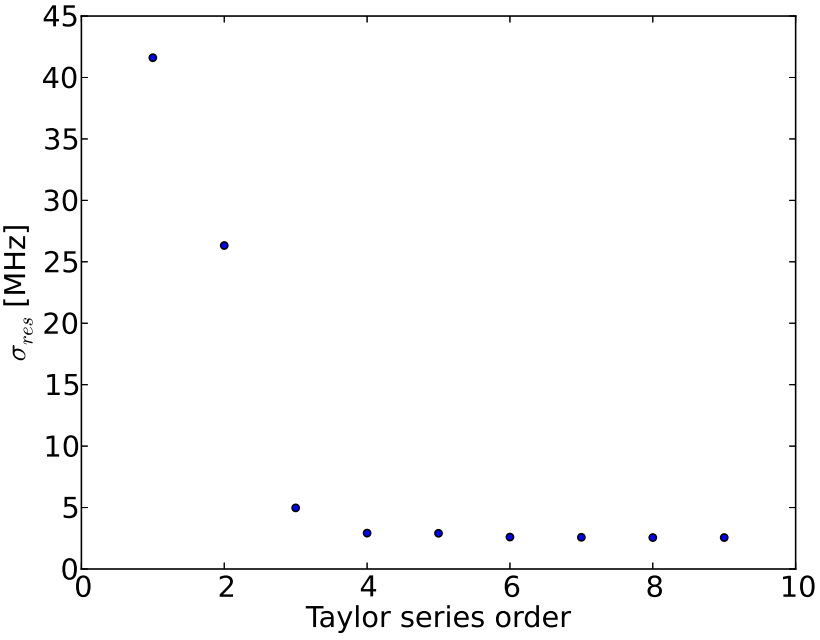
3.2.2 Extracting a Weakly Nonlinear Model

While comparing completely linear models ensures that two models are equivalent at their I/O boundary, one also wants to create an accurate Verilog model and compare the model with its circuit netlist. As mentioned previously in Section 2.3, the realistic behavior of an analog circuit is often well modeled by a weakly nonlinear system, and thus its extraction still requires a small number of vectors.

As we reviewed in Chapter 2, the analog response surface is smooth, which implies that an unknown response can be estimated by interpolating the existing response samples. Although a true analog response is not completely linear and thus not a complete hyperplane, such a response is weakly nonlinear and can be formulated



(a)



(b)

Figure 3.3: Weakly nonlinear models of a VCO transfer curve: (a) VCO transfer curve (b) standard deviation of the residual errors vs. fitting polynomial order.

with a low-order series expansion such as Taylor series with relatively small residual errors as explained in Section 2.3. Thus the gain matrix serves as an interpolating function which reconstructs the response with reasonable accuracy. Figure 3.3a shows the transfer curve of a ring oscillator circuit (output frequency vs. control voltage), and Figure 3.3b shows how the standard deviation of the residual errors decreases as the order of the fitted Taylor series increases. As shown in the figure, the weakly nonlinear behavior of the circuit is well captured with a low order Taylor series.

In terms of test vector generation, a smooth analog response means that the selection of analog vectors is not critical, and one can easily build a response model with a few vectors. Since the interpolating function for the response construction is at most a linear regression model with a low order series expansion, e.g., a third order Taylor series, a gain matrix can be extracted by simply finding the coefficients of the function, using only a few response samples.

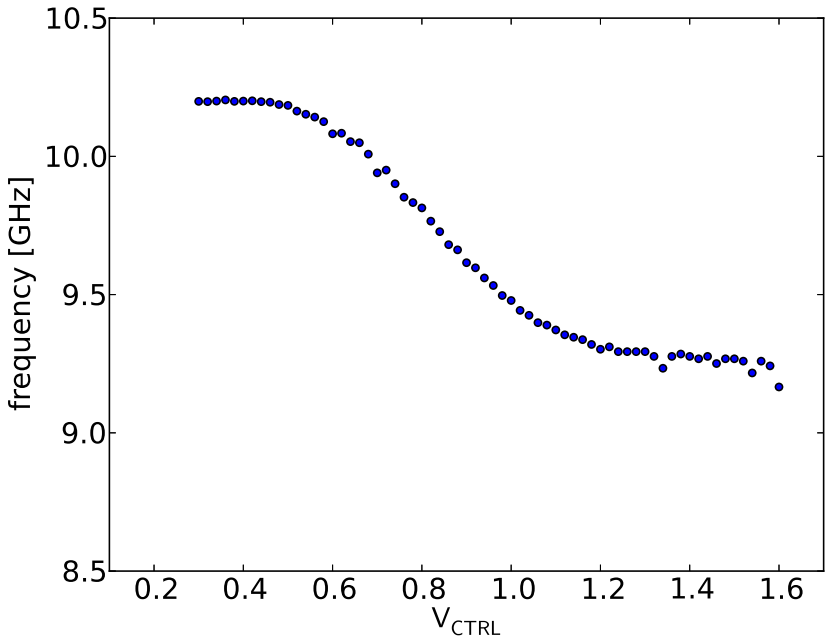
3.2.3 Piecewise Modeling of Analog Response

The ability to compare small nonlinear effects is enough to ensure that the circuits and models match for many analog circuits. In some cases, the functional model must map the response of a circuit over an input range when there is a large change in the transfer function. In these cases, the polynomial models do not work well, and using a piecewise linear abstraction is a better approach.

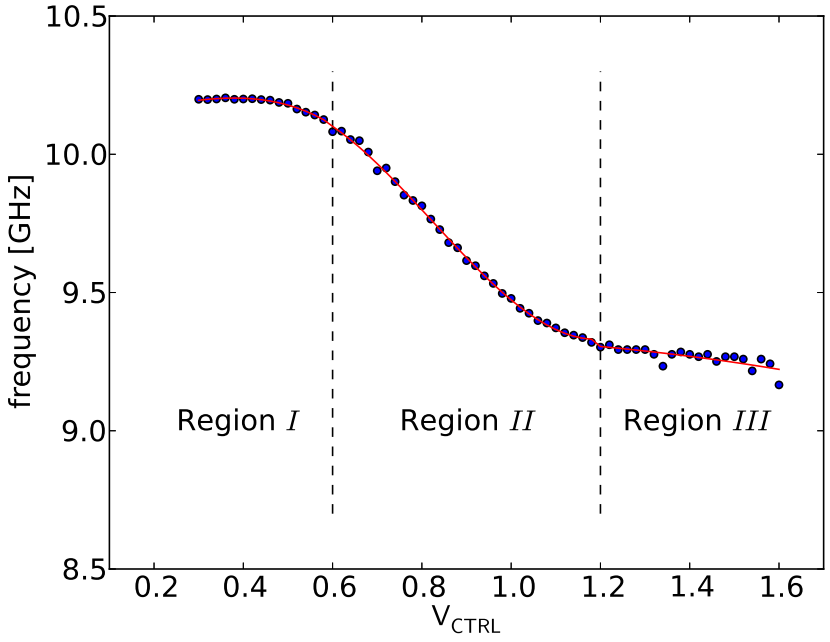
All the models must contain specifications of the valid input range on all of its inputs. This defines the space of possible input values that is used to generate the output response. When the transfer curve of a circuit is strongly nonlinear within valid input range, the input space can be partitioned into multiple regions and then the linear model in each region can be extracted. With this piecewise segmentation, one is able to build and check the model with high accuracy without resorting to a complicated model.[3]

For instance, Figure 3.4a shows the transfer curve of a LC-VCO [43], i.e., output

[3]In our checker tool, the user needs to partition the input space manually, so the legal input range information is provided for the tool. For each region, the tool checks if a linear system model is valid.



(a)



(b)

Figure 3.4: Piecewise modeling of LC-VCO output clock frequency response: (a) LC-VCO response and (b) piecewise model of the LC-VCO response.

Port Intent / Data Type	Analog	Digital	Circuit
Real	Analog	-	Function
Boolean	Quantized Analog	True Digital	

Table 3.2: Port classification in our linear system context.

clock frequency vs. input control voltage V_{CTRL}, which has a strongly nonlinear response as the input control voltage is away from the center of the input range. When a PLL is built with this LC-VCO, the nonlinear behavior affects PLL locking time which may need to be accurately modeled. Instead of a higher order model, one can partition the control voltage input into a weakly nonlinear region (Region II) and two compressed regions (Region I and III) as shown in Figure 3.4b, and model the response in each region with less than third order Tayllor series.

3.3 Port Classification

If a circuit has only analog I/O ports, the vector generation for extracting its linear system model is simple as explained in the previous section. However, digital I/O ports must be considered in the test vector generation of mixed-signal circuits. We have classified the port intent in our linear system model framework to guide the test vector generation of mixed-signal circuits.

As shown in Table 3.2, ports are classified into analog ports, digital ports, quantized analog ports, and function ports. A signal into/out of a circuit may be represented as either a real value, i.e., an analog signal, or a boolean value, i.e., a logic signal. As will be explained later, a digital input could have two different intents which need to be handled differently. A function port is a special type of port, which is explained at the end of this section.

The following subsections explain the details of each port type with the amplifier circuit example illustrated in Figure 3.5. This amplifier takes differential voltage signals as inputs from either $V_{IN+/-}$ or $V_{CAL+/-}$ depending on the digital input calib_en, and outputs an amplified differential voltage signal to $V_{OUT+/-}$. The bias

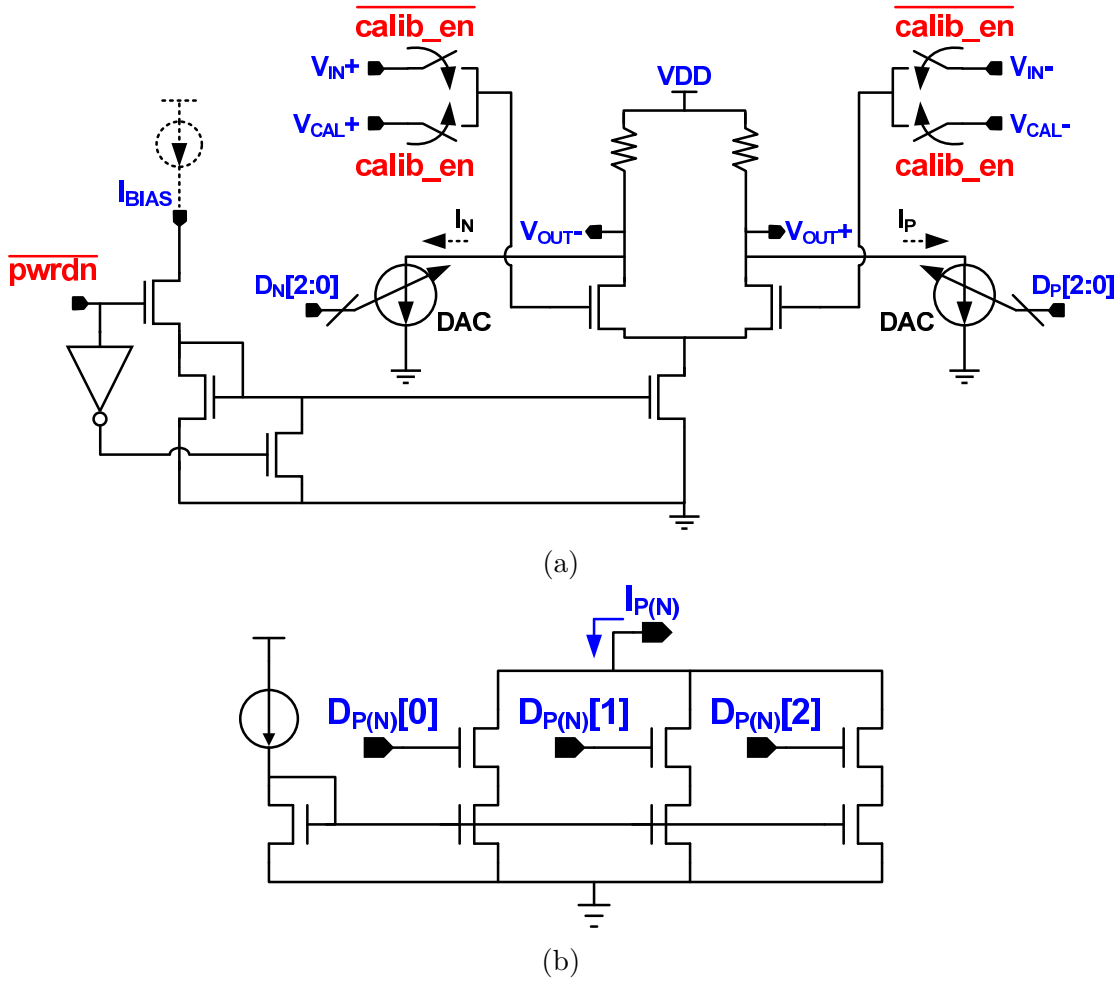


Figure 3.5: An example circuit to explain the type of port: (a) differential amplifier and (b) current steering D/A converter for controlling the input offset voltage in (a).

current I_{BIAS} adjusts the gain of the amplifier while the input offset voltage is adjusted digitally by D_P and D_N.[4] The digital input /pwrdn shuts down the circuit by disconnecting the bias current, and VDD is the power supply input of the circuit.

[4]It is assumed that I_{BIAS} adjusts the gain although it may adjust either output voltage swing, bandwidth, or gain depending on the designer's intent.

3.3.1 Analog Port

Analog I/O ports are the inputs and outputs of a linear system model. The signals on analog I/O ports are real values and analog information is carried from the input port to the output port. An analog input port does not change any circuit's property and the response at the analog output port can be directly measured. In Figure 3.5, $V_{IN+/-}$, $V_{CAL+/-}$, and $V_{OUT+/-}$ are such ports. For the amplifier, the intended linear system is between the input and output differential voltages with specifications such as gain and bandwidth. Of course VDD is also an analog input port as the signal on VDD is amplified to $V_{OUT+/-}$.

In coupled linear systems mentioned in Section 2.3.2, an input controls some parameters of a circuit, and thus this input is named as an *analog control input port*. The controlled parameter may depend on the designer's intent which is usually described explicitly in a Verilog model and/or a measurement script, i.e., a circuit testbench. For example, I_{BIAS} in Figure 3.5 is such an analog control input port as it adjusts the gain of the linear system between analog I/O ports.

In such coupled systems, it is necessary to pull out the properties being controlled as outputs since the properties are not the direct outputs of the system. *Pseudo output ports* are explicit representation of such properties, and usually extracted by measurement modules;[5] pseudo outputs are the results of simulations which measure the properties of interest. We assume that the simulation setup that measures these properties is likely to be available from circuit designers since that setup is also necessary for the circuit design and verification. From the test setup, one easily recognizes which properties of a system are being controlled. For instance, the pseudo output of the amplifier shown in Figure 3.5 is its gain.

It is noteworthy that signals on analog ports are not necessarily defined in electrical domains. Sometimes, analog inputs and outputs of a linear system are defined in other domains by translating the domain of circuit's I/Os. For example, we observed that the duty-cycle adjuster shown in Figure 2.1 is linear with the clock input and output in the duty-cycle domain. To test the circuit, one drives a real-valued signal to the

[5]To do this, one may write a measurement module in HDLs, or extract the property by post processing of simulation results.

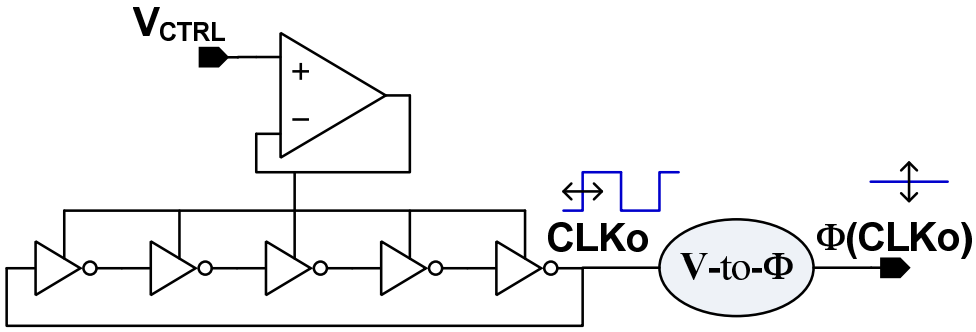


Figure 3.6: Voltage controlled oscillator.

input in duty-cycle domain rather than voltage domain, and measures a real-valued response at the output in the same domain. Since the linear system model cannot be constructed without a domain translator, the translator should be considered as a part of a circuit

It is possible that the variable domains of I/Os are mixed. For instance, for checking the voltage-controlled oscillator (VCO) shown in Figure 3.6, its linear system model is between the input voltage V_{CTRL} and the phase or frequency of the output clock CLKo. Hence, the clock output of the VCO is regarded as an analog output port in phase/frequency domain despite its binary waveform in voltage domain, and V_{VCTRL} is an analog input port in voltage domain.

3.3.2 True Digital Port

Some digital inputs change the functional mode of a circuit, generating different circuit configurations. We call these inputs *true digital ports* since they control what circuit configuration is formed. Since each circuit configuration can have a completely different result surface, one needs to measure each configuration. Similar to the digital vector generation, N true digital port inputs can create 2^N different linear systems and each system configuration must be verified for modeling checking. While this exponential order seems troubling, the number of this type of input is usually small in mixed-signal circuits. Thus the verification complexity from true digital inputs is generally manageable.

In Figure 3.5a, the power down input /pwrdn and the signal select input calib_en

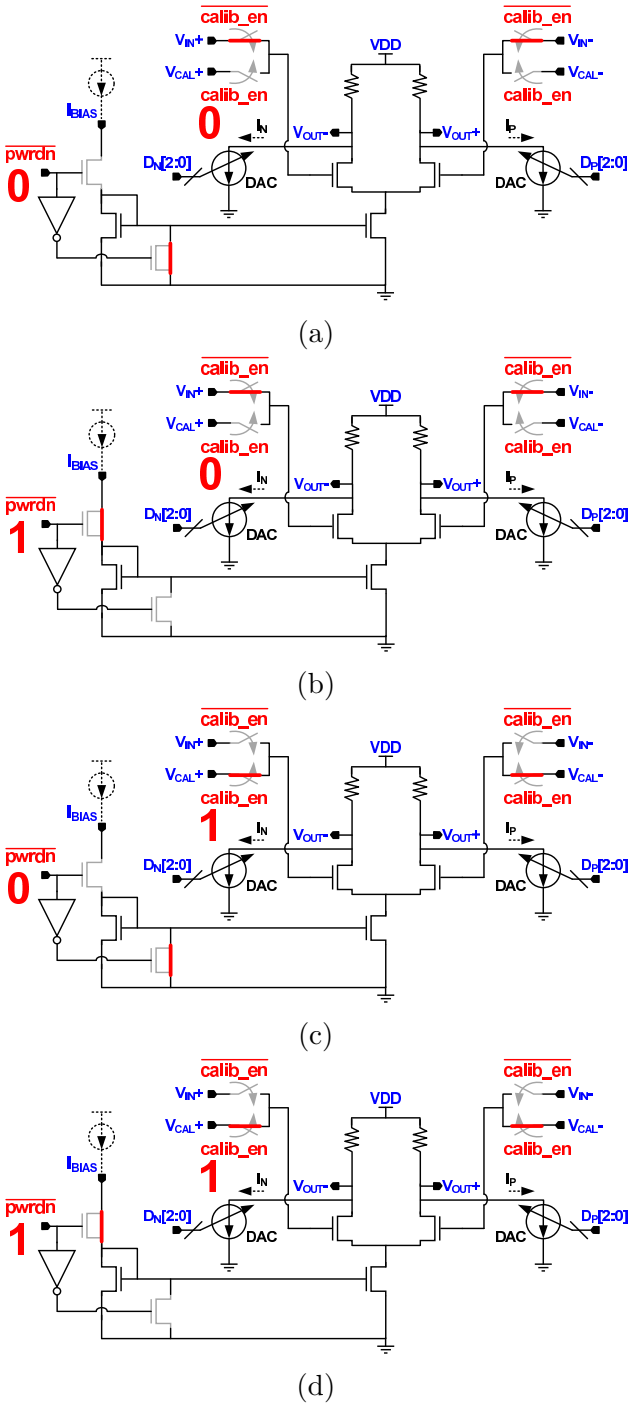


Figure 3.7: Circuit configurations of the differential amplifier shown in Figure 3.5, enumerated by true digital ports: the combination of calib_en and /pwrdn are (a) '00', (b) '01', (c) '10', and (d) '11'.

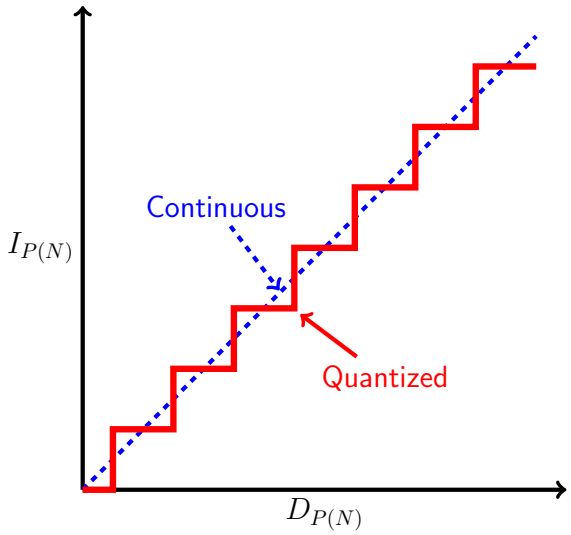


Figure 3.8: Transfer curve of the D/A converter in Figure 3.5b.

controlling the switches are true digital ports. When /pwrdn is asserted, the circuit is in a power down mode which operates on a completely different mode from the normal operation when /pwrdn is de-asserted. The calib_en configures the signal flow — it selects inputs, either $V_{IN+/-}$ or $V_{CAL+/-}$, to the amplifier. Since there are two true digital inputs, there are $2^2 = 4$ different linear circuit configurations as enumerated in Figure 3.7. For each circuit configuration, one needs to check the functional equivalence between the circuit and the model.

3.3.3 Quantized Analog Port

The dominant kind of digital port in mixed-signal circuits is a quantized analog I/O or control port, i.e., *quantized analog port*. For these ports, the boolean value does not change the circuit function, but acts as a quantized analog input or output of the circuit. As stated in Section 1.1, it is quite common that digital inputs adjust properties of a circuit in a quantized step or just add a quantized analog signal in most digitally-assisted analog circuits. For example, the digital inputs of a digital-to-analog converter (DAC) are essentially analog despite their data representation being boolean.

For instance, D_P and D_N in Figure 3.5a are quantized analog inputs which adjust the input offset voltage of the amplifier. The inputs skew the differential currents drawing on the amplifier's outputs with the current steering D/A converter illustrated in Figure 3.5b. The offset currents, I_P and I_N, are controlled in a quantized step as shown in Figure 3.8 and the relationship to inputs is linear, which is given by

$$I_{P(N)} = \sum_{k=0}^{2} \alpha_k \cdot \mathrm{D_{P(N)}[k]} \tag{3.2}$$

where α_k is the partial gain contributing to $I_{P(N)}$ from $\mathrm{D_{P(N)}[k]}$, k-th bit of the quantized input $\mathrm{D_{P(N)}}$. Although these ports take on discrete values, their intended functionality is essentially analog in nature. Also, since their circuit implementations are often in regular, parallel structures, they can be automatically distinguished from true digital ports via graph isomorphism which is discussed further in Section 3.5.

From a vector generation perspective, the most important feature of such ports is that superposition holds, which can greatly reduce the number of test vectors needed to extract a system model. The number of vectors for a quantized analog input is linearly proportional to its bit width. This linear growth is in strong contrast to the exponential growth with the bit width for true digital inputs. For example, one only needs to toggle each bit of $\mathrm{D_{P(N)}}$ once to extract α_k's in Equation 3.2. This linear growth is especially important because the number of quantized analog inputs increases as analog circuits exploit more digital calibration and compensation loops. Furthermore, outputs adjusted by these quantized analog inputs are still analog signals, e.g., $I_{P(N)}$ in Equation 3.2, so that quantized analog inputs can be tested independently of other analog inputs.

3.3.4 Function Port

There is a special type of port that is essentially part of the system rather than an I/O port of the system. These ports help enable the operation of the circuit. For instance, the sequencing clocks in switched capacitor circuits, the local oscillator in mixers, and the sampling clock of clocked comparators are signals of this type. These

ports do not carry any information at all and do not control any property of the system. However, the circuit will not work unless these function ports are driven correctly. From a model checking perspective, these ports are verified by checking if the Verilog model is receiving the same signal patterns from the function ports during system validation as the corresponding circuit under characterization, rather than if it produces a consistent response to different input values. Therefore, there should be proper assertions in a functional model to ensure that the patterns from the ports are valid.

Function ports also arise in controlled systems. For instance, the VGA circuit in Figure 2.2 is checked with two tests for each of coupled linear systems: from the voltage input (analog input port) to the voltage output (analog output port) and from the bias current input (analog control input) to the gain of the first system (pseudo output port). In the first test, the bias current input is driven with a constant current source to set the operating point of the circuit while this input is examined in testing the second system. Thus the bias current input is a function port in testing the first system.

3.4 Model Checking Procedure

Figure 3.9 provides an overview of how the model checking is performed given that the port intents are labeled and the test setup for running simulations is prepared. The detailed description of port labeling and test setup is discussed in Section 3.5.

Test vector generation is straightforward given the port labels. M-bit true digital inputs generate 2^M linear circuit configurations by enumerating all possible combinations of such inputs. For each linear circuit configuration, the same test vectors of N analog/quantized (control) inputs are generated and exercised. Ideally, $N+1$ test vectors are necessary for N analog/quantized (control) inputs to extract a completely linear system model. Also, the sampling of input space could be random as analog response is smooth and thus the choice of test vectors is not critical. For robustness, however, we generate more than the minimum number of vectors, and perform design of experiment (DoE) rather than sample the response with random vectors. This will

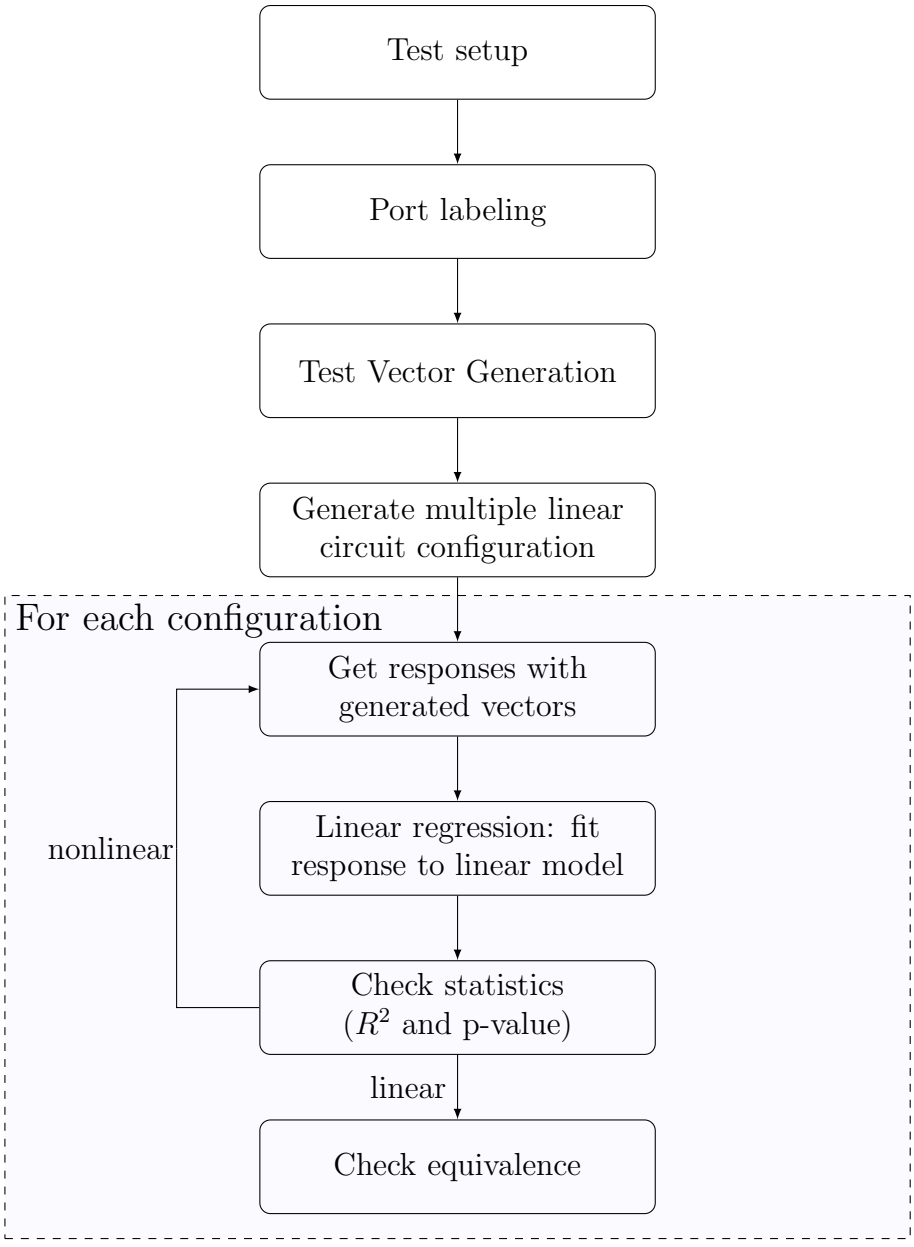


Figure 3.9: Procedure for functional equivalence checking.

be explained further in Section 3.4.1.

For each circuit configuration enumerated by true digital inputs, the generated (quantized) analog vectors are run on the circuit netlist using an analog/mixed-signal (AMS) simulator and on the Verilog model using a functional simulator. The gain matrices of both models are then extracted by performing linear regression on the response samples of analog/pseudo outputs to analog/quantized inputs.

After performing linear regression on response samples, the residual errors of the fitted model are examined to check the fidelity of the model. This is mainly judged by R^2 which is given by

$$R^2 = 1 - \frac{\sigma_{res}^2}{\sigma_Y^2} \tag{3.3}$$

where σ_{res} and σ_Y are the standard deviation of the residual errors and the output responses, respectively [38]. Large fit errors, i.e., R^2 far from 1 (e.g., 0.90), indicate problems in the test setup such as incorrect analog input range, inappropriate domain translator, or a digital input labeled as a quantized analog input. If R^2 is low, an error message is given and the designer must intervene. For the analysis, the number of vectors is adaptively controlled while observing statistical results. Of course the test vectors and their simulation results from the previous iterations are put together with current data for the regression. If the number of iteration exceeds a certain predefined value, the tool gives a warning that the sanity check has failed.

Another important statistical metric of linear regression is p-value of partial gains. If the residual errors of a fitted model are large, some partial gains become statistically insignificant, identified by a high p-value [38]. Intuitively, the p-value is inversely proportional to the signal-to-noise ratio, where the signal is the response change contributed by the partial gain and the noise is the residual error of the fitted model. Thus, if p-value is high, one cannot reject a null hypothesis which postulates the corresponding partial gain is zero. We treat the linear regression result statistically insignificant if p-value of any partial gain is higher than 0.05 when performing case studies.[6]

In checking the models, the sign and relative magnitude of the matrices (output

[6]In statistics, one rejects null hypothesis when the p-value is less than the significance level, which is often 0.05 or 0.01.

response vs. analog/quantized input ports) should be consistent for each circuit configuration created by true digital inputs.

3.4.1 Oversampling of Response

As mentioned in the previous section, more than the minimum number of test vectors for analog/quantized inputs are sampled. To get system responses for a circuit netlist, our characterization method relies on circuit simulations through SPICE-like simulators, which inevitably have simulation noise because of numerical integration during transient simulations. This simulation noise is averaged out by sampling more responses when running linear regression. It is also obvious that building a more accurate model with Taylor series expansion requires more response samples to run than simple linear regression. Doing so allows us to model the weakly nonlinear behavior of a circuit. Most importantly, we need to sample more responses to ensure that the linear assumption is valid.

When the linear assumption fails, there are two main classes of causes: either the test was not set up properly or the circuit structure is not what we assumed. There are four typical examples: 1) the variable domain in the test setup is incorrect, 2) some inputs are out of their legal region, 3) a tunable circuit is not properly decoupled into coupled linear systems in the test setup, and 4) a false labeling of a quantized analog port is possible for an unusual circuit. For instance, the output current I_{OUT} in Figure 3.10 is not a linear, weighted summation of D[0] and D[1] while our tool would recognize them as a quantized analog port. In any of these cases, it is necessary to sample the response enough, so that the tool checks whether the fitted linear model is valid or not. Otherwise, the checker may result in a false negative.

There are various sampling methods to check this linear assumption. In the implemented checker, orthogonal array (OA) testing is adopted to generate analog vectors. OA testing is an efficient way of performing experiments to minimize the number of experiments and to check any interaction between inputs which assumed to be independent [44]. OA vectors with M-level, strength of two for N analog/quantized analog inputs are examined to get the response. The number of level M is determined

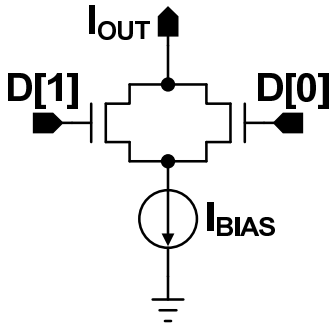


Figure 3.10: Circuit structure causing false labeling of a quantized analog port.

by the user. One may want to select more levels to get more accurate models with higher order polynomials and smaller residual errors. On the other hand, the vectors with strength of two examine the interaction terms between any of two inputs either to validate the system is linear or to get the cross product of inputs, which reduces the overhead in the number of test vectors and performs the model comparison faster.

3.5 Equivalence Checker Implementation

This section describes the implemented checker tool, beginning by explaining how the test setup is configured. Next, we describe how the port intents are labeled to generate test vectors for comparing analog models.

3.5.1 Test Setup

This checker requires user inputs to set up tests. While generic tests are available in digital circuits because of the unique domain for boolean abstraction and confined boolean values, the intended linear system is often manifested in different variable domains (e.g., phase domain for a PLL and duty-cycle domain for a duty-cycle adjuster) and the system's properties of interest are different. Therefore, proper tests should be provided by users, which means that the checker is not completely automatic, but is implemented to minimize user inputs.

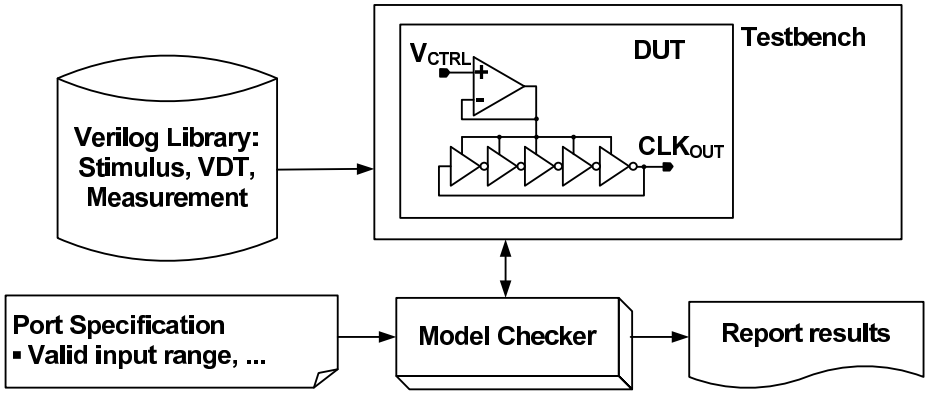


Figure 3.11: Elements of the test setup.

Data Type	Property	Data Type	Property
real	port type	boolean	bit width
	domain		encode
	range		prohibited

Table 3.3: Port specification.

Figure 3.11 illustrates the elements of a required test setup.[7] It is composed of various information needed for building simulation testbenches and generating test vectors. Some circuits require multiple tests. For instance, we decoupled a variable gain amplifier exemplified in Section 2.3.2 into two coupled linear systems which need separate testbenches.

First, port specifications should be provided for generating test vectors. The attributes on port specifications are different depending on whether the port signal is real or boolean, which is summarized in Table 3.3. A port with a real value signal is an analog port and thus *port type* property can be analog I/O, control input, and pseudo output port. It also has a *domain* property to define its variable domain.[8] The legal input/output range during device operation is specified by *range*. On the

[7] In our implementation, a user also needs provide compilation, elaboration, and simulation options of simulators.

[8] A user chooses the value of this property among the reserved words such as phase, delay, voltage, and so on.

other hand, a port with boolean signal is either quantized analog port or true digital port. The *bit width* defines the number of wires in this port, and *encode* indicates how data is encoded on these wires, e.g., binary or thermometer. This encoding information is double-checked with the extracted gain matrix later. Sometimes, only a subset of possible values is allowed for digital inputs so that *prohibited* defines a list of prohibited values.

Since the output responses of a circuit/Verilog are collected through simulations, designers need to find and create testbenches to measure the (pseudo) analog outputs for given test vectors. Testbenches either directly measure the response or post-process the simulation results depending on what is being measured. For example, the output current of a current steering D/A converter can be directly measured. If one wants to estimate the transfer function in the frequency domain from the impulse response, a script for post-processing the transient simulation data (impulse response) is required.

The setup also describes how components are connected. The components include device under test (DUT), stimulus, variable domain translator, and measurement, which are essential parts of a typical testbench. Each analog input must have a domain translator to define the correspondence between the number in a functional model and the analog signal in a circuit netlist. The translator is quite simple for electrical domains, since it maps real numbers to voltages or currents, or can be slightly more complex if the system is linear in other domains such as duty cycle and phase. All these data except the domain translators should be available from the circuit level block verification, so it should not require much additional effort.

Note that there could be mismatches between the testbenches of a circuit netlist and a Verilog unless the testbenches are generated from the same source. To prevent the mismatches, a common testbench description is embedded in the setup. Based on this embedded description, testbenches for both models are generated from the template shown in Listing 3.1.[9] From the information relevant to the testbench in the test setup, the tool elaborates the template so that it generates both a Verilog-AMS testbench for simulating a circuit netlist and a Verilog testbench for running a Verilog

[9]The template is written in `EmPy` markup language [45].

Listing 3.1: EmPy template for generating a Verilog(-AMS) testbench.

```
@# Template for testbench
@[if model == 'ams']
`include "disciplines.vams"
`include "constants.vams"
@[end if]

@testbench['pre_module_declaration']

`timescale @simulation['timeunit_str']/@simulation['timeprec_str']
////////////////////////////////////////////////////////////////////
// @testname testbench
////////////////////////////////////////////////////////////////////
//module @testname;
module test;

////////////////////////////////////////////////////////////////////
// declaration of wires
////////////////////////////////////////////////////////////////////
@[for p,v in testbench['wire'].items()] @[for x in v]
@p @x;
@[end for] @[end for]

////////////////////////////////////////////////////////////////////
// Custom code here
////////////////////////////////////////////////////////////////////
@testbench['custom_code']

////////////////////////////////////////////////////////////////////
// instantiation of modules
////////////////////////////////////////////////////////////////////
@[for p in testbench['instance']]
@p @[end for]

////////////////////////////////////////////////////////////////////
// instantiation of file dump statement
////////////////////////////////////////////////////////////////////
@[for p,v in testbench['response'].items()]
@v['verilog_str']
@[end for]

////////////////////////////////////////////////////////////////////
// simulation time control
////////////////////////////////////////////////////////////////////
initial
  #(@(int(simulation['sim_time']/simulation['sim_timeunit'])))
  $finish;

endmodule
```

model. For this to work, all the components used in the testbenches such as stimulus, domain translator, and measurement module must exist in both Verilog-AMS and Verilog HDL environments.

3.5.2 Labeling Ports

The tool starts the checking process by labeling ports. Although the ports are already specified in the test setup, the tool checks the sanity of their specifications to detect possible human errors. For example, a user may incorrectly specify a voltage input to an amplifier as a true digital input port instead of an analog input port. Such an error results in invalid vectors for the input. This port labeling checks whether the port specifications from the user inputs match the underlying port intents in the testbench.

It is trivial to distinguish analog ports from digital ports since all analog ports require domain translators being attached to a DUT — it is necessary even for voltage or current domains since no signal discipline exists for real numbers in HDLs. Thus the tool checks if the variable domain of an analog port is connected to the corresponding domain translator.

It is essential to distinguish a quantized analog port from a true digital port to reduce the number of test vectors required during validation.[10] Since a quantized analog port has very regular, parallel physical structure in most cases, we apply a simple graph isomorphism test to identify it.[11] As depicted in Figure 3.12, the algorithm simply checks whether the circuit structure is changed by removing each input. If the circuit topology remains unchanged, implying that it has another structure in parallel, the inputs are labeled as quantized analog inputs. The graph comparison is done without device size information since the effect of different element size is manifested in the gain matrix from each bit of a quantized analog input. While this approach works on most inputs we have seen, it will fail in some circuits, like R-2R

[10] All ports except analog and quantized analog ports will be recognized as true digital ports.

[11] A circuit netlist is converted into an undirected graph; all devices and ports become labeled nodes, and wires become edges. The graph isomorphism of converted graphs is checked using `NetworkX` software library [46].

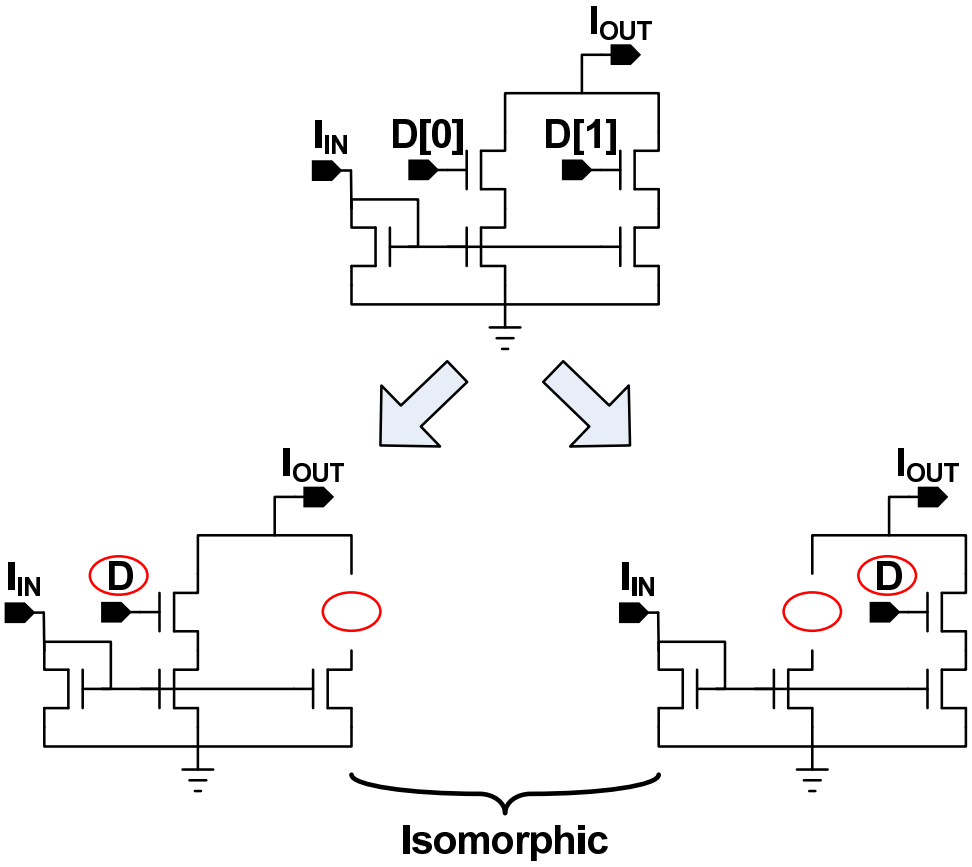


Figure 3.12: Graph isomorphism test for identifying a quantized analog port.

ladders where the topology does depend on inputs. In addition, our current tool is not very sophisticated. One needs to manually strip off any digital gates driving the quantized analog inputs. Otherwise, they will fail the isomorphism test.

3.6 Summary

In this chapter, we described a simple but efficient method to check the functional equivalence between a mixed-signal circuit and its HDL functional model. This problem is made vastly simpler by leveraging the fact that most analog circuits are trying to implement a linear function in some domain. Since superposition holds in linear systems, the linearity assumption reduces the number of input test vectors needed

to be linear on the number of analog inputs. In most cases one would use slightly larger number of input vectors than the minimum required, allowing the system to check the modeling assumptions, as well as extract the needed partial gains. The only complexity is with digital inputs that end up changing the underlying circuit. While these inputs are rare, each potential circuit must be characterized and compared with the functional model.

Chapter 4

A 40-Way, Time-Interleaved ADC

In this chapter, a sub-ADC and phase interpolator in a 40-way, time-interleaved ADC are examined to demonstrate the utility of the model checking method explained in Chapter 3. This ADC uses 40 sub-ADCs in parallel to increase the effective sampling rate. The concept of this time-interleaved ADC is illustrated in Figure 4.1 [47]. The 40-channel sub-ADCs alternatively sample an input signal, so that the conversion cycle of each sub-ADC is relaxed to $40 \cdot T_S$, where T_S is the sampling period of the ADC.

A single-slope ADC is used for the sub-ADC implementation, which is shown in Figure 4.2 [47]. The operation of the ADC during a single conversion cycle is as follows. A differential input voltage on in+ and in− is first sampled onto a pair of hold capacitors C_{H+} and C_{H-} by a sampling clock sclk. The current source from a ramp generator is then used to charge C_{H+}, so the voltage on node hold+ ($V_{\text{hold+}}$) ramps up linearly. When $V_{\text{hold+}}$ crosses the voltage on node hold−, the comparator fires and triggers a set of latches to latch the current counter value. Since the latched counter state is proportional to the differential input voltage, the counter value holds a digital representation of the ADC input.

The counter is driven by a set of clock signals from an integrated PLL which is also the source of 40 phase sampling clocks. To ease the clock frequency requirement of the PLL, the phase of two clocks, a clock and its quadrature clock from the PLL, are interpolated instead of generating higher frequency clocks, so that the finely-spaced

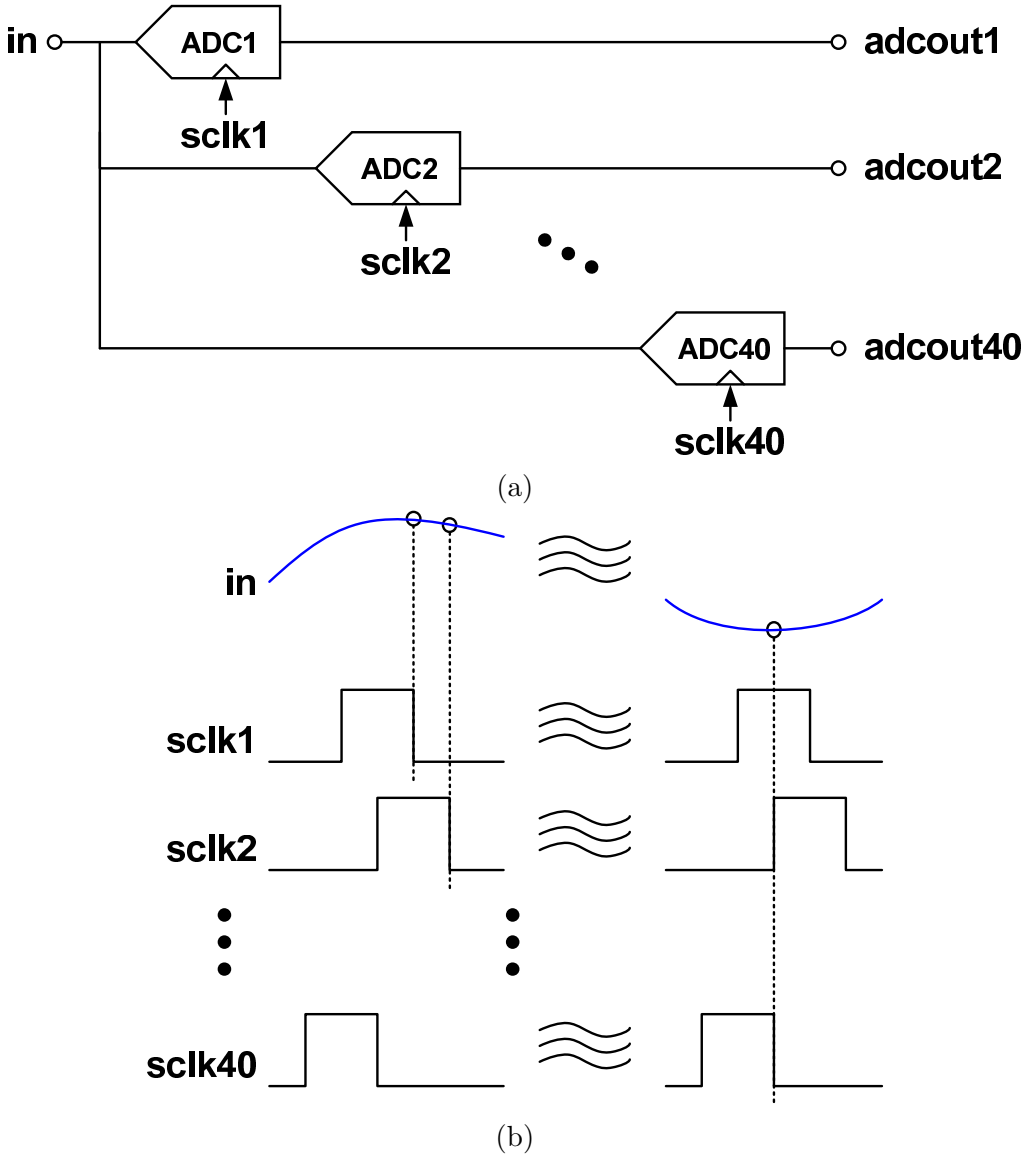


Figure 4.1: Concept of 40-way, time-interleaved A/D conversion: (a) simplified block diagram of a converter (b) 40-way, time-interleaved sampling.

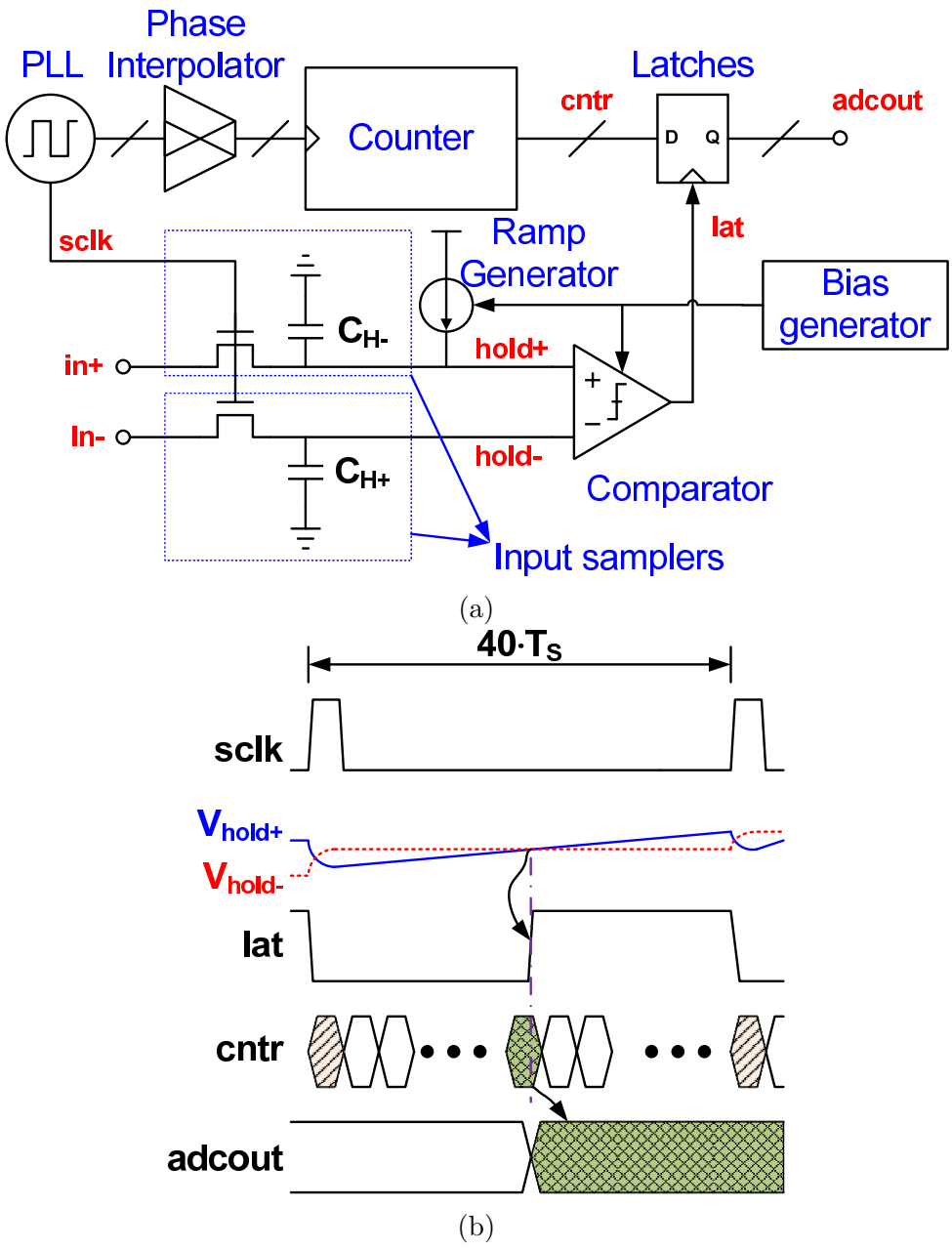


Figure 4.2: A simplified single-slope ADC: (a) block diagram and (b) timing diagram.

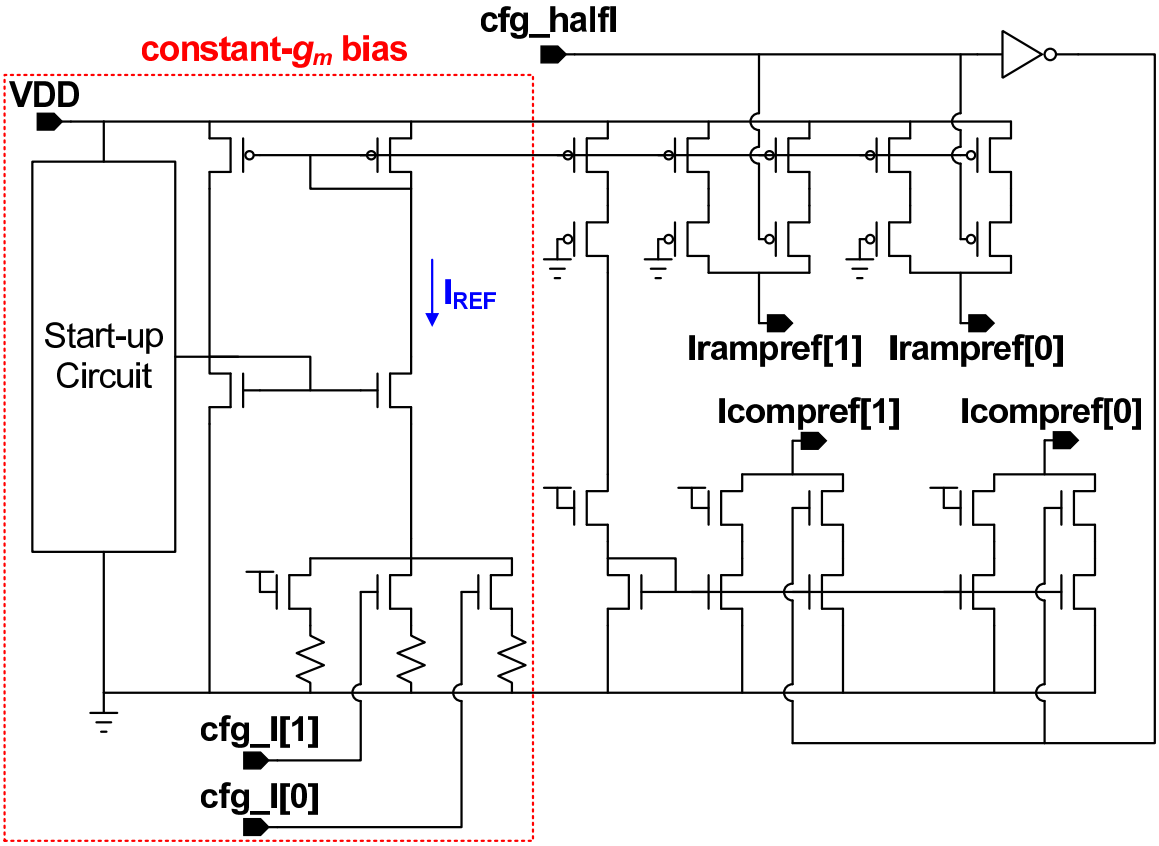


Figure 4.3: Bias generator.

clocks from the phase interpolator drive the counter.

The sub-ADC was broken into circuit blocks, and a number of different designers created the blocks and corresponding Verilog models. The model checking is performed for each block with our checking tool.

4.1 Bias Generator

The circuit diagram of a bias generator is depicted in Figure 4.3. It generates multiple currents that are distributed to the ramp generators and comparators in a sub-ADC. A constant-g_m bias circuit generates an adjustable reference current I_{REF}. The two-bit digital input cfg_I controls the total resistance of the resistor array, and sets I_{REF}. The generated current is then amplified by the current mirrors, and the mirrored

Pin Name	Signal Type	I/O	Bit Width	Description
Irampref[0]	Analog	Output	-	Current output to ramp generator
Irampref[1]	Analog	Output	-	Current output to ramp generator
Icompref[0]	Analog	Output	-	Current output to comparator
Icompref[1]	Analog	Output	-	Current output to comparator
VDD	Analog	Input	-	Power supply voltage input
cfg_I	Digital	Input	2	Adjust the reference current I_{REF}
cfg_halfI	Digital	Input	1	Reduce the output currents by half

Table 4.1: Physical pin description of the bias generator.

output currents are fed to the other circuit blocks. The currents flowing through Irampref [1: 0] and Icompref [1: 0] outputs are reduced to half of their nominal values when the digital input cfg_halfI is set to '1'. Table 4.1 summarizes the physical pins of the circuit.

Test Description

The intent of the circuit is easily understood in current domain since it generates current sources. The output current through Irampref[0] from the inputs is modeled as

$$\mathbf{I}(\text{Irampref[0]}) = \left(1 - \frac{1}{2} \cdot \text{cfg\_halfI}\right) \cdot \left(1 + \sum_{k=0}^{1} \alpha_k \cdot \text{cfg\_I[k]}\right) \cdot I_{BASE} \tag{4.1}$$

where $\mathbf{I}$(Irampref[0]) is the output Irampref[0] in current domain, α_k is the partial gain from the cfg_I[k] input, and I_{BASE} is the current $\mathbf{I}$(Irampref[0]) when cfg_I and cfg_halfI are equal to '00' and '0', respectively. The other outputs are modeled in a

Port Name	Properties		
	Port Type	Domain	Range
Irampref[0]	Analog Output	Current	[0.0,10µ] [A]
Irampref[1]	Analog Output	Current	[0.0,10µ] [A]
Icompref[0]	Analog Output	Current	[0.0,10µ] [A]
Icompref[1]	Analog Output	Current	[0.0,10µ] [A]

Port Name	Properties			
	Port Type	Bit Width	Encoding	Prohibited
cfg_I	Quantized Analog Control Input	2	Binary	-
cfg_halfI	True Digital Input	1	Binary	-

Table 4.2: Port classification of the bias generator.

similar way.

From Equation 4.1, the intent of each physical pin, i.e., the port class in our linear system model, can be identified, which is summarized in Table 4.2. The digital input cfg_I is labeled as a quantized analog control input because it adjusts the output currents in a quantized step. The other digital input cfg_halfI is a true digital input port since it changes the functional mode of the circuit, i.e., nominal current output mode vs. half current output mode, which generates two linear models given by

$$\mathbf{I}(\text{Irampref[0]}) = \left(1 + \sum_{k=0}^{1} \alpha_k \cdot \text{cfg\_I[k]}\right) \cdot I_{BASE} \tag{4.2}$$

where cfg_halfI is equal to '0', and

$$\mathbf{I}(\text{Irampref[0]}) = \frac{1}{2} \cdot \left(1 + \sum_{k=0}^{1} \alpha_k \cdot \text{cfg\_I[k]}\right) \cdot I_{BASE} \tag{4.3}$$

where cfg_halfI is equal to '1'. Based on these linear models, the test is configured and the linear models of both the circuit and its Verilog model are extracted by the checking tool.

Note that the test to measure the effect of the power supply input VDD on circuit performance, e.g., power supply rejection, is not included because its effect is not described in the original Verilog model. However, it should be included to build more accurate model since the power supply is also an analog (control) input to the circuit.

Model Checking Result

Since Equation 4.1 is linear, a single test is prepared for sampling the response of output currents to inputs. Given this test setup, the extracted linear system equations of a circuit netlist are given by

$$\begin{pmatrix} \mathbf{I}(\text{Irampref[0]}) \\ \mathbf{I}(\text{Icompref[0]}) \end{pmatrix} = \begin{pmatrix} 1.17 & 1.22 \\ 4.39 & 4.21 \end{pmatrix} \begin{pmatrix} 1 \\ \text{cfg\_I} \end{pmatrix} [\mu\text{A}] \tag{4.4}$$

when cfg_halfI equals to '0', and

$$\begin{pmatrix} \mathbf{I}(\text{Irampref[0]}) \\ \mathbf{I}(\text{Icompref[0]}) \end{pmatrix} = \begin{pmatrix} 0.59 & 0.61 \\ 2.51 & 2.40 \end{pmatrix} \begin{pmatrix} 1 \\ \text{cfg\_I} \end{pmatrix} [\mu\text{A}] \tag{4.5}$$

when cfg_halfI equals to '1'.[1] Note that the '1' in the column vector of inputs accounts for the offset of the outputs. The results show only half of the outputs since Irampref[1] is identical to Irampref[0] and so is Icompref[1] to Icompref[0]. The statistics of the linear regression are shown in Table 4.3. The R^2's are close to 1, which

[1] In this example, cfg_I is represented as a single number. However, the implemented tool checks the gain from each bit of such digital bus input. The reason for this bitwise comparison is explained at the end of this section and Section 4.2.

Mode	R^2
cfg_halfI = 0	$\begin{pmatrix} 0.999 \\ 0.999 \end{pmatrix}$
cfg_halfI = 1	$\begin{pmatrix} 0.999 \\ 0.999 \end{pmatrix}$

Table 4.3: Linear regression statistics of the bias generator: with the circuit netlist.

means that the responses fit well into linear models.

In addition to the original Verilog model which had an error, a few Verilog models with errors are manually created to demonstrate the comparison. All models are tested with the same test vector. The checking results are summarized Table 4.4. Several problems of the original Verilog model (Model 1) are detected by observing the relative errors in partial gains. First, the offset current of the output was not included in the model. Second, the partial gain from cfg_I to Icompref[0] does not change for different modes configured by cfg_halfI. Lastly, the sign of G_{22}, i.e., the partial gain from cfg_I to **I**(Icompref[0]), is different from that of the circuit netlist because of the polarity inversion of the output current.[2]

After checking the original model, the model is revised manually by referring to the gain matrix of the circuit netlist. The system model of the revised Verilog model (Model 2) matches well with that of the circuit netlist.

Verilog models are further modified from the revised model. In Model 3, the gain matrices of two modes are swapped since the polarity of cfg_halfI is inverted. The error in Model 4 and Model 5 is less obvious than the other cases, but the R^2's of

[2]Since it is possible that transmitting and receiving blocks of the current are designed by different designers, the polarity convention of the current is important. For example, a current source with PMOS device in a transmitting block is possibly connected to a diode-connected PMOS load in a receiving block, which causes an electrical error. Thus we assume that the current flowing out of a PMOS device is positive and the current flowing into a NMOS device is negative in sign. When measuring the current in the checker, the current measurement module for a current sink, i.e., the output of a circuit connected to a NMOS current source, changes the measured current. Thus all the measured currents should have positive sign in the checker.

Model	Mode	Gain Matrix	Relative Error of partial gains in [%] to Circuit	$\min(R^2)$
Model 1: Original Verilog	cfg_halfI=0	$\begin{pmatrix} 0.0 & 1.0 \\ 0.0 & -4.0 \end{pmatrix}$	$\begin{pmatrix} -100 & -18 \\ -100 & -195 \end{pmatrix}$	1.0
	cfg_halfI=1	$\begin{pmatrix} 0.0 & 0.5 \\ 0.0 & -4.0 \end{pmatrix}$	$\begin{pmatrix} -100 & -18 \\ -100 & -267 \end{pmatrix}$	1.0
Model 2: Revised Verilog	cfg_halfI=0	$\begin{pmatrix} 1.17 & 1.22 \\ 4.21 & 4.39 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ -4 & 4 \end{pmatrix}$	1.0
	cfg_halfI=1	$\begin{pmatrix} 0.59 & 0.61 \\ 2.11 & 2.20 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 \\ -16 & -8 \end{pmatrix}$	1.0
Model 3: cfg_halfI is inverted	cfg_halfI=0	$\begin{pmatrix} 0.59 & 0.61 \\ 2.11 & 2.20 \end{pmatrix}$	$\begin{pmatrix} -50 & -50 \\ -51 & -48 \end{pmatrix}$	1.0
	cfg_halfI=1	$\begin{pmatrix} 1.17 & 1.22 \\ 4.39 & 4.21 \end{pmatrix}$	$\begin{pmatrix} 190 & 100 \\ 75 & 75 \end{pmatrix}$	1.0
Model 4: cfg_I is thermometer coded	cfg_halfI=0	$\begin{pmatrix} 1.35 & 0.78 \\ 4.84 & 2.83 \end{pmatrix}$	$\begin{pmatrix} 15 & -36 \\ 10 & -32 \end{pmatrix}$	0.929
	cfg_halfI=1	$\begin{pmatrix} 0.67 & 0.39 \\ 2.42 & 1.41 \end{pmatrix}$	$\begin{pmatrix} 14 & -36 \\ -4 & -41 \end{pmatrix}$	0.929
Model 5: cfg_I is thermometer coded and cfg_halfI is inverted	cfg_halfI=0	$\begin{pmatrix} 0.67 & 0.39 \\ 2.42 & 1.41 \end{pmatrix}$	$\begin{pmatrix} -43 & -68 \\ -45 & -67 \end{pmatrix}$	0.929
	cfg_halfI=1	$\begin{pmatrix} 1.35 & 0.78 \\ 4.84 & 2.83 \end{pmatrix}$	$\begin{pmatrix} 129 & 28 \\ 93 & 18 \end{pmatrix}$	0.929

Table 4.4: Model checking results of the bias generator.

Model	Mode	Gain Matrix	Relative Error of partial gains in [%] to Circuit
Circuit	cfg_halfI=0	$\begin{pmatrix} 1.17 & 2.43 & 1.21 \\ 4.39 & 8.40 & 4.20 \end{pmatrix}$	-
	cfg_halfI=1	$\begin{pmatrix} 0.58 & 1.21 & 0.61 \\ 2.50 & 4.79 & 2.40 \end{pmatrix}$	-
Model 4	cfg_halfI=0	$\begin{pmatrix} 1.17 & 1.22 & 1.22 \\ 4.21 & 4.39 & 4.39 \end{pmatrix}$	$\begin{pmatrix} 0 & -50 & 0 \\ -4 & -48 & 5 \end{pmatrix}$
	cfg_halfI=1	$\begin{pmatrix} 0.59 & 0.61 & 0.61 \\ 2.11 & 2.20 & 2.20 \end{pmatrix}$	$\begin{pmatrix} 2 & -50 & 0 \\ -16 & -54 & -8 \end{pmatrix}$

Table 4.5: Bitwise representation of system models for the circuit and Model 4 in Table 4.4.

the two cases become lower, which means that the responses fit less well into linear models. Moreover, if the partial gain from cfg_I is expanded to cfg_[1] and cfg_[0], the error becomes obvious. For example, Table 4.5 shows the bitwise gain matrices of both the circuit netlist and Model 4 with the following relationship:

$$\begin{pmatrix} \mathbf{I}(\text{Irampref}[0]) \\ \mathbf{I}(\text{Icompref}[0]) \end{pmatrix} = \mathbf{G} \begin{pmatrix} 1 \\ \text{cfg\_I}[1] \\ \text{cfg\_I}[0] \end{pmatrix} [\mu\text{A}] \tag{4.6}$$

As shown in Table 4.5, the bitwise comparison of partial gains to quantized analog input clearly shows that the model errors are about -50%.

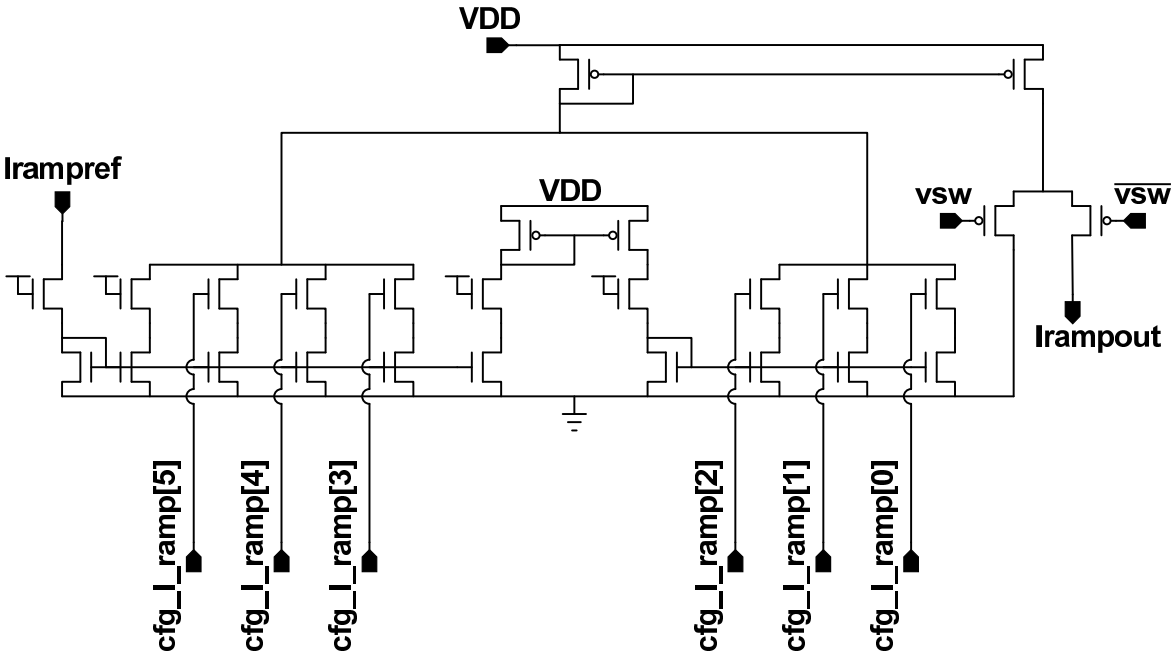


Figure 4.4: Ramp current generator.

4.2 Ramp Current Generator

The next design is a ramp current generator shown in Figure 4.4. When the input sampler is in hold mode, this ramp generator draws a current into the hold capacitor in the input sampler to ramp up the voltage across the capacitor. The circuit takes a reference current from the bias generator, and outputs a current to the input sampler. The ramp current needs to be finely controlled because the gain of the ADC transfer curve is primarily determined by the capacitance of the hold capacitor divided by this ramp current. At the same time, the area overhead of the circuit is also a concern because the circuit is replicated for every sub-ADC. To provide a wide range with moderate area, a multistage current mirror circuit is used [48].

The reference current from Irampref input is amplified to produce currents of the first and second current mirror stages, which are combined together to generate the ramp current. The output currents of both mirror stages are digitally controlled by cfg_I_ramp [5: 3] and cfg_I_ramp [2: 0], respectively. The current flow to the input sampler is then controlled by the digital input vsw and its complementary input $\overline{\text{vsw}}$.

Pin Name	Signal Type	I/O	Bit Width	Description
Irampout	Analog	Output	-	Ramp current output
Irampref	Analog	Input	-	Reference current input
VDD	Analog	Input	-	Power supply voltage input
cfg_I_ramp	Digital	Input	6	Adjust ramp current
vsw	Digital	Input	1	Enable ramp current output
$\overline{\text{vsw}}$	Digital	Input	1	Complementary input of vsw

Table 4.6: Physical pin description of the ramp current generator.

Physical pins of the circuit are summarized in Table 4.6.

Test Description

The ramp generator is linear in the current domain of the output Irampout for a similar reason as the bias generator is. However, in contrast to the bias generator, this circuit has an analog input **I**(Irampref) that propagates to the output while the current gain is controlled by two current mirror stages. This relationship is given by

$$\mathbf{I}(\text{Irampout}) = \text{vsw} \cdot G \cdot \mathbf{I}(\text{Irampref}) \tag{4.7}$$

$$G = \sum_{k=0}^{5} \alpha_k \cdot \text{cfg\_I\_ramp[k]} + g_0 \tag{4.8}$$

where G is the current gain of this circuit, α_k is the partial gain from each bit of cfg_I_ramp, and g_0 is the offset term of G.

The system is a tunable circuit, which could be nonlinear from Equation 4.7 and 4.8. Thus the system is decoupled into two coupled linear systems: a linear system from analog input **I**(Irampref) to analog output **I**(Irampout) and the other linear system from control input cfg_I_ramp to pseudo output (G) which is the gain of the first system. Therefore, two tests are necessary to check both coupled linear systems.

Port Name	Properties		
	Port Type	Domain	Range
Irampout	Analog Output	Current	[0.0,10µ] [A]
Irampref	Analog Input	Current	[0.0,10µ] [A]

Port Name	Properties			
	Port Type	Bit Width	Encoding	Prohibited
cfg_I_ramp	Quantized Analog Control Input	6	binary	None
vsw	True Digital Input	1	binary	None
$\overline{\text{vsw}}$	True Digital Input	1	binary	None

Table 4.7: Port classification of the ramp current generator: For testing the system in Equation 4.7, cfg_I_ramp is randomly sampled (e.g., '100000') in its valid values. For testing the system in Equation 4.8, **I**(Irampref) is randomly sampled (e.g., 2.0 µA) in its valid values.

Table 4.7 summarizes the ports for testing the systems. Note that G is indirectly measured by observing **I**(Irampout) rather than calculated by post-processing script.[3] Given this setup, coupled linear systems used in the checking are given by

$$\mathbf{I}(\text{Irampout})_1 = \text{vsw} \cdot G_{\text{cfg\_I\_ramp=constant}} \cdot \mathbf{I}(\text{Irampref}) \tag{4.9}$$

$$\mathbf{I}(\text{Irampout})_2 = \text{vsw} \cdot G \cdot \mathbf{I}(\text{Irampref})_{\text{Irampref=constant}} \tag{4.10}$$

It is also noteworthy that the checker fails to perfectly recognize the quantized port

[3]Given that **I**(Irampref) is pinned, the change in G to cfg_I_ramp can be indirectly measured by observing **I**(Irampout). As long as one is interested in comparing two analog models, not extracting the parameters, this indirect measurement gives the same checking result.

cfg_I_ramp by pattern matching due to the irregular circuit structure. The tool successfully identifies the quantized ports for two subsets of cfg_I_ramp, cfg_I_ramp [5: 3] and cfg_I_ramp [2: 0], because each subset has a regular structure, but it fails to recognize the quantized port for the whole bits of cfg_I_ramp [5: 0] because these two groups are not regular to each other. However, this does not increase the number of test vectors since two subsets of cfg_I_ramp still contribute to G linearly.

Model Checking Result

Linear system models are extracted for two modes created by the true digital input vsw, which are given by

$$\begin{pmatrix} \mathbf{I}(\text{Irampout}) \\ G_{\mathbf{I}(\text{Irampref})=2\ \mu\text{A}} \end{pmatrix} = \begin{pmatrix} 0 & 0 & - \\ 0 & - & 0 \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{I}(\text{Irampref}) \\ \text{cfg\_I\_ramp} \end{pmatrix} \begin{bmatrix} \mu\text{A} \\ \text{A/A} \end{bmatrix} \tag{4.11}$$

when vsw equals to '0', and

$$\begin{pmatrix} \mathbf{I}(\text{Irampout}) \\ G_{\mathbf{I}(\text{Irampref})=2\ \mu\text{A}} \end{pmatrix} = \begin{pmatrix} 0 & 1.24 & - \\ 0.99 & - & 0.013 \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{I}(\text{Irampref}) \\ \text{cfg\_I\_ramp} \end{pmatrix} \begin{bmatrix} \mu\text{A} \\ \text{A/A} \end{bmatrix} \tag{4.12}$$

when vsw is '1'. When vsw equals to '1', the R^2 of the linear regression is $\begin{pmatrix} 1.0 & 0.985 \end{pmatrix}^T$.

In testing quantized analog ports, bitwise comparison of the partial gains is important. This is not only because the error in the model becomes more obvious as exemplified in Section 4.1, but also the error caused by a single bit of a quantized analog port can be smeared out if the response change contributed by the bit is small compared with the residual errors of fitted model. Consider the case of a Verilog model where the least significant bit (LSB) of cfg_I_ramp is inverted. If one runs

linear regression on $G_{\mathbf{I}(\text{Irampref})=2\ \mu\text{A}}$ to cfg_I_ramp, its partial gains are 0.0132 A/A for the circuit netlist and 0.0131 A/A for the Verilog model, respectively. The relative error is only about -1% such that it appears to have no error in the model. The reason for this failure is that the residual errors of the linear regression on the circuit response are much larger than the response change contributed by the LSB cfg_I_ramp [0]; the standard deviation of the residual errors is 0.037 A/A, which is much larger than the partial gain from the LSB, 0.0132 A/A.

Comparing the partial gains from each bit of cfg_I_ramp makes the error detectable. The following two equations show the system equations $G_{\mathbf{I}(\text{Irampref})=2\ \mu\text{A}}$ of both the netlist and the model in order when vsw is set to '1'.

$$G_{Circuit,\mathbf{I}(\text{Irampref})=2\ \mu\text{A}} = \begin{pmatrix} 0.949 \\ 0.027 \\ 0.052 \\ 0.102 \\ 0.097 \\ 0.194 \\ 0.387 \end{pmatrix}^T \begin{pmatrix} 1 \\ \text{cfg\_I\_ramp}\,[0] \\ \text{cfg\_I\_ramp}\,[1] \\ \text{cfg\_I\_ramp}\,[2] \\ \text{cfg\_I\_ramp}\,[3] \\ \text{cfg\_I\_ramp}\,[4] \\ \text{cfg\_I\_ramp}\,[5] \end{pmatrix} [\text{A/A}] \tag{4.13}$$

$$G_{Verilog,\mathbf{I}(\text{Irampref})=2\ \mu\text{A}} = \begin{pmatrix} 0.975 \\ -0.027 \\ 0.052 \\ 0.102 \\ 0.097 \\ 0.194 \\ 0.387 \end{pmatrix}^T \begin{pmatrix} 1 \\ \text{cfg\_I\_ramp}\,[0] \\ \text{cfg\_I\_ramp}\,[1] \\ \text{cfg\_I\_ramp}\,[2] \\ \text{cfg\_I\_ramp}\,[3] \\ \text{cfg\_I\_ramp}\,[4] \\ \text{cfg\_I\_ramp}\,[5] \end{pmatrix} [\text{A/A}] \tag{4.14}$$

In this case, the standard deviation of the residual errors is lowered to 0.003 A/A. From these extracted equations, one is able to easily detect that the polarity of the

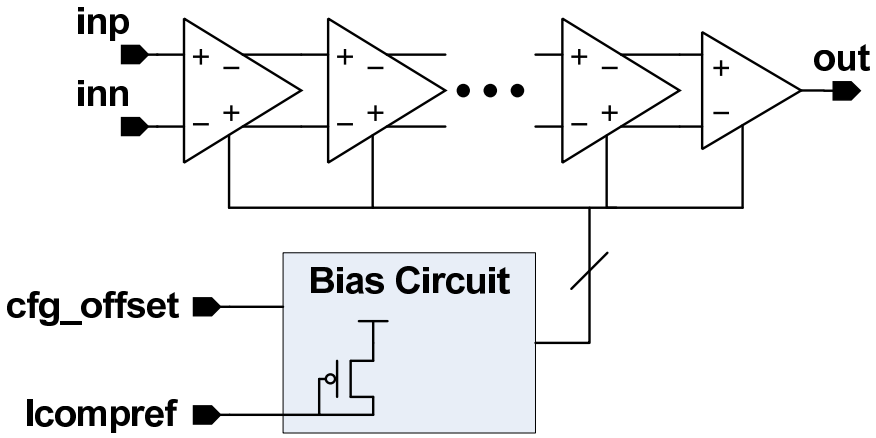


Figure 4.5: Block diagram of a comparator.

partial gain from cfg_I_ramp [0] is inverted. Also, the bitwise gain matrices show that cfg_I_ramp is not a single binary code, and this is the reason why the standard deviation of the residual errors is lowered when performing linear regression with individual bits of cfg_I_ramp as predictor variables. In Equation 4.13, the partial gain from cfg_I_ramp [2] and the gain from cfg_I_ramp [3] are close because the size of current mirror devices contributing to these partial gains are the same.

4.3 Comparator

A simplified block diagram of a comparator is shown in Figure 4.5. It compares the differential voltage inputs from two input samplers, triggering a latch to hold the current value of a global counter. The comparator in this example is a continuous-time, multistage amplifier, which compares two analog voltage inputs, inp and inn, and outputs the digital signal out. The bias circuit in the comparator takes a reference current Icompref from the bias generator, generating and distributing bias signals to the amplifiers. The digital input cfg_offset adjusts the input offset voltage of the circuit. Table 4.8 summarizes physical pins.

Pin Name	Signal Type	I/O	Bit Width	Description
out	Digital	Output	1	'1' if $\mathbf{V}(inp) \geq \mathbf{V}(inn)$, else '0'
inp	Analog	Input	-	Positive-going voltage input
inn	Analog	Input	-	Negative-going voltage input
Icompref	Analog	Input	-	Reference current input
VDD	Analog	Input	-	Power supply voltage input
cfg_offset	Digital	Input	3	Adjust input offset voltage

Table 4.8: Physical pin description of the comparator.

Test Description

Testing a comparator is difficult because its output is quantized. The discrete response makes it impossible to directly measure the linear intent of the circuit. Instead of directly observing the digital output, a virtual feedback network is implemented to convert the digital signal output to an analog signal as shown in Figure 4.6. It is essentially a unity gain feedback. The digital output is converted into a small voltage step by 1-bit, D/A converter which is clocked by a virtual clock. The feedback would be stuck at infinite loop in simulating a Verilog model unless the D/A is clocked. To make the loop negative feedback and pass through an integrator, the user sets the scale factor of the D/A output to either -1 if the comparator is non-inverting, or +1 if the comparator is inverting in the test setup. If the user incorrectly sets up the scale factor, the ouput is stuck at a value and thus the output response does not fit well to a linear model. To speed up the convergence of the loop, the gain of the D/A converter is adaptively controlled.[4]

The linear response of the circuit is now exposed with the virtual output vout, which is depicted in Figure 4.6b. The analog voltage output $\mathbf{V}$(vout) is a linear function of analog voltage inputs, $\mathbf{V}$(inp) and $\mathbf{V}$(inn), making it easy to analyze the

[4]This virtual feedback loop is written in HDLs, and its SystemVerilog example is listed in Appendix A.

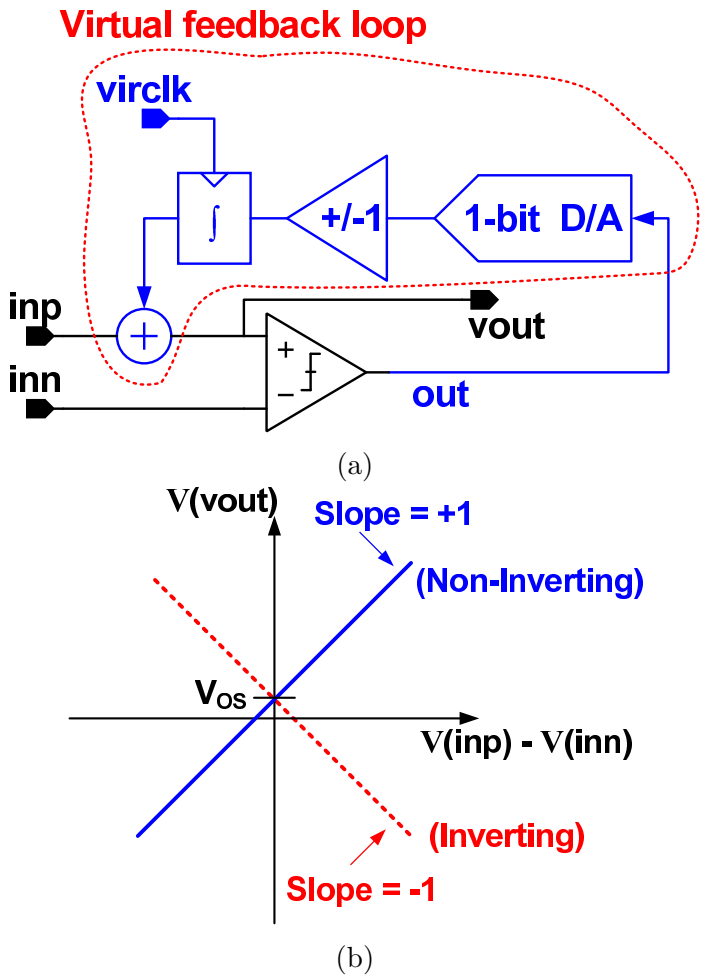


Figure 4.6: Testing a comparator: (a) test circuit configuration and (b) transfer curve.

system. The slope of the transfer curve shows whether the polarity from the inputs to the output is inverted or not, and the y-intercept, i.e. the point at which the line crosses the $\mathbf{V}$(vout)-axis, is the input offset voltage V_{OS} of the circuit.

Table 4.9 summarizes ports used in testing the comparator. Note that the digital output pin *out* is not the port of the circuit in this linear system model. Instead, $\mathbf{V}$(inp) and $\mathbf{V}$(inn) are analog input ports of the system and $\mathbf{V}$(vout) is the system's analog output port. Also, the input offset voltage adjusted by cfg_offset is extracted together with analog I/O ports since the quantized analog input cfg_offset and the analog inputs independently contribute to the analog output.

Port Name	Properties		
	Port Type	Domain	Range
vout	Analog Output	Voltage	[-0.4,0.4] [V]
inp	Analog Input	Voltage	[0.0,0.3] [V]
inn	Analog Input	Voltage	[0.0,0.3] [V]
Icompref	Analog Function Input	Current	[7,7] [µA]
VDD	Analog Function Input	Voltage	[1.0,1.0] [V]

Port Name	Properties			
	Port Type	Bit Width	Encoding	Prohibited
cfg_offset	Quantized Analog Input	3	Binary	None

Table 4.9: Port classification of the comparator.

Note that two other analog inputs, VDD and Icompref are not tested in the example since the effect of these two inputs are not modeled in the original Verilog model. Input signals on VDD and Icompref ports are set to their typical values in testing the circuit, working as function ports of the system.

Model Checking Result

In checking this comparator, two functions are of interest; one is whether the polarity from two inputs, inp and inn, to the output out is correct, and the other is how the input offset voltage is adjusted with the quantized analog port cfg_offset.

Model	Gain Matrix	Relative Error of partial gains in [%] to Circuit	$\min(R^2)$
Model 1: Original Verilog	$\begin{pmatrix} 0.022 & -0.006 & 0.961 \end{pmatrix}$	$\begin{pmatrix} 4 & 0 & 0 \end{pmatrix}$	0.996
Model 2: Inverting output	$\begin{pmatrix} 0.0 & 0.0 & -1.285 \end{pmatrix}$	$\begin{pmatrix} -100 & -100 & -234 \end{pmatrix}$	0.741
Model 3: cfg_offset is inverted	$\begin{pmatrix} -0.022 & 0.006 & 0.961 \end{pmatrix}$	$\begin{pmatrix} -200 & 203 & 0 \end{pmatrix}$	0.996

Table 4.10: Model checking results of the comparator.

The extracted linear system equation of the circuit netlist is given by

$$\mathbf{V}(\text{vout}) = \mathbf{Gx} = \begin{pmatrix} 0.021 & -0.006 & 0.962 \end{pmatrix} \begin{pmatrix} 1 \\ \text{cfg\_offset} \\ \mathbf{V}(\text{inp}) - \mathbf{V}(\text{inn}) \end{pmatrix} [\text{V}] \qquad (4.15)$$

where the R^2 of the fitted model is 0.996. The relationship from the analog inputs to the analog output is non-inverting because G_3 is close to +1. Also, G_2 represents the gain of the controlled input offset voltage.

Two incorrect Verilog models are created and checked against the circuit netlist. One model is producing an inverting digital output, and the other model adjusts the input offset voltage by cfg_offset in the opposite direction to what the circuit does. As shown in Table 4.10, the errors can be detected by comparing the gain matrices. In testing Model 2, the extraction results in an abnormal gain matrix since the test assumed that the comparator should be non-inverting such that the scale factor after the D/A converter is set to -1, which causes the virtual feedback loop in Figure 4.6a to be positive feedback in this incorrect Verilog model. The input offset voltage is not affected by cfg_offset since the output $\mathbf{V}(\text{vout})$ is already stuck because of the

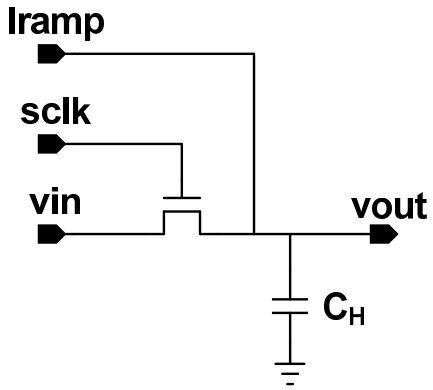


Figure 4.7: Input sampler.

positive feedback loop. This abnormal result can also be identified by observing the low R^2 value which means that the response does not fit well into a linear model.

4.4 Input Sampler

The analog input of the ADC is sampled on the hold capacitor using the circuit shown in Figure 4.7. For each sub-ADC, two identical samplers form a pseudo-differential circuit, which first tracks and then holds the differential input signal. The track and hold (T/H) modes are controlled by a clock input sclk. The output voltage vout tracks the input voltage vin when sclk is '1' while vin is disconnected from vout when sclk goes to '0'. When the circuit is in hold mode, a constant current from the ramp generator is drawn through the Iramp input to the hold capacitor C_H, so that vout voltage is ramping up for the rest of a conversion cycle. Table 4.11 summarizes physical pins.

Test Description

Unlike other circuits examined, this circuit has states due to the hold capacitor. In addition to checking the transfer function of the circuit in both states, we also need to check the state transition when sclk changes. To verify the state transition as well, a slight modification in driving the sclk input is necessary. For other circuits, we treated a true digital input as a static input, i.e., a circuit is already configured by

Pin Name	Signal Type	I/O	Bit Width	Description
vout	Analog	Output	-	voltage output
vin	Analog	Input	-	voltage input
sclk	Digital	Input	1	the sampler is in track mode if sclk=''1', and in hold mode if sclk='0'
Iramp	Analog	Input	-	Ramp current input

Table 4.11: Physical pin description of the input sampler.

true digital inputs before testing a circuit or a model. However, sclk in this circuit is triggered during simulations while all possible state transitions are enumerated. Thus one test is needed to check a linear system between analog I/O ports when sclk switches from '0' to '1', and another test is needed to check the other linear system between ports when sclk changes from '1' to '0'.

It is also interesting that this circuit has *time* as an analog input or an analog control input, coupled with the current flowing through Iramp input because the ramp-up voltage across the hold capacitor is a product of the input current and the duration of the current integration. Therefore, the linear system models of this circuit are given by

$$\mathbf{V}(\text{vout}) = \mathbf{V}(\text{vin}) \tag{4.16}$$

when the circuit is in track mode,[5] and

$$\mathbf{V}(\text{vout}) = \mathbf{I}(\text{Iramp}) \cdot \delta t + \alpha \cdot \mathbf{V}(\text{voutp}) + V_{feed} \tag{4.17}$$

when the circuit is in hold mode. V_{feed} is an offset voltage caused by clock feedthrough when the circuit switches to hold mode. The charge injection error depends on the

[5]We assume that the output impedance of driving source for vin is very low such that the output voltage $\mathbf{V}$(vout) in track mode is not affected by the ramp current although the sampling switch has finite impedance.

previous state of the voltage on vout ($\mathbf{V}$(voutp)), which is the tracked voltage by the sampler. Table 4.12 summarizes the ports for checking models. It is noteworthy that the system is decoupled into two coupled systems by assuming that δt is an analog control port.

Model Checking Result

The extracted linear system equations of the circuit netlist are given by

$$\begin{pmatrix} \mathbf{V}(\text{vout})_1 \\ \mathbf{V}(\text{vout})_2 \end{pmatrix} = \begin{pmatrix} -27.4\text{m} & 0.0 & 87.1\text{k} & - & 9.97\text{m} \\ -23.1\text{m} & - & - & 104.1\text{M} & - \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{V}(\text{vin}) \\ \mathbf{I}(\text{Iramp}) \\ \delta t \\ \mathbf{V}(\text{voutp}) \end{pmatrix} [\text{V}] \quad (4.18)$$

when sclk changes from '1' to '0', and

$$\begin{pmatrix} \mathbf{V}(\text{vout})_1 \\ \mathbf{V}(\text{vout})_2 \end{pmatrix} = \begin{pmatrix} 0.0 & 1.0 & 0.0 & - & 0.0 \\ 0.15 & - & - & 0.0 & - \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{V}(\text{vin}) \\ \mathbf{I}(\text{Iramp}) \\ \delta t \\ \mathbf{V}(\text{voutp}) \end{pmatrix} [\text{V}] \quad (4.19)$$

when sclk changes from '0' to '1'. The first row, the output is $\mathbf{V}(\text{vout})_1$, of the matrices is the system between analog I/Os while δt is set to 2.5 ns, and the second row, the output is $\mathbf{V}(\text{vout})_2$, is the indirect measurement of the system's gain by observing $\mathbf{V}$(vout) to δt while $\mathbf{I}$(Iramp) and $\mathbf{V}$(vin) are set to 3.0 µA and 0.15 V, respectively.

The system equations of a Verilog model is summarized in Table 4.13. The result shows that the Verilog model does not match with the circuit netlist as pedestal

Port Name	Properties		
	Port Type	Domain	Valid Range
vout	Analog Output	Voltage	[0.0,0.4] [V]
vin	Analog Input	Voltage	[0.0,0.3] [V]
voutp	Analog Input	Voltage	[0.0,0.4] [V]
Iramp	Analog Input	Current	[2.0,4.0] [µA]
δt	Analog Control Input	Time	[1,4] [ns]

Port Name	Properties			
	Port Type	Bit Width	Encoding	Prohibited
sclk	True Digital Input	1	Binary	None

Table 4.12: Port classification of the input sampler.

Model	Mode	Gain Matrix	Relative Error of partial gains in [%] to Circuit	$\min(R^2)$
Model 1: Original Verilog	sclk: '1' → '0'	$\begin{pmatrix} 0.0 & 0.0 & 87.9\text{k} & - & 0.0 \\ 0.0 & - & - & 105.4\text{M} & - \end{pmatrix}$	$\begin{pmatrix} -100 & 0 & 2 & - & -100 \\ -100 & - & - & 2 & - \end{pmatrix}$	1.0
	sclk: '0' → '1'	$\begin{pmatrix} 0.0 & 1.0 & 0.0 & - & 0.0 \\ 0.15 & - & - & 0.0 & - \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & - & 0 \\ 0 & - & - & 0 & - \end{pmatrix}$	1.0

Table 4.13: Model checking results of the input sampler.

error is not modeled in the Verilog model. The offset values, -27.4 mV and -23.1 mV, in Equation 4.18 caused by clock feedthrough are not considered in the Verilog model. The Verilog model does not include the charge injection error dependent on

the previous state of vout.

4.5 Phase Interpolator

The last example is the phase interpolator shown in Figure 4.8. It takes a clock and its quadrature clock as input signals. The output clock is generated by interpolating the phase of the two input clocks. The circuit is composed of a phase mixer, a capacitor array, and a buffer chain. The phase mixer consists of 12 identical copies of the unit cell shown in Figure 4.8b, and interpolates the phase of two incoming clocks, clki and clkq. The amount of phase interpolation is determined by the relative driving strength of the clki and clkq clock signals at the phase mixing node N_{MIX}. The driving strength is controlled by the 12-bit digital input sel_iq. Given that the clki phase leads the clkq phase, the phase of the interpolated clock is earlier if sel_iq has more '1's than '0's, and vice versa; sel_iq is a thermometer-coded bus signal. The rise time of the signal at N_{MIX} also affects the circuit performance. Thus the rise time is adjustable by a capacitor array with the 4-bit digital input sel_cap, making the circuit robust to process, temperature, and supply voltage variation. Finally, a buffer chain is used for driving a capacitive output load. Table 4.14 summarizes physical pins of the circuit.

Test Description

Testing the phase interpolator is split into two tests. One is to test a system from two clock inputs to the clock output with sel_iq. The other is to check the rise time of the signal at the node N_{MIX} with sel_cap.

The underlying linear system model of the first system is built in phase domain. The system model of the circuit is given by

$$\mathbf{\Phi}(\text{clko}) = \left(\sum_{k=0}^{11} \alpha_k \cdot \text{sel\_iq[k]} \right) \cdot (\mathbf{\Phi}(\text{clkq}) - \mathbf{\Phi}(\text{clki})) + \mathbf{\Phi_0} + \mathbf{\Phi}(\text{clki}) \tag{4.20}$$

where α_k is the partial gain from sel_iq[k], $\mathbf{\Phi_0}$ is the phase offset, and $\mathbf{\Phi}(\cdot)$ represents

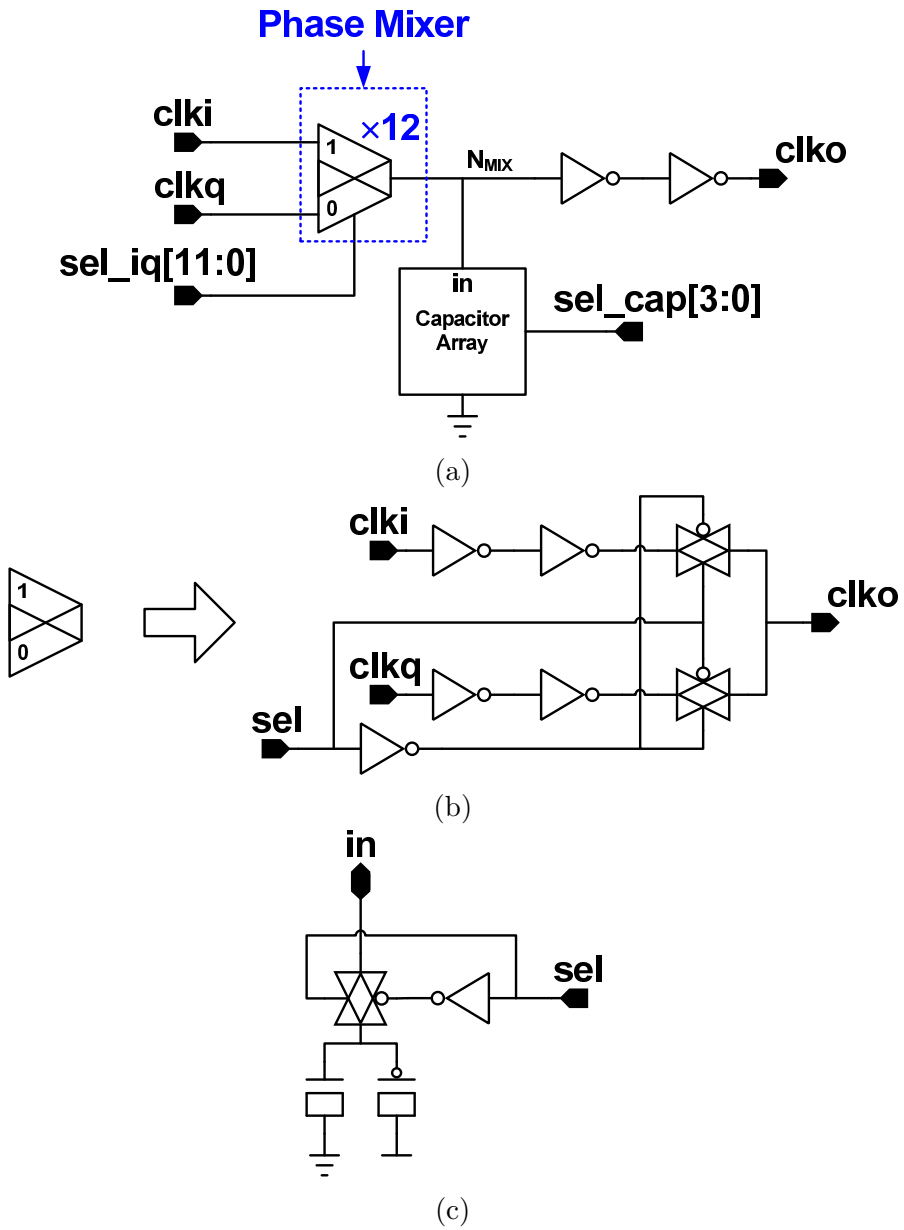


Figure 4.8: Phase interpolator: (a) Block diagram of a phase interpolator, (b) unit cell of the phase mixer shown in (a), and (c) unit cell of a capacitor array.

Pin Name	Signal Type	I/O	Bit Width	Description
clko	Digital	Output	-	Phase interpolated clock output
clki	Digital	Input	-	In-phase clock input
clkq	Digital	Input	-	Quadrature clock input
VDD	Analog	Input	-	Power supply voltage input
sel_iq	Digital	Input	12	Adjust clko phase
sel_cap	Digital	Input	4	Adjust rise time of internal signal

Table 4.14: Physical pin description of the phase interpolator.

a clock in phase domain. Since the inputs, $\mathbf{\Phi}$(clki) − $\mathbf{\Phi}$(clkq) and sel_iq, do not contribute to the output independently, this nonlinear system is decoupled into two coupled linear systems as follows.

$$\mathbf{\Phi}(\text{clko}) = G_1 \cdot (\mathbf{\Phi}(\text{clkq}) - \mathbf{\Phi}(\text{clki})) + \mathbf{\Phi_0} + \mathbf{\Phi}(\text{clki}) \tag{4.21}$$

$$G_1 = \sum_{k=0}^{11} \alpha_k \cdot \text{sel\_iq[k]} \tag{4.22}$$

These coupled linear systems require two tests. One is to check the relationship between analog I/Os, i.e., $\mathbf{\Phi}$(clki), $\mathbf{\Phi}$(clkq), and $\mathbf{\Phi}$(clko) in Equation 4.21. Another test is to validate how the pseudo output G_1, i.e., the amount of phase interpolation, is varying with quantized analog input sel_iq.

The last test is to check the rise time of the internal node N_{MIX} to sel_cap input. Since N_{MIX} is not observable, the delay (phase offset) of the circuit, $\mathbf{\Phi_0}$ in Equation 4.21, is measured instead since the change of the rise time affects the propagation delay through the circuit. This relationship is given by

$$\mathbf{\Phi_0} = \sum_{k=0}^{3} \beta_k \cdot \text{sel\_cap[k]} + \phi_{\mathbf{0}} \tag{4.23}$$

Port Name	Properties		
	Port Type	Domain	Valid Range
clko	Analog Output	Phase	[-π,π] [rad]
clki	Analog Input	Phase	[0.0,0.0] [rad]
clkq	Analog Input	Phase	[$-\frac{1}{4}\pi$,$\frac{1}{4}\pi$] [rad]

Port Name	Properties			
	Port Type	Bit Width	Encoding	Prohibited
cfg_iq	Quantized Analog Control Input	12	thermometer	None
cfg_cap	Quantized Analog Control Input	4	thermometer	None

Table 4.15: Port classification of the phase interpolator: 1) sel_iq and sel_cap are randomly sampled in valid values when testing the system in Equation 4.21, 2) $\mathbf{\Phi}$(clkq) is set to $\frac{1}{4}\pi$ and sel_cap is randomly sampled in valid values when testing the system in Equation 4.22, and 3) $\mathbf{\Phi}$(clki), $\mathbf{\Phi}$(clkq), and sel_iq are randomly sampled in valid values when testing the system in Equation 4.23.

where β_k is the partial gain from sel_cap[k] and $\phi_{\mathbf{0}}$ is the intrinsic phase offset. The port specifications for testing the above three tests are summarized in Table 4.15.

Model Checking Result

The system equations of the phase interpolator are extracted with the implemented checker tool. When testing the circuit, $\mathbf{\Phi}(\cdot)$ domain requires frequency information of a clock, an implicit variable to the domain translator. Thus the frequency of all clocks are set to 5 GHz which is the system specification of the ADC. Also, the phase spacing between clki and clkq is set to 50 ps, quadrature phase difference of a 5 GHz clock when extracting G_1 and $\mathbf{\Phi_0}$.

Model	Gain Matrix	Relative Error of partial gains in [%] to Circuit	$\min(R^2)$
Model 1: Original Verilog model	$\begin{pmatrix} 3.02 & 0.50 & - & - \\ 4.89 & - & -0.17 & - \\ 3.66 & - & - & 0.06 \end{pmatrix}$	$\begin{pmatrix} 29 & 0 & - & - \\ 21 & - & -4 & - \\ 25 & - & - & 51 \end{pmatrix}$	0.937
Model 2: sel_iq is inverted	$\begin{pmatrix} 3.021 & 0.500 & - & - \\ 2.91 & - & 0.17 & - \\ 3.66 & - & - & 0.06 \end{pmatrix}$	$\begin{pmatrix} 29 & 0 & - & - \\ -28 & - & 204 & - \\ 25 & - & - & 51 \end{pmatrix}$	0.937
Model 3: sel_iq is binary coded	$\begin{pmatrix} 2.58 & 0.50 & - & - \\ 5.29 & - & -0.11 & - \\ 3.03 & - & - & -0.01 \end{pmatrix}$	$\begin{pmatrix} 11 & 0 & - & - \\ 31 & - & 28 & - \\ 4 & - & - & -115 \end{pmatrix}$	0.333

Table 4.16: Model checking results of the phase interpolator.

Note that G_1 and $\mathbf{\Phi_0}$ are indirectly inspected by measuring $\mathbf{\Phi}(\text{clko})$. Thus the output responses measured for the model checking are labeled as $\mathbf{\Phi}(\text{clko})_1$, $\mathbf{\Phi}(\text{clko})_2$, and $\mathbf{\Phi}(\text{clko})_3$ for the three tests, respectively. Given this setup, the extracted linear system equation of a circuit netlist is given by

$$\begin{pmatrix} \mathbf{\Phi}(\text{clko})_1 \\ \mathbf{\Phi}(\text{clko})_2 \\ \mathbf{\Phi}(\text{clko})_3 \end{pmatrix} = \begin{pmatrix} 2.33 & 0.50 & - & - \\ 4.04 & - & -0.16 & - \\ 2.92 & - & - & 0.09 \end{pmatrix} \begin{pmatrix} 1 \\ \mathbf{\Phi}(\text{clkq}) - \mathbf{\Phi}(\text{clki}) \\ \text{sel\_iq} \\ \text{sel\_cap} \end{pmatrix} \tag{4.24}$$

where sel_iq and sel_cap are variables encoded as specified in the port description. For example, sel_cap is thermometer-coded such that a binary vector '0011' is encoded to a decimal number '2' when running linear regression. The R^2 of the linear regression is $\begin{pmatrix} 0.994 & 0.947 & 0.926 \end{pmatrix}^T$ due to weakly nonlinear behavior of the circuit.

In addition to the original Verilog model, a few Verilog models with errors are created manually for the comparison, which are tested with the same test vector run for the circuit netlist. The checking results are summarized Table 4.16.

An invalid test setup can be detected by observing R^2 and p-value of partial gains. For example, an experiment is performed to measure $\mathbf{\Phi}$(clko) of the circuit netlist without decoupling the systems in Equation 4.20, i.e., a single test is run with varying $\mathbf{\Phi}$(clkq) − $\mathbf{\Phi}$(clki) and sel_iq together. In the experiment, the response does not fit well into a linear model. The linear regression result shows that the R^2 is 0.916, but the partial gain from sel_iq is statistically insignificant; p-value is 0.613 which is much higher than 0.05.

Chapter 5

Process Variation in Mixed-Signal Systems

Monte Carlo simulation of a mixed-signal circuit is often performed to predict the distribution of the circuit performance due to process variations. However, Monte Carlo simulations of a system at the transistor level are very time consuming. Moreover, running the simulations at the system-level is extremely difficult because analog and digital subsystems should be validated together, but the digital subsystem in a chip contains billions of transistors. In this chapter, we discuss a way to run system-level Monte Carlo simulations using analog functional models for faster evaluation.

In Chapter 3, we showed how the linear abstraction of analog circuits enables the equivalence checking of analog functional models. The underlying linear intent of analog circuits is formally mapped into a gain matrix, and the checking is done by comparing the gain matrices of two analog models. In this chapter, we describe how to create a functional model with process variations by again leveraging the linear abstraction. With such functional models, one is able to rapidly run process corner simulations for worst-case analysis and system-level Monte Carlo simulations for parametric yield estimation.

When verifying a mixed-signal system with analog functional models, it is important to ensure that all input signals are within the range where the models were verified. This is checked during simulations by assertions written into the models.

As process parameters vary, the valid input range changes, leaving us with two options. One is to create assertions that hold for all process parameters, which yields a conservative bound. The other option is to parameterize the assertions with process parameters. Section 5.4 describes how one can use a support vector machine (SVM) to generate these parametric assertions.

5.1 Failure in Mixed-Signal Systems

Circuit performance varies as process parameters change. Sometimes, these process variations only change the overall parameters of the circuit such as gain and bandwidth, but there are times when these process variations can cause the circuit to functionally fail, especially if the circuit is linear in a transformed domain. For instance, a frequency divider that is linear in phase can suffer from this type of failure. If the delay of the divider with these process parameters is greater than the clock cycle time minus the flop timing overhead, the circuit will not divide the frequency correctly. This is a functional failure. On the other hand, it is called parametric failure when the circuit is still functioning properly, but some of the performance specifications are out of the normal operating range. For example, leakage power in standby mode may be beyond the specified limit because the threshold voltage of the transistors is too low.

The PLL shown in Figure 5.1 is used to illustrate both types of failures. If the output clock frequency of the voltage controlled oscillator (VCO) is too high for the frequency divider to operate in a divided-by-two mode, it might stop transitioning its output (or divide at a different ratio); in this case, since the feedback clock is too slow, the feedback loop tries to increase the frequency, pushing it further in the wrong direction.[1] This is often called a deadlock situation. This is a functional failure because the system operates in a completely different mode from the expected mode. On the other hand, if the up and down currents in the charge-pump circuit

[1]This could be understood as global convergence problem due to uncertain initial states of the system [49]. However, the scope in the thesis is limited to the case where it occurs because of the malfunction of circuit blocks due to process variation.

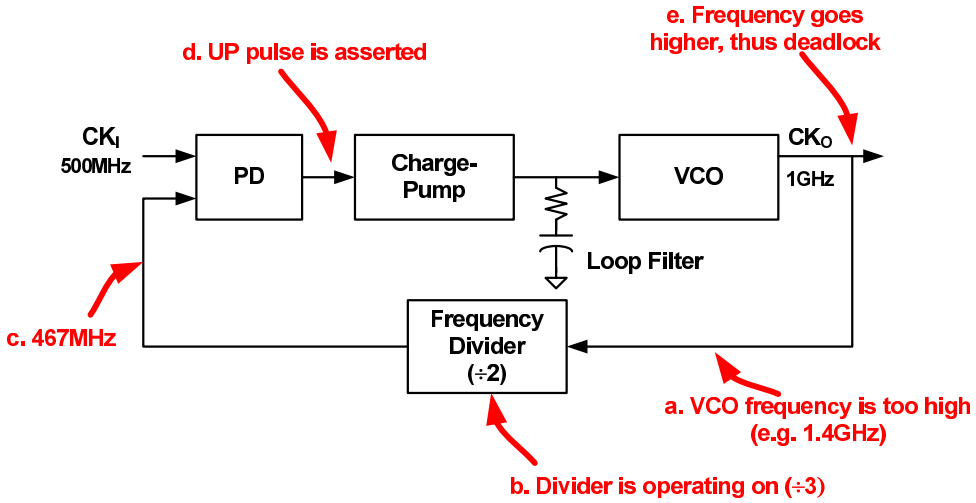


Figure 5.1: Deadlock example of a PLL.

are unequal due to device mismatches, the static phase offset of the system will be nonzero. While the system still operates as intended, it might fail to meet the system specification on the static phase offset; this would be a parametric failure.

Since the occurrence of these failures depends on the fabricated chip's process parameters, exploring a circuit over this space is of great interest. The following sections explain how we exploit our functional models to perform this analysis efficiently.

5.2 Process-Aware Analog Model

When creating a process-aware analog model, we handle functional failures separately from parametric failures. For parametric failures, the circuit's behavior is still specified by a gain matrix $\mathbf{G}$, so the effect of process variation can be mapped onto a variation in $\mathbf{G}$. On the other hand, since functional failures break the model, we need to find how this functional boundary depends on process parameters. The following subsections describe each of these two cases. The procedures for model creation and simulation are discussed in Section 5.3 and 5.4.

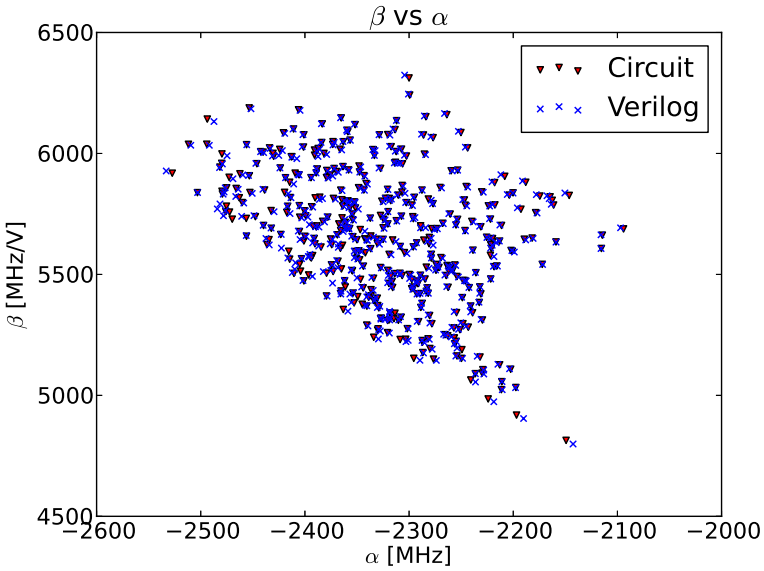


Figure 5.2: Partial gains of VCO linear system model by randomly sampling process parameters (Plot of both the circuit and the Verilog simulation results).

5.2.1 Parametric Model

Given that the underlying linear assumption of an analog circuit is valid, process variations only modify the properties of the circuit, rather than change its function. In other words, the effects of process variations are manifested as a change in a gain matrix **G**. When process parameters vary, the parametric variation of a gain matrix is likely to be smooth because the overall analog response is smooth. In this case, **G**'s dependence on process parameters can be fitted as a low order Taylor series such as a linear or quadratic function by extracting **G** from a few number of circuits with different process parameters. The remaining step is to create a Verilog model that corresponds to the given process parameters. Once the mapping function of functional parameters in a Verilog model onto the gain matrix is known from equivalence checking, it is an inverse problem to find the Verilog parameters which represent the circuit function (gain matrix) for a given set of process parameters. The detailed procedure is explained in Section 5.3.

For instance, the process variation of the VCO circuit shown in Figure 3.6 is

characterized to see how accurate the model is. The intended function of the circuit is to generate a clock output whose frequency f_{CLKo} is adjusted by the control voltage input $\mathrm{V_{CTRL}}$, and it is modeled as a linear function as follows.

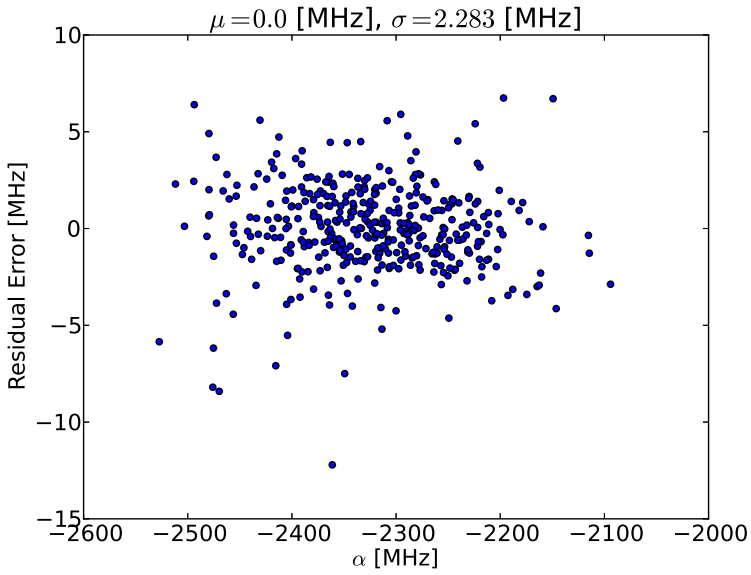
$$f_{\mathrm{CLKo}} = \alpha + \beta \cdot \mathrm{V_{CTRL}} \tag{5.1}$$

where α and β are partial gains of the system. With this linear model, the partial gains of the real circuit are then extracted by randomly sampling the process parameters. By running linear regression on the extracted partial gains with respect to the sampled process parameters, each partial gain is fitted to a third order Taylor series function of process parameters including the first order interaction term between the parameters.[2] With this parametric model, both circuit simulations and functional simulations are performed with the same set of randomly sampled process parameters.[3] As shown in Figure 5.2, the responses from the circuit and the model are well matched. To quantify the modeling error, the residual errors of the partial gains are plotted in Figure 5.3. The R^2 of α and β are 0.9998 and 0.9991, respectively.

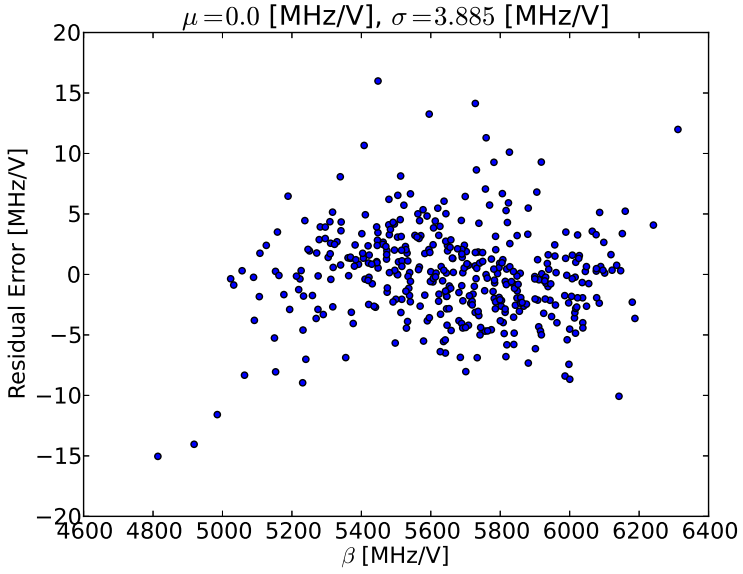
The way we model the process variation is very similar to the way that circuit performance is modeled using the response surface model and its variants [50–55]. However, the computational cost of our approach is not high. Since the conventional modeling is generally done by running end-to-end tests, the computational cost is high for a system with a large number of components. Moreover, it is very time consuming to run a system-level verification of mixed-signal systems at the transistor level, particularly when analog and digital subsystems are tightly coupled. We create variation models for smaller blocks where the number of parameters is manageable. The created analog models are then seamlessly integrated with digital RTL models, so one is able to run parametric yield analysis of a strongly coupled mixed-signal system with fast, event-driven Monte Carlo simulations.

[2]Initially, the maximum order of Taylor series is set to 5, and the response is fitted to that function. A stepwise linear regression is then performed to reduce the order by removing some insignificant terms, resulting in third order Taylor series [38].

[3]The Verilog parameters in the functional model are obtained from the extracted parametric model. The simulation procedure is discussed in Section 5.3.



(a)



(b)

Figure 5.3: Model errors of VCO: (a) residual errors of α and (b) residual errors of β.

5.2.2 Linear Model Failure

All circuits constrain some of their operating parameters, e.g., input amplitude, common-mode range, power supply voltage, and frequency, to ensure their proper operation. These constraints are a part of the circuit, and ensure that the environment is within the valid operating range of that circuit [56]. The assumption of the parametric model is that the circuit remains operating as its intended linear system. However, process parameters also change the valid operation range of the circuit, and it is possible that the model we are using breaks down for some process parameters.

This is especially true when circuits consume or produce large signals, e.g., clocks. The response of such circuits tends to change abruptly when the operating parameters are out of their expected range. For the response to be linear, the large signal inputs/outputs of the circuits may need domain translators, e.g., Φ-to-V and its inverse function for the clock input/output of a PLL, or they are serving as function ports, e.g., clock inputs of a switched-capacitor circuit when constraints on these signals fail. Rather than incrementally changing the transfer function of the circuit, the linear model can be completely broken, leading to functional failure.

An example is a frequency divider model in phase domain, which is given by

$$\Phi_O = \frac{\Phi_I}{M} + \phi_0 \tag{5.2}$$

where M is the dividing ratio, ϕ_0 is the phase (delay) offset, and Φ_I and Φ_O are phase input and output of the divider, respectively. The model also has implicit variables that should be set for the domain translators Φ-to-V and V-to-Φ: the frequency of the input and output clock signals. Their relationship is given by

$$f_O = \frac{f_I}{M} \tag{5.3}$$

where f_I and f_O are the frequency of the input and output clock signals, respectively.[4]

[4]To explicitly represent the underlying linear intent of the circuit, one may write the model by inserting two domain translators: Φ-to-V at the clock input and V-to-Φ at the clock output. Conversely, the two domain translators should be switched when making the model compatible with other blocks in voltage domain. For V-to-Φ domain translator, we can write its model in a way that the frequency is also extracted from the voltage signal rather than declaring the frequency as

This divider model is broken if it receives an input clock running at a frequency higher than the divider circuit can operate. When testing the divider in the phase domain, the linear model is valid only when the input clock frequency is not too high. If the input clock is too fast, the divider's output will not be at the right frequency and the model fails. Since both the input frequency from the previous VCO block and the maximum operating frequency of the divider itself depend on process parameters, process variations can cause functional failures.

Note that this failure does not occur for linear circuits in electrical domains. For these circuits, linear (AC) analysis is possible in any case since the DC operating point always exists and the response is always smooth around the operating point. For example, even if some of MOS transistors in a voltage amplifier enter into the triode region, the circuit is still operating as an amplifier, but now it just has a small voltage gain which changes smoothly.

To prevent designers from running the functional simulation with a broken linear model, the functional Verilog model should have assertions which monitor its input conditions and warn the designers if the operating conditions are out of the valid range. Process parameters must be considered in these assertions since the valid range of the operating parameters depends on process parameters. We demonstrate the use of a support vector machine classifier to create parameterized assertions that depend on process parameters and input conditions in Section 5.4.

5.3 Parametric Yield Estimation

With the parametric model described in Section 5.2.1, process variation effect on a system can be estimated by performing Monte Carlo simulations using process-aware Verilog models. First, the system is hierarchically decomposed into small blocks. The parametric model for each block is generated by characterizing how the gain matrix of a circuit netlist $\mathbf{G_{CKT}}$ varies with process parameters $\mathbf{p}$. This characterization process

the model parameter. However, for Φ-to-V, the frequency is an implicit variable of the translator although the variable actually sets the operating point of the circuit. To resolve this issue, there are two ways: 1) attach the inverse domain translator ahead of Φ-to-V and drive the input with a voltage pulse or 2) take the frequency input as well as the phase in Φ-to-V.

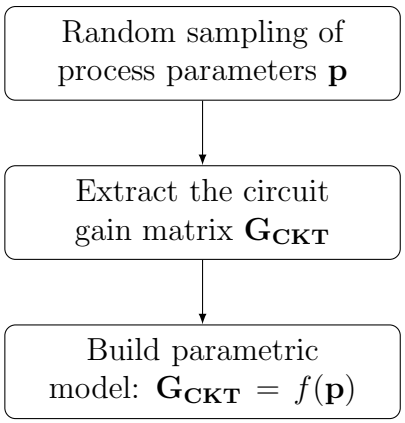


Figure 5.4: Procedure for generating a parametric model.

is depicted in Figure 5.4. The process begins by randomly sampling $\mathbf{p}$. For each $\mathbf{p}$ vector, the corresponding device model is used for circuit simulations from which $\mathbf{G_{CKT}}$ is extracted. Note that all the tests for extracting $\mathbf{G_{CKT}}$ already exist. Since we are observing the variation in $\mathbf{G_{CKT}}$, we reuse the tests used for the equivalence checking described in Chapter 3. This means that we also use the equivalence checker for this characterization.

The parametric model, i.e., $\mathbf{G_{CKT}} = f(\mathbf{p})$, is generated by running linear regression from the extracted samples. To judge the accuracy of the model, the R^2 metric used in the equivalence checker is adopted again. The order of the fitting polynomial function along each axis of $\mathbf{p}$ and the order of the interaction terms may vary depending on the nonlinearity of the response. Our case studies show that R^2 of the fitted model is usually larger than 0.999 using less than fifth order polynomial function and the first order interaction terms.

After building the parametric models of all subcircuits, parametric yield analysis is performed with the procedure shown in Figure 5.5. The column vector of the process parameter $\mathbf{p}$ is supposed to have some distributions and statistics, which are usually provided by process engineers. For given distribution functions, $\mathbf{p}$ is sampled to generate the corresponding Verilog models by solving the inverse of $\mathbf{G_{CKT}} = f(\mathbf{p})$, i.e., $\mathbf{p_{Verilog}} = f^{-1}(\mathbf{G_{CKT}})$, and plugging $\mathbf{p_{Verilog}}$ into the parameterized Verilog models. The functional Verilog models should be parameterized to map the circuit

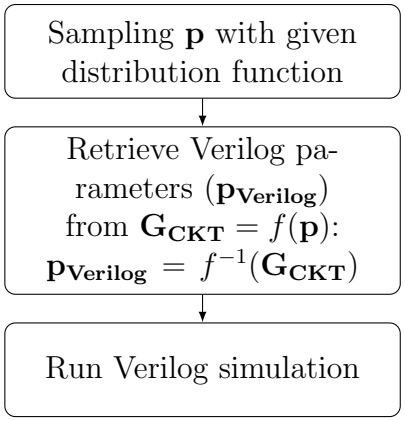


Figure 5.5: System-level Monte Carlo simulation flow.

gain matrix to the Verilog parameters to run Monte Carlo simulations. To ease the mapping, the Verilog model should be written to give a one-to-one correspondence between the partial gains of the gain matrix and the Verilog parameters. That is, each Verilog parameter in the functional model represents the partial gain of the corresponding $\mathbf{G_{CKT}}$. Then, Verilog parameters for each block-level Verilog model are easily retrieved by calculating $\mathbf{G_{CKT}}$ from the parametric model, which instantiates the Verilog models corresponding to the sampled $\mathbf{p}$.

One can build a system model by connecting the instantiated Verilog models. Simulating this is trivial because all test vectors and measurement scripts are already available for measuring the system performance metrics. By running Monte Carlo simulations for those system tests, the distributions and statistics of the metrics are estimated.

In addition to yield analysis, designers sometimes want to quickly estimate the feasible process parameter space for the circuit to meet system specifications. This is especially useful for process parameter tuning. Rather than characterizing the circuit first, one can perform reverse engineering to explore the feasible device parameter space from the Verilog parameter space by running the model simulation, and then use this information to tune the process parameters.

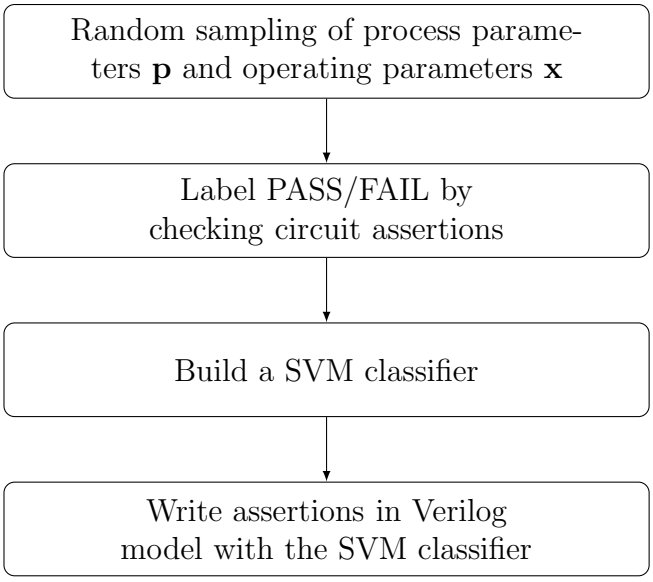


Figure 5.6: Procedure for creating Verilog assertions.

5.4 Assertions on Model Failure

As noted in Section 5.2.2, one needs to ensure that the linear models are valid for the operating parameters when running the system-level simulation. Since the valid region depends on process parameters, we learn this relationship by using circuit simulation results to train a classifier and then use this classifier to predict failures in the Verilog model as shown in Figure 5.6.

For each block, the classifier is built based on training data which are obtained through circuit simulations of a circuit netlist. By sampling different set of operating parameters $\mathbf{x}$ and process parameters $\mathbf{p}$ for each simulation, the simulated responses are labeled as 'PASS' if the circuit is working properly or 'FAIL' if the circuit is not. The assertions to label 'PASS' or 'FAIL' on the circuit simulation result should be provided by the user because the user knows the intended function of a circuit.

From the training data, we build the prediction model which essentially does binary classification. For simplicity, a support vector machine (SVM) classifier with Gaussian radial basis kernel function is adopted for the modeling, and the existing

Listing 5.1: Verilog code example with model failure assertion

```
1  import "DPI-C" function int dpi_failure
2  ( input string line_input,
3    input string modelfilename,
4    input string rangefilename ) ;
5
6  parameter string svm_model = "train.dat.model.div2";
7  parameter string svm_model_scale = "train.dat.scale.div2";
8
9  always @(vdd or freq_cki) begin
10   // SVM input from operating/process parameter
11   str = build_svm_str( vdd, freq_cki, nmos_dvth_norm, pmos_dvth_norm
         ,
12                        nmos_dkc_norm, pmos_dkc_norm);
13   // check if the model is broken
14   //returns 1 if pass, 0 else
15   pass_div2 = dpi_failure(str, svm_model, svm_model_scale);
16 end
```

LIBSVM C library is used for the implementation [57, 58]. The prediction model parameters are found and optimized by grid search and cross validation.

The assertion is then inserted into the Verilog model, which calls the prediction model (SVM classifier) via SystemVerilog direct programming interface (DPI) [15]. When the event occurs in any of operating parameters during the simulation, the assertion checks if the model is valid. For example, Listing 5.1 shows part of the code for the assertion in the frequency divider model. The files assigned by *svm_model* and *svm_model_scale* variables store the trained classifier of the frequency divider. When either the input clock frequency *freq_cki* or the power supply voltage *vdd* is changed, the function *dpi_failure* calls the prediction model with the operating parameters (*freq_cki* and *vdd*) and the process parameters (*nmos_dvth_norm*, *pmos_dvth_norm*, *nmos_dkc_norm*, and *pmos_dkc_norm*), which returns either 1 (PASS) if the model is valid or 0 (FAIL) if the model is invalid. All PASS/FAIL flags in the block models are collected and monitored during the system simulation for the functional failure detection.

5.5 Thoughts on Mismatch Model

For device mismatch analysis, we measure the statistical variation over the collected set of gain matrices from the circuit netlist and map the distribution to the Verilog parameters by performing linear sensitivity analysis. In case of parametric variation model (global variation model) explained in Section 5.2.1, the range of process parameters is usually large such that the nonlinear response should be considered; we model this nonlinear effect with a Taylor series. However, the mismatch analysis is often done with linear analysis, especially for analog circuits since the device size of the circuits is usually large enough for the circuit variation to be approximated by a linear function. Moreover, the linear sensitivity analysis requires much smaller set of samples for modeling device mismatch.[5]

For block-level modeling, we collect the gain matrices of a circuit netlist for random device parameters chosen according to the statistics of the parameter variation. Then we calculate the statistical distribution over the collected gain matrices. We directly calculate the distribution of the corresponding Verilog parameters from the gain matrix distribution.

For generating Verilog model with device mismatch effect, we change the model parameters instead of changing device parameters. Assuming that the device parameter variation is small, the response change is small so that linear perturbation analysis is possible for calculating Verilog parameter variation. Given that linear perturbation analysis holds, we can model the effect of the parameters on the gain matrix as follows:

$$\mathbf{G_{MDL}} = \mathbf{S_{MDL}}\mathbf{p_{MDL}} \tag{5.4}$$

where $\mathbf{G_{MDL}}$ is the gain matrix of the Verilog model, $\mathbf{S_{MDL}}$ is the sensitivity matrix, and $\mathbf{p_{MDL}}$ is a column vector of Verilog model parameters. Since the number of Verilog parameters is modest, we can directly find the above relationship by sampling a few Verilog parameters.

[5]In the mismatch model, all devices in a circuit have different set of process parameter values, and thus the number of device parameters becomes unmanageable as the circuit size grows. If one wants to model the device mismatch effect with Taylor series, it requires a huge number of samples to fit the response.

Given that we know the distribution of the circuit gain matrix $\mathbf{G_{CKT}}$ and $\mathbf{S_{MDL}}$, we can obtain the distribution of $\mathbf{p_{MDL}}$ by solving the inverse of Equation 5.4 with matching the distribution of $\mathbf{G_{CKT}}$ and $\mathbf{G_{MDL}}$. The Verilog parameters can be calculated as follows.

$$\mathbf{p_{MDL}} = \mathbf{S_{MDL}^{-1}} \mathbf{G_{CKT}} \tag{5.5}$$

Usually, the distribution functions of device parameters are supposed to be Gaussian when running Monte Carlo simulations for circuits such that the distribution function of $\mathbf{G_{CKT}}$ shows the same distribution function because we assume the linear approximation is valid. Thus, if the distribution function of $\mathbf{G_{CKT}}$ is Gaussian and $\mathbf{p_{MDL}}$ contains only independent random variables, $\mathbf{p_{MDL}}$ is also Gaussian random variable with its mean and variance are linear combination of $\mathbf{G_{CKT}}$ by $\mathbf{S_{MDL}^{-1}}$. With the known distribution of $\mathbf{p_{MDL}}$, one is able to run Monte Carlo simulations with the analog models for mismatch analysis.

5.6 Example: Phase-Locked Loop

To demonstrate the feasibility of this model-based approach for process variation analysis, the PLL circuit shown in Figure 5.1 is examined. The PLL generates a 1 GHz clock $\mathrm{CK_O}$, where its phase is aligned to the 500 MHz reference clock $\mathrm{CK_I}$. The loop filter in the PLL is simply an integrator with a zero implemented by a passive resistor and capacitor.

For the analysis, the PLL is divided into three sub blocks: 1) voltage controlled oscillator, 2) frequency divider, 3) phase detector/charge pump/loop filter. Phase detector, charge pump, and loop filter are analyzed together for two reasons: 1) testbenches at the circuit-level verification were ready for these combined blocks to get the transfer function, and 2) the loop filter and charge-pump must be analyzed together for writing the Verilog model. For each block, the parametric model is extracted for parametric failure analysis, and the SVM classifier is trained to detect functional failures. Using the parametric model and the classifier, Monte Carlo simulations are performed on the complete PLL model to estimate the parametric variation of the

PLL performance, and detect PLL deadlock.

In this example, we first explore the statistical distribution of the static phase offset for given distributions of process parameters. There are many causes for this offset. One example is that the finite output impedance of transistors causes the UP/DN current mismatch in the charge-pump circuit even without device mismatches. We also use the SVM classifiers to explore functional failures. Failures may occur because either the input clock frequency of the feedback divider is too high, the phase detector is too slow to generate correct UP/DN pulses for given phase difference, or the level-shifter in the VCO is too slow to convert a low-swing clock to a full-swing clock. We show how accurately the SVM classifier-based assertions in Verilog models capture these functional failures by comparing the model simulation with circuit simulation results.

5.6.1 Device Model for Process Variation

For the experiments, we use a simple corner model for both NMOS and PMOS transistors, shown in Figure 5.7.[6] We add a threshold voltage variation ΔV_{TH} and scale factor k_c of the drain current I_{DS} to a 65 nm PTM (Predictive Technology Model) transistor [60]. These parameters are independently set for nMOS and pMOS devices, giving us four process parameters to vary. While this is a very simple fabrication model, it allows us to demonstrate how the modeling framework works. For both parametric and functional failure analysis, only global variation is considered; all the transistors of the same type in a circuit have the same characteristics.

The range of device parameters for the circuit characterization is summarized in Table 5.1. At the block-level modeling, the device parameters within the range specified in the table are randomly sampled for the circuit characterization while some distribution functions of the parameters are given in running Monte Carlo simulations at the top level.

[6]This model is similar to Pelgrom's device mismatch model [59]. We slightly modify the model for this global variation analysis.

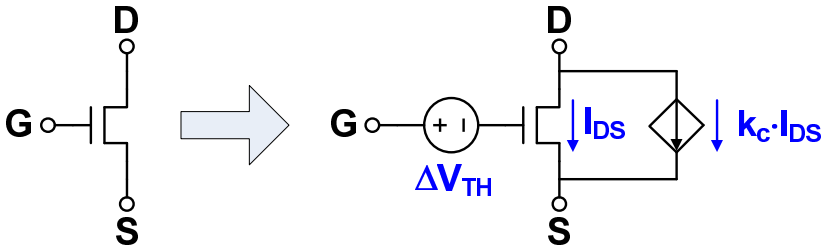


Figure 5.7: Process variation model of a MOS transistor.

Parameter	Range
$\Delta V_{TH,NMOS}$	$[-50, 50]$ [mV]
$k_{c,NMOS}$	$[-0.1, 0.1]$
$\Delta V_{TH,PMOS}$	$[-50, 50]$ [mV]
$k_{c,PMOS}$	$[-0.1, 0.1]$

Table 5.1: Range of device variation parameters for process variation analysis.

5.6.2 Block-Level Modeling

We first write a Verilog model for each sub block of the PLL, and then characterize the impact of process variation.

Phase Detector/Charge Pump/Loop Filter

Figure 5.8 shows the phase detector and the charge-pump circuit with the loop filter. The phase detector (PD) measures the phase difference between the reference and feedback clock inputs, CKR and CKF, at their rising edge, which generates UP and DN pulses. The difference in UP/DN pulse width is the same as the phase difference between the two clocks. In the charge-pump (CP) circuit, these two pulses control the current flow into/out of the loop filter. The CP injects a current onto the filter when UP pulse is asserted while a current is flowing out of the filter when DN pulse is asserted.

This is the most important block for the verification because it mainly affects the static phase offset among sub blocks in the PLL. The charge injection onto the

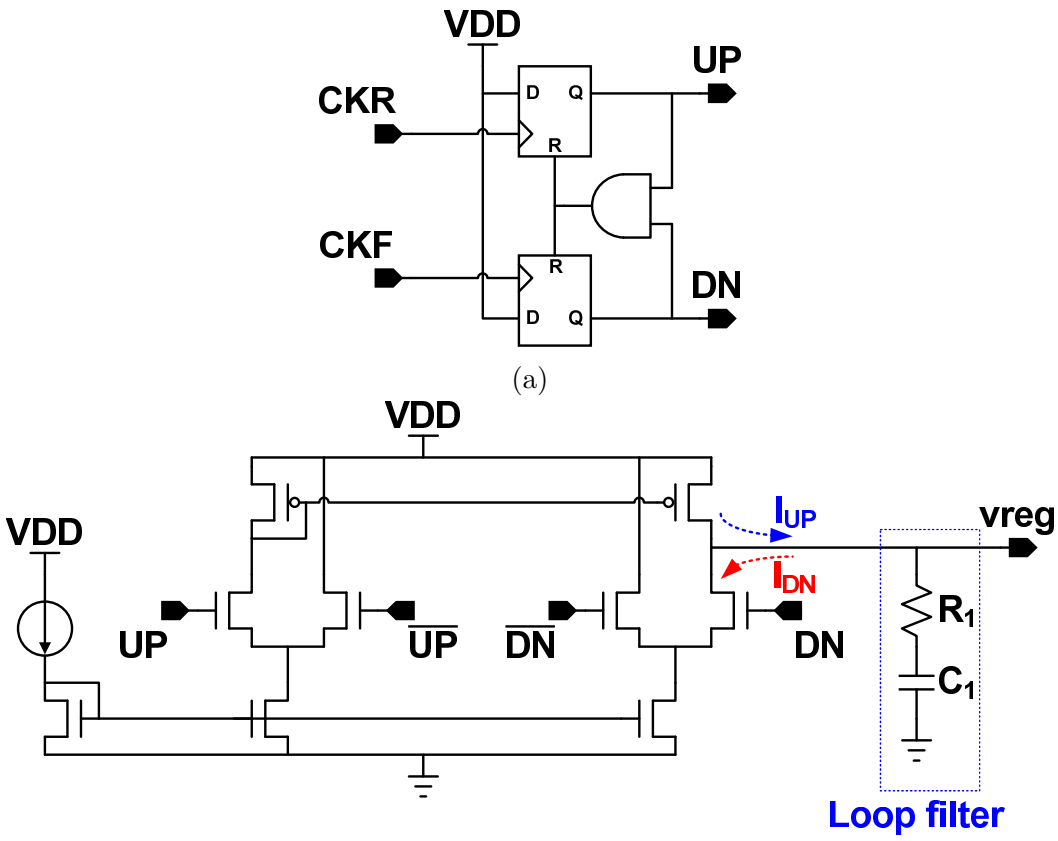


Figure 5.8: A simplified circuit diagram: (a) phase detector and (b) charge-pump with loop filter.

loop filter from the charge-pump circuit should be zero on average when the PLL is locked. A phase offset exists between the reference clock and the feedback clock if up/down currents into/out of the loop filter are not equal. In the circuit examined, since the signal paths for generating UP and DN pulses are symmetric and only global variation of device parameters is modeled, it is unlikely that the PD causes the static phase offset. Instead, the offset is mainly caused by two mismatch sources in the charge-pump circuit: different time constants of UP and DN pulse propagation and up/down current, I_{UP} and I_{DN}, mismatch. First, DN pulse directly controls the I_{DN} current flow while I_{UP} controlled by UP pulse is mirrored by PMOS devices. The pulse width of two currents are different although the pulse width of UP and DN

from the PD circuit are the same. Second, I_{UP} and I_{DN} values are different due to the finite output impedance of PMOS and NMOS current sources. In addition, the difference also depends on the output voltage vreg of the circuit for the same reason. Therefore, these effects should be considered when generating the model.

To build a simple, but accurate model for emulating the CP current mismatch, we write a test to see how the accumulated charge Q_{tot} on the filter changes to the phase difference $\Delta\phi_{err}$ of input clocks and the loop filter voltage $\mathbf{V}(\text{vreg})$, of which relationship is given by

$$Q_{tot} = \alpha_{0,cp} + \alpha_{1,cp} \cdot \Delta\phi_{err} + \alpha_{2,cp} \cdot \mathbf{V}(\text{vreg}) \tag{5.6}$$

where $\begin{pmatrix} \alpha_{0,cp} & \alpha_{1,cp} & \alpha_{2,cp} \end{pmatrix}$ is the gain matrix of the system. From the equation, the incremental change of the filter voltage $\Delta\mathbf{V}(\text{vreg})$ is given by

$$\Delta\mathbf{V}(\text{vreg}) = Q_{tot}/C_1 \tag{5.7}$$

where C_1 is the capacitance of the filter capacitor. Based on these models, the process variation effect on the gain matrix is extracted with the checker tool.

For functional failure analysis, we sweep the clock frequency of input clocks and check whether the circuit is too slow to generate UP/DN pulses for given environmental parameters, i.e., supply voltage, the frequency of input clocks, and device parameters. Given two in-phase clock inputs with the same frequency, the circuit is labeled as 'FAIL' if the frequency of either UP or DN is different from the input clock frequency.

Voltage Controlled Oscillator

The VCO shown in Figure 5.9 consists of a differential ring oscillator with weakly cross-coupled inverters and a level-shifter to recover a full-swing clock from a low-swing clock generated by the oscillator.

The VCO generates a clock whose instantaneous frequency f_{CK} and phase $\mathbf{\Phi}(\text{CK})$

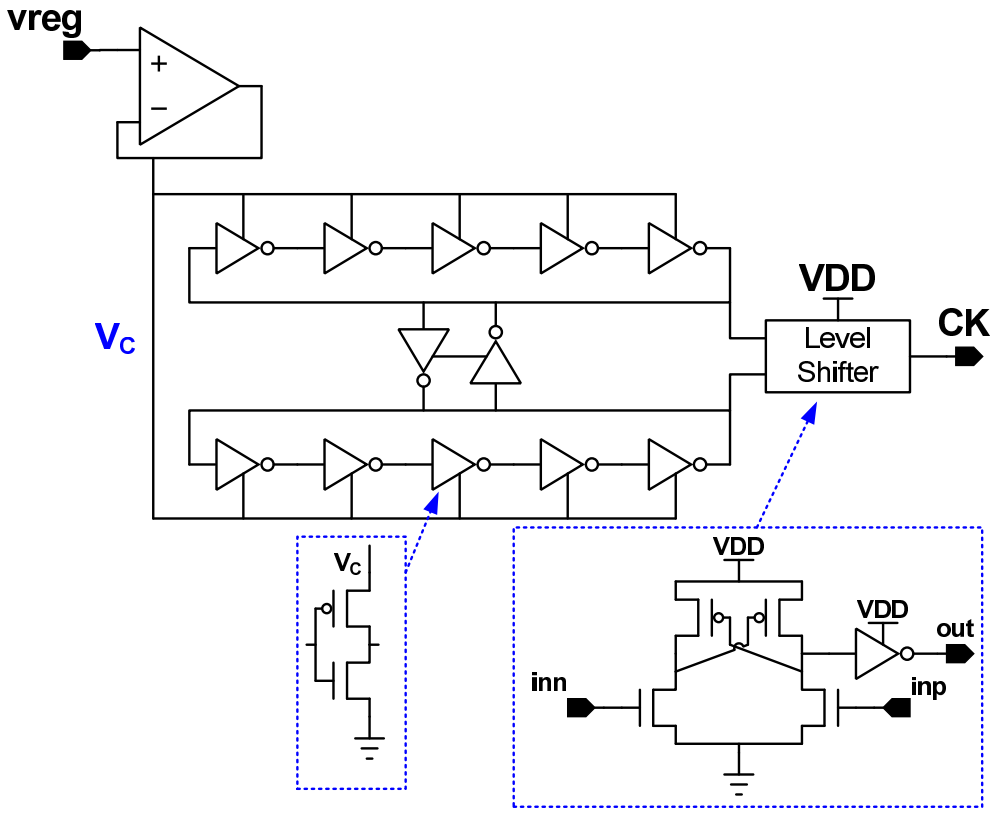


Figure 5.9: A simplified circuit diagram of the VCO.

are given by

$$f_{\mathrm{CK}}(t) = \alpha_{0,vco} + \alpha_{1,vco} \cdot \mathbf{V}(\mathrm{vreg})(t) \tag{5.8}$$

$$\mathbf{\Phi}(\mathrm{CK})(t) = 2\pi \cdot \int f_{\mathrm{CK}}(t)dt \tag{5.9}$$

where $\alpha_{0,vco}$ is a free-running frequency and $\alpha_{1,vco}$ is the gain of VCO in Hz/V. Thus the intended system of interest is Equation 5.8, and its gain matrix, $(\alpha_{0,vco} \quad \alpha_{1,vco})$, of a circuit netlist is characterized with the equivalence checker for randomly sampled device parameters. For example, the variation of the gain matrix to process parameters is fitted with hundred sets of the parameters, resulting in 0.9991 of R^2, and the parametric model is shown in Table 5.2. Note that accurate modeling of the VCO is also important for estimating the static phase offset. The control voltage of the VCO at locking condition varies with process parameters, and this voltage affects the mismatch current of the charge-pump circuit.

$$\begin{pmatrix}\alpha_{0,vco}\\ \alpha_{1,vco}\end{pmatrix} = \begin{pmatrix} -2.332 & 5.673 \\ -0.394 & -5.743 \\ 0.000 & -8.077 \\ 75.177 & -119.603 \\ 372.791 & 0.000 \\ -1.050 & 2.540 \\ 0.582 & -1.439 \\ 0.276 & 4.432 \\ -5.540 & 0.000 \\ -40.423 & 78.604 \\ -0.320 & 0.770 \\ 0.674 & -1.595 \\ 0.854 & -2.476 \\ -55.644 & 61.510 \\ -0.749 & -1.380 \\ 0.695 & 2.774 \\ -0.699 & 1.677 \\ -0.680 & 0.000 \\ -19.770 & 19.453 \\ -7.985 & 0.000 \end{pmatrix}^T \begin{pmatrix} 1 \\ \Delta V_{TH,NMOS} \\ \Delta V_{TH,NMOS}^2 \\ \Delta V_{TH,NMOS}^3 \\ k_{c,NMOS} \\ k_{c,NMOS}^2 \\ \Delta V_{TH,PMOS} \\ \Delta V_{TH,PMOS}^2 \\ \Delta V_{TH,PMOS}^3 \\ k_{c,NMOS} \\ k_{c,NMOS}^2 \\ \Delta V_{TH,NMOS} \cdot k_{c,NMOS} \\ \Delta V_{TH,NMOS} \cdot \Delta V_{TH,PMOS} \\ \Delta V_{TH,NMOS} \cdot k_{c,NMOS} \\ k_{c,NMOS} \cdot \Delta V_{TH,PMOS} \\ k_{c,NMOS} \cdot k_{c,NMOS} \\ \Delta V_{TH,PMOS} \cdot k_{c,NMOS} \\ \Delta V_{TH,NMOS} \cdot k_{c,NMOS} \cdot \Delta V_{TH,PMOS} \\ \Delta V_{TH,NMOS} \cdot \Delta V_{TH,PMOS} \cdot k_{c,NMOS} \end{pmatrix} \begin{bmatrix} \text{GHz} \\ \text{GHz/V} \end{bmatrix}$$

Table 5.2: VCO parametric model: variation of VCO gain matrix to process parameters.

The circuit data is also used to train a SVM classifier. For given set of process parameters, the level-shifter in the VCO may fail to function correctly if the frequency of its input clock from the ring oscillator is too high or the supply voltage of the level shifter is too low or too high. To build the classifier, transient simulations for a short period of time (e.g., 20 ns) are performed to measure the VCO output clock frequency

with respect to sets of process parameters, VCO control voltage, and supply voltage. The measured results are labeled as either 'PASS' if the output clock oscillates, or 'FAIL' if it fails to oscillate.

Frequency Divider

The frequency divider is less interesting for parametric yield analysis. The propagation delay only changes the feedback loop dynamics slightly. While the delay adds additional phase shift in the loop, it does not affect the phase offset.

Functional failure is more interesting since the PLL may be stuck at deadlock condition if the divider fails. When testing the divider in phase domain, the output frequency of the circuit should be what is supposed to be in the V-to-Φ domain translator; the response is labeled as 'FAIL' if the following condition does not hold:

$$f_O = \frac{f_I}{2} \tag{5.10}$$

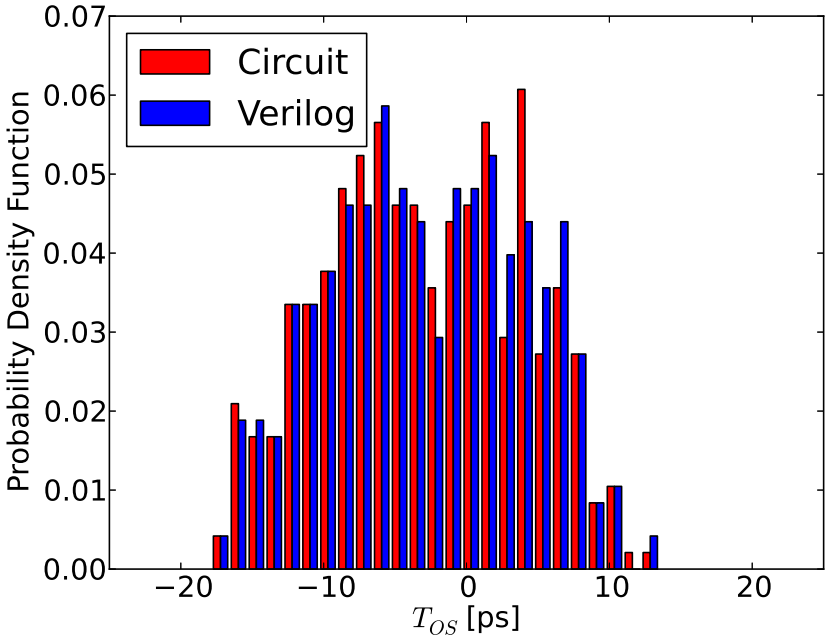
where f_I and f_O are input and output clock frequency, respectively.

5.6.3 Verification Results

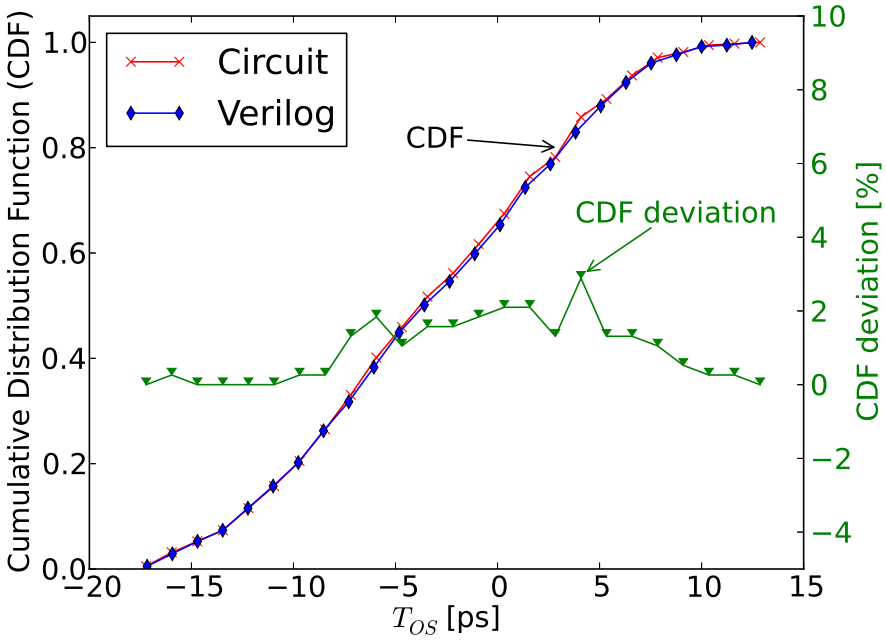
After characterizing each sub block, parametric yield and functional failure analysis are performed with a complete Verilog model of the PLL. For each set of process parameters, the corresponding Verilog parameters are retrieved from the parametric models, and the Verilog model simulation is performed through a functional simulator. Meanwhile, the same analyses are performed with the circuit netlist through a circuit simulator to compare the results to the model simulation results.

To extract the static phase offset T_{OS} distribution, Monte Carlo simulations are performed by sampling sets of process parameters with given uniform distribution functions. As summarized in Table 5.3, the mean and standard deviation of T_{OS} distribution are well matched between the results from the circuit and model simulation.

Figure 5.10 shows the probability density function (PDF) and the cumulative distribution function (CDF) of T_{OS}. The distribution profile is very similar for the



(a)



(b)

Figure 5.10: Distribution of T_{OS}: (a) probability density function and (b) cumulative distribution function.

Statistics \ Model	Circuit in [ps]	Verilog in [ps]	Relative Error in [%]
μ	-3.00	-2.82	-6
σ	6.78	6.85	1

Table 5.3: Summary of parametric variation analysis: mean (μ) and standard deviation (σ) of the static phase offset T_{OS} from the circuit and model simulations. The distributions of device parameters are $\Delta V_{TH} \in \mathcal{U}[-50, 50]$ [mV] and $k_c \in \mathcal{U}[-0.1, 0.1]$. $\mathcal{U}$ is a uniform distribution function.

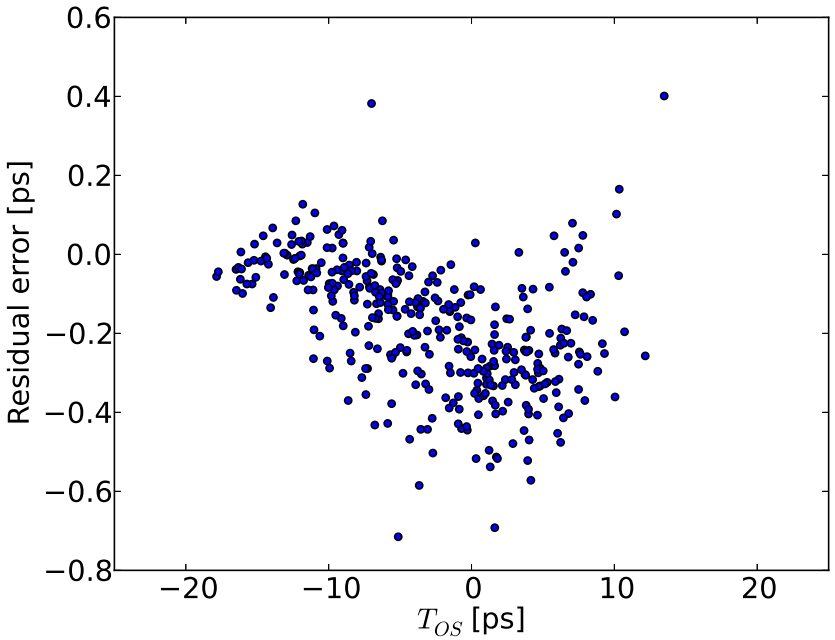


Figure 5.11: Residual errors of T_{OS}: model simulation results to circuit simulation results.

two models; the maximum deviation in CDF between the circuit and model simulation is only 2.9%.

Figure 5.11 shows the residual errors of the model simulation results compared to the circuit simulation results. The mean and standard deviation of the errors are -0.18 ps and 0.16 ps, which are negligible in comparison with the statistical results of T_{OS}.

Statistics \ Model	Circuit in [ps]	Verilog in [ps]	Relative Error in [%]
μ	-2.31	-2.16	-6.5
σ	11.97	12.04	0.6

Table 5.4: Summary of parametric variation analysis with Gaussian distributions of process parameters: mean (μ) and standard deviation (σ) of the static phase offset T_{OS} from the circuit and model simulations. The distributions of device parameters are $\Delta V_{TH} \in \mathcal{N}[0, 50]$ [mV] and $k_c \in \mathcal{N}[0, 0.1]$. $\mathcal{N}$ is a Gaussian distribution function.

With the same parametric models, one is able to estimate the statistics of T_{OS} for different distributions of process parameters. Table 5.4 summarizes the statistics of T_{OS} distribution with Gaussian distributions of process parameters. The simulation results between the circuit and model match well.

Note that the Verilog simulation ran about 400 times faster than the circuit simulation for this analysis; a simulation runs in three seconds with the Verilog model, but needs 1200 seconds with the transistor netlist.

The functional failure simulation is performed with the assumption that the loop filter voltage initially starts at 0.8 V.[7] When running the simulation, we were not able to find a deadlock condition for the given range of process parameters shown in Table 5.1. Thus we insert a faulty capacitor C_F at the output of the feedback divider to make the circuit to fail for some process parameters. Of course the classifier model of the divider is regenerated with C_F.[8] The 'PASS/FAIL' flags from all the sub blocks are checked after running the simulation for a few cycles of the reference clock. If any of the flags is labeled as 'FAIL', the PLL is not working as its intended system.

The results of the functional failure analysis is summarized in Table 5.5.[9] For each of the swept C_F's, the confusion matrix and performance of the classifier is summarized. In the confusion matrix, 'Prediction' is the result from the Verilog simulation and 'Measurement' is that from the circuit simulation. As shown in the

[7]The upper bound of the control voltage vreg is set to 0.8 V as a specification.

[8]When generating the classifier, one can include C_F in the environmental parameters of the classifier, instead of having a fixed value.

[9]Denote true positive, true negative, false positive, and false negative as tp, tn, fp, and fn, respectively. Precision = $\frac{\text{tp}}{\text{tp+fp}}$, Recall = $\frac{\text{tp}}{\text{tp+fn}}$, Accuracy=$\frac{\text{tp+tn}}{\text{tp+tn+fp+fn}}$.

C_F	Measurement / Prediction	PASS	FAIL	Precision [%]	Recall [%]	Accuracy [%]
25 fF	PASS	407	20	93.4	74.0	95.1
	FAIL	4	57			
30 fF	PASS	132	16	95.7	95.4	93.7
	FAIL	15	332			
35 fF	PASS	18	9	98.8	97.9	97.0
	FAIL	5	427			

Table 5.5: Results of functional failure analyses of the PLL shown in Figure 5.1.

table, the accuracy of the prediction model is very high, i.e., over 90%. Although there are some false positive/negative results, these circuits are on the border of functionality, i.e., they are marginal circuits. A small fraction of errors on some of these circuits will not cause a significant error in our yield estimation.

5.7 Summary

In this chapter, we showed how the system-level models written in HDL can be leveraged to rapidly evaluate the effect of process variation on the overall system — essentially running Monte Carlo simulations at the Verilog level. Running Monte Carlo simulations at this level has two advantages: performance, since the Verilog simulations take much less time than the equivalent SPICE simulation, and the capability of measuring the system-level performance metrics with integrated digital logic. The latter is very important since designers often use digital correction to deal with many device effects (e.g. offset correction, gain/bandwidth control, etc.).

A linear abstraction is leveraged to build Verilog models with process variation, and to check if these models are valid for given operating parameters. We showed how the Verilog assertions for this check can be generated using support vector machine classifiers.

Chapter 6

Conclusions

Modern mixed-signal designs need accurate analog models to accelerate the verification as well as to make the verification more complete. Running system-level verification with these models creates many challenges in the verification flow, which we have addressed in this thesis. In particular, based on the insight that analog response is smooth, we have made two main contributions on the model-based verification:

I. We have developed a tool that formally checks if an analog functional model matches its real circuit implementation. We accomplish this checking by using a linear abstraction to define the functional space of analog circuits and to classify the intent of I/O ports in mixed-signal circuits. Using this information, complete and efficient analog vector generation is possible for the system identification. Checking the equivalence of analog models allows designers to run the system-level verification with trusted analog models.

II. We enabled a method to run statistical analysis of process variations using analog functional models. By running fast, system-level Monte Carlo simulations with our functional models, we are able to quickly estimate the parametric variation of system performance so that the iteration cycle between circuit design and process parameter tuning is accelerated.

6.1 Future Work

As can be seen in Chapter 3 and Chapter 4, the main deficiency in our checker tool is inability to automatically generate testbenches. An interesting challenge in generating testbenches is determining what functions need to be measured and how to generate the corresponding testbenches automatically. The circuit parameters being measured often depend on the circuits, which requires design expertise. One way to mitigate this issue is to build a circuit test repository, e.g., Stanford's CircuitBook repository [61]. If the tests already exist in the repository, one can simply pull out and plug the existing tests into the model checker. A more general approach to this testbench generation is to build a generic test. The formal space of our model checking method is a linear system model. Once the proper variable domain for the circuit is selected by the user, the test is simply to explore the linear system's properties (e.g., gain, bandwidth) and the circuit's deviations from the linear system model such as distortion. Since these properties can be calculated from the circuit's impulse response and a Fourier analysis of transient simulation results, the testbench for extracting these responses can be generic and automated.

Another interesting avenue of future exploration related to model validation is assertions between analog cells, i.e., intermodule assertions. A mixed-signal system can have certain bugs if the effect of interconnecting wires between analog cells is not considered, although the analog model of each cell is created and checked against its circuit netlist. Because cells are often created by different designers and models are created before drawing the corresponding layouts, the properties of the connecting wires such as resistance and capacitance are sometimes not considered. This may lead to design mistakes. Consider the bias generator in Section 4.1 that feeds currents into the ramp generator in Section 4.2. If the wire resistance connecting the two modules is too high, the voltage drop across the wire is large, making the PMOS current output device in the bias generator operate in the triode region. This might cause the ramp generator to operate improperly, such that either parametric or functional failures occur in the A/D converter. If proper assertions on pin constraints of a module are embedded in the Verilog models, the interconnection between modules can be checked

by monitoring the assertions. We believe that mixed-signal validation with functional models is more complete with this analysis.

Finally, the next step in this work is to improve the robustness and usability of our checker tool. While our work in this thesis addressed a fundamental problem in analog model validation, the implemented tool is only a prototype intended to demonstrate the concept. One way to improve the tool is by looking at additional case studies, which help to identify problems with the current test setup. Some circuits may require the modification of a test setup to make the linear abstraction valid. Another benefit is that we can build more test library components essential for generating the tests such as stimulus, measurement, and domain translator modules. We believe that case studies help to improve the checker tool as well as make designers more confident that our approach leveraging the linear abstraction works for their analog circuits.

Appendix A

Domain Translator Example

The following listings show some examples of domain translator written in SystemVerilog. These domain translators can be easily ported to other HDLs such as Verilog-AMS.

Listing A.1: SystemVerilog code of **D**-to-**V** domain translator

```
1  // Duty cycle to Logic domain translator
2  // for time-domain simulation only
3  // user specifies 'freq' when instantiating this module
4
5  `timescale 1fs/1fs
6  module duty2ck (
7    input  real dcin,  // duty cycle input
8    output reg cko  // clock output
9    );
10
11 parameter TIME_UNIT = 1e-15;  // verilog time unit
12 parameter real freq = 1e6;  // clock frequency
13
14 real thigh;  // high pulse duration
15 integer period; // clock period in verilog time unit
16
17 initial cko = 1'b0;
18 initial period = $rtoi(1.0/freq/TIME_UNIT);
19
```

```
20 always begin
21   thigh = $rtoi((dcin/period)/TIME_UNIT);
22   cko = #(thigh) 1'b1;
23   cko = #(period-thigh) 1'b0;
24 end
25
26 endmodule
```

Listing A.2: SystemVerilog code of **V**-to-**D** domain translator

```
1  // Logic to Duty cycle domain translator
2  // for time-domain simulation only
3  // frequency information is extracted from cki
4
5  `timescale 1fs/1fs
6  module ck2duty (
7    input        cki,  // logic clock input
8    output real dcout  // duty cycle output
9    );
10
11 real t_pos, t_pos0, t_neg;
12 real period;
13
14 always @(posedge cki) begin
15   t_pos = $realtime;
16   if (t_pos0 > 0.0) // discard the first rising edge
17     period = (t_pos - t_pos0);
18   dcout = (t_neg-t_pos0)/period;
19   t_pos0 = t_pos;
20 end
21
22 always @(negedge cki) // check cki falling edge time
23   t_neg = $realtime;
24
25 endmodule
```

Listing A.3: SystemVerilog code of **Φ**-to-**V** domain translator

```
1  // phase to clock converter for transient simulation
2
3  `timescale 1fs/1fs
4  module phase2ck (
5    `input_real phin,
6    output reg ckout);
7
8  parameter TIME_UNIT = 1e-15; // verilog time unit
9  parameter M_PI = 3.141592; // pi
10 parameter freq = 1.0; // ckout frequency
11
12 reg timer_clk;  // timer for detecting phase exceed first or ckout
       transition first
13 real phin_by_M_PI; // phin normalized by 2*pi
14 real ph0; // accumultaed phase
15 real t0, t1;  // time stamps
16 real ph_transition; // constant to define when ckout flips
17 real half_delay;  // half delay for timer
18 real phin_prev;
19
20 // some initialization
21 initial begin
22   ph0 = 0.0;
23   phin_prev = 0.0;
24   ph_transition = 0.5;
25   ckout = 1'b0;
26   timer_clk = 1'b0;
27   #1 timer_clk = 1'b0;
28 end
29
30 // normalize phin by 2*pi
31 always @(phin) phin_by_M_PI = phin/M_PI;
32
33 // this code is adopted from ringosc.v for PLL monte-carlo
       simulation
34 always @(phin_by_M_PI or timer_clk) begin
```

```
35    t1 = $realtime*TIME_UNIT;
36    ph0 = ph0 + freq*(t1-t0) - (phin_by_M_PI - phin_prev)*
          ph_transition ; // accumulate phase
37    if (ph0 > ph_transition) begin  // flip clock if phase exceed some
           degree (e.g., 180 if ph_transition=0.5)
38      ckout = ~ckout;
39      ph0 = ph0 - ph_transition ;
40    end
41    half_delay = $rtoi((ph_transition-ph0)/(freq/ph_transition)/
          TIME_UNIT);
42    if (half_delay < 1.0) // in case half_delay goes negative at the
          initial start-up
43      half_delay = 1.0;
44    timer_clk <= #(half_delay) ~timer_clk;  // schedule a timer_clk
          flip
45    t0 = t1;
46    phin_prev = phin_by_M_PI;
47  end
48
49  endmodule
```

Listing A.4: SystemVerilog code of **V**-to-**Φ** domain translator

```
1   // clock to phase converter for transient simulation
2
3   `timescale 1fs/1fs
4   module ck2phase (
5     input ckin,
6     input real vdd,
7     output real phout);
8
9   parameter TIME_UNIT = 1e-15; // verilog time unit
10  parameter M_TWO_PI = 2.0*3.141592 // 2*pi
11  parameter freq = 1.0; // ckin frequency in Hz
12
13  real UI_abs, UI_diff, UI_out;
14
15  integer i;  // check this is the first rising edge
```

```
16 initial i = 0;
17
18 always @(posedge ckin) begin
19   if (i == 0) begin // initialize UI_out with the first rising edge
         of ckin
20     i = 1;
21     UI_out = $realtime*TIME_UNIT*freq;
22   end
23   else begin
24     UI_abs = $realtime*TIME_UNIT*freq - UI_out;
25     UI_diff = UI_abs - $rtoi(UI_abs + 0.5) ;
26     UI_out = UI_out + UI_diff;
27     phout = M_TWO_PI*UI_out;
28   end
29 end
30
31 endmodule
```

Listing A.5: SystemVerilog code of the domain translator for checking A-to-D circuit

```
1  // Module for checking A-D ciircut
2
3  module check_a2d (
4    input real inp,  // comparator +input
5    input real inn,  // comparator -input
6    input comp_out, // comparator's digital output (feedback to this
         block)
7    input virclk,  // clock input for running recursive 1-bit DAC
8    output real vcomp_inp, // comparator's actual input(positvely
         going)
9    output real vcomp_inn, // comparator's actual input(negatively
         going)
10   output real vout, // extacted analog output (differential)
11   output real vos // extracted a2d circuit offset
12 );
13
14 import "DPI-C" pure function real fabs (input real x);
15
```

```
16 parameter integer inverting = 0;  // 1 if non-inverting
17
18 real vcomp_diff; // actual differential input for comparator
19 real dv,dv_prev;  // delta V and its previous state for voltage step
      (dv) of vcomp_diff
20 real dv_sign ;  // dv_sign should be -1 if the comparator is
      noninverting, i.e., inverting != 1
21 real vin_diff;  // input difference
22 real vin_cm;  // input common-mode
23
24 parameter vin_diff_high = 0.2; // allowable max vin_diff to
      comparator
25 parameter vin_diff_low = -0.2; // allowable min vin_diff to
      comparator
26
27 assign vcomp_inp = vin_cm + vcomp_diff/2.0 + vin_diff/2.0;
28 assign vcomp_inn = vin_cm - vcomp_diff/2.0 - vin_diff/2.0;
29 assign vout = -1.0*vcomp_diff ;
30
31 initial dv_prev = (vin_diff_high-vin_diff_low)/2.0;
32
33 always @(inp or inn) begin
34   vin_diff = inp-inn;
35   vin_cm = (inp+inn)/2.0;
36 end
37
38 always @(posedge virclk) begin
39   // inverting/non-inverting
40   if (inverting == 1) dv_sign = 1;
41   else dv_sign = -1;
42   // 1-bit dac
43   if(comp_out) dv = dv_sign*dv_prev/2.0; // vcomp_diff should be
        decreased
44   else dv = -1.0*dv_sign*dv_prev/2.0;  // vcomp_diff should be
        increased
45   // integrator
46   vcomp_diff = vcomp_diff + dv;
```

```
47    dv_prev = fabs(dv);
48    vos = vcomp_diff + vin_diff;
49  end
50
51  endmodule
```

Bibliography

[1] Jaeha Kim, M. Jeeradit, Byongchan Lim, and M.A. Horowitz. Leveraging designer's intent: A path toward simpler analog cad tools. In *Custom Integrated Circuits Conference, 2009. CICC '09. IEEE*, pages 613 –620, sept. 2009.

[2] N.A. Kurd, S. Bhamidipati, C. Mozak, J.L. Miller, T.M. Wilson, M. Nemani, and M. Chowdhury. Westmere: A family of 32nm ia processors. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 96 –97, feb. 2010.

[3] S. Damaraju, V. George, S. Jahagirdar, T. Khondker, R. Milstrey, S. Sarkar, S. Siers, I. Stolero, and A. Subbiah. A 22nm ia multi-cpu and gpu system-on-chip. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2012 IEEE International*, pages 56 –57, feb. 2012.

[4] F. O'Mahony, J. Kennedy, J.E. Jaussi, G. Balamurugan, M. Mansuri, C. Roberts, S. Shekhar, R. Mooney, and B. Casper. A 10gb/s 1.4mw/(gb/s) parallel interface in 45nm cmos. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 156 –157, feb. 2010.

[5] D.M. Fischette, A.L.S. Loke, M.M. Oshima, B.A. Doyle, R. Bakalski, R.J. DeSantis, A. Thiruvengadam, C.L. Wang, G.R. Talbot, and E.S. Fang. A 45nm soi-cmos dual-pll processor clock system for multi-protocol i/o. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 246 –247, feb. 2010.

[6] K. Iwata, T. Irita, S. Mochizuki, H. Ueda, M. Ehama, M. Kimura, J. Takemura, K. Matsumoto, E. Yamamoto, T. Teranuma, K. Takakubo, H. Watanabe, S. Yoshioka, and T. Hattori. A 342 mw mobile application processor with full-hd multi-standard video codec and tile-based address-translation circuits. *Solid-State Circuits, IEEE Journal of*, 45(1):59 –68, jan. 2010.

[7] C.P. Lee, A. Behzad, B. Marholev, V. Magoon, I. Bhatti, D. Li, S. Bothra, A. Afsahi, D. Ojo, R. Roufoogaran, T. Li, Yuyu Chang, K.R. Rao, S. Au, P. Seetharam, K. Carter, J. Rael, M. Macintosh, B. Lee, M. Rofougaran, R. Rofougaran, A. Hadji-Abdolhamid, M. Nariman, S. Khorram, S. Anand, E. Chien, S. Wu, C. Barrett, Lijun Zhang, A. Zolfaghari, H. Darabi, A. Sarfaraz, B. Ibrahim, M. Gonikberg, M. Forbes, C. Fraser, L. Gutierrez, Y. Gonikberg, M. Hafizi, S. Mak, J. Castaneda, K. Kim, Zhenhua Liu, S. Bouras, K. Chien, V. Chandrasekhar, P. Chang, E. Li, and Zhimin Zhao. A multistandard, multiband soc with integrated bt, fm, wlan radios and integrated power amplifier. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 454 –455, feb. 2010.

[8] A. Hadjichristos, M. Cassia, H. Kim, C.H. Park, K. Wang, W. Zhuo, B. Ahrari, R. Brockenbrough, J. Chen, C. Donovan, R. Jonnalagedda, J. Kim, J. Ko, H. Lee, S. Lee, E. Lei, T. Nguyen, T. Pan, S. Sridhara, W. Su, H. Yan, J. Yang, C. Conroy, C. Persico, K. Sahota, and B. Kim. Single-chip rf cmos umts/egsm transceiver with integrated receive diversity and gps. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 118 –119,119a, feb. 2009.

[9] Xin He and J. van Sinderen. A 45nm low-power saw-less wcdma transmit modulator using direct quadrature voltage modulation. In *Solid-State Circuits Conference - Digest of Technical Papers, 2009. ISSCC 2009. IEEE International*, pages 120 –121,121a, feb. 2009.

[10] A.M.A. Ali, A. Morgan, C. Dillon, G. Patterson, S. Puckett, M. Hensley, R. Stop, P. Bhoraskar, S. Bardsley, D. Lattimore, J. Bray, C. Speir, and R. Sneed. A 16b

250ms/s if-sampling pipelined a/d converter with background calibration. In *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, pages 292 –293, feb. 2010.

[11] D.C. Daly, P.P. Mercier, M. Bhardwaj, A.L. Stone, Z.N. Aldworth, T.L. Daniel, J. Voldman, J.G. Hildebrand, and A.P. Chandrakasan. A pulsed uwb receiver soc for insect motion control. *Solid-State Circuits, IEEE Journal of*, 45(1):153 –166, jan. 2010.

[12] R. Bagheri, A. Mirzaei, S. Chehrazi, M.E. Heidari, Minjae Lee, M. Mikhemar, Wai Tang, and A.A. Abidi. An 800-mhz ndash;6-ghz software-defined wireless receiver in 90-nm cmos. *Solid-State Circuits, IEEE Journal of*, 41(12):2860 – 2876, dec. 2006.

[13] R.S. Muller and T.I. Kamins. *Device Electronics for Integrated Circuits*. Wiley, 1986.

[14] Boris Murmann. Digitally assisted analog circuits. *IEEE Micro*, 26:38–47, 2006.

[15] Chris Spear and Greg Tumbush. *SystemVerilog for Verification: A Guide to Learning the Testbench Language Features*. Springer, 3rd ed. 2012 edition, February 2012.

[16] M.J.S. Smith. *Application-Specific Integrated Circuits*. VLSI Systems Series. Prentice Hall, 2008.

[17] R.O. Peruzzi. Verification of digitally calibrated analog systems with verilog-ams behavioral models. In *Behavioral Modeling and Simulation Workshop, Proceedings of the 2006 IEEE International*, pages 7 –16, sept. 2006.

[18] W. Hartong, L. Hedrich, and E. Barke. Model checking algorithms for analog verification. In *Design Automation Conference, 2002. Proceedings. 39th*, pages 542 – 547, 2002.

[19] S. Gupta, B.H. Krogh, and R.A. Rutenbar. Towards formal verification of analog designs. In *Computer Aided Design, 2004. ICCAD-2004. IEEE/ACM International Conference on*, pages 210 – 217, nov. 2004.

[20] Thao Dang, Re Donz, and Oded Maler. Verification of analog and mixed-signal circuits using hybrid systems techniques. In *In FMCAD, LNCS*, pages 21–36. Springer, 2004.

[21] Mohamed H. Zaki, Sofiene Tahar, and Guy Bois. Formal verification of analog and mixed signal designs: Survey and comparison. In *Circuits and Systems, 2006 IEEE North-East Workshop on*, pages 281 –284, june 2006.

[22] H. Chang and K. Kundert. Verification of complex analog and rf ic designs. *Proceedings of the IEEE*, 95(3):622 –639, march 2007.

[23] Jaeha Kim, K.D. Jones, and M.A. Horowitz. Variable domain transformation for linear pac analysis of mixed-signal systems. In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on*, pages 887 –894, nov. 2007.

[24] Jaeha Kim, M.A. Horowitz, and Jihong Ren. Stochastic steady-state and ac analyses of mixed-signal systems. In *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE*, pages 376 –381, july 2009.

[25] M. Horowitz, M. Jeeradit, F. Lau, S. Liao, ByongChan Lim, and J. Mao. Fortifying analog models with equivalence checking and coverage analysis. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 425 –430, june 2010.

[26] Byong Chan Lim, Jaeha Kim, and M.A. Horowitz. An efficient test vector generation for checking analog/mixed-signal functional models. In *Design Automation Conference (DAC), 2010 47th ACM/IEEE*, pages 767 –772, June 2010.

[27] Peng Li and L.T. Pileggi. Compact reduced-order modeling of weakly nonlinear analog and rf circuits. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 24(2):184 – 203, feb. 2005.

[28] Joel R. Phillips. Projection frameworks for model reduction of weakly nonlinear systems. In *Proceedings of the 37th Annual Design Automation Conference*, DAC '00, pages 184–189, New York, NY, USA, 2000. ACM.

[29] J.R. Phillips. Projection-based approaches for model reduction of weakly nonlinear, time-varying systems. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 22(2):171 – 187, feb. 2003.

[30] Jinghong Chen and Sung Mo Kang. An algorithm for automatic model-order reduction of nonlinear mems devices. In *Circuits and Systems, 2000. Proceedings. ISCAS 2000 Geneva. The 2000 IEEE International Symposium on*, volume 2, pages 445 –448 vol.2, 2000.

[31] W. Sansen. Distortion in elementary transistor circuits. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(3):315 –325, mar 1999.

[32] P. Wambacq, G.G.E. Gielen, P.R. Kinget, and W. Sansen. High-frequency distortion analysis of analog integrated circuits. *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, 46(3):335 –345, mar 1999.

[33] A. Buonomo and A. Lo Schiavo. Perturbation analysis of nonlinear distortion in analog integrated circuits. *Circuits and Systems I: Regular Papers, IEEE Transactions on*, 52(8):1620 –1631, aug. 2005.

[34] MichałRewieński and Jacob White. A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices. In *Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, ICCAD '01, pages 252–257, Piscataway, NJ, USA, 2001. IEEE Press.

[35] B.N. Bond, Z. Mahmood, Yan Li, R. Sredojevic, A. Megretski, V. Stojanovi, Y. Avniel, and L. Daniel. Compact modeling of nonlinear analog circuits using system identification via semidefinite programming and incremental stability

certification. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 29(8):1149 –1162, aug. 2010.

[36] Ning Dong and J. Roychowdhury. General-purpose nonlinear model-order reduction using piecewise-polynomial representations. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 27(2):249 –264, feb. 2008.

[37] S.K. Tiwary and R.A. Rutenbar. Scalable trajectory methods for on-demand analog macromodel extraction. In *Design Automation Conference, 2005. Proceedings. 42nd*, pages 403 – 408, june 2005.

[38] Douglas C. Montgomery, Elizabeth A. Peck, and Geoffrey G. Vining. *Introduction to Linear Regression Analysis (4th ed.)*. Wiley & Sons, Hoboken, July 2006.

[39] V. Volterra, L. Fantappiè, and M. Long. *Theory of functionals and of integral and integro-differential equations*. Blackie & Son Limited, 1930.

[40] S. Narayanan. Transistor distortion analysis using volterra series representation. *Bell Syst. Tech. J*, 46(3):991–1024, 1967.

[41] K. Kundert. Simulation methods for rf integrated circuits. In *Computer-Aided Design, 1997. Digest of Technical Papers., 1997 IEEE/ACM International Conference on*, pages 752 –765, nov 1997.

[42] P.J. Antsaklis and A.N. Michel. *A Linear Systems Primer*. Birkhäuser Boston, 2007.

[43] B. Razavi. *Design of Integrated Circuits for Optical Communications*. Wiley, 2012.

[44] A.S. Hedayat, N.J.A. Sloane, and J. Stufken. *Orthogonal Arrays: Theory and Applications*. Springer Series in Statistics. Springer, 1999.

[45] Erik Max Francis. Empy 3.1. `http://www.alcyone.com/software/empy`, October 2003.

[46] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pages 11–15, Pasadena, CA USA, August 2008.

[47] Valentin Abramzon. *Analog-to-digital Converters for High-speed Links*. PhD thesis, Stanford University, 2008.

[48] J.G. Maneatis, J. Kim, I. McClatchie, J. Maxey, and M. Shankaradas. Self-biased high-bandwidth low-jitter 1-to-4096 multiplier clock generator pll. *Solid-State Circuits, IEEE Journal of*, 38(11):1795 – 1803, nov. 2003.

[49] Sangho Youn, Jaeha Kim, and M. Horowitz. Global convergence analysis of mixed-signal systems. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 498 –503, june 2011.

[50] R.H. Myers, D.C. Montgomery, and C.M. Anderson-Cook. *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*. Wiley Series in Probability and Statistics. Wiley, 2009.

[51] K.K. Low and S.W. Director. An efficient methodology for building macromodels of ic fabrication processes. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 8(12):1299 –1313, dec 1989.

[52] Zhuo Feng and Peng Li. Performance-oriented statistical parameter reduction of parameterized systems via reduced rank regression. In *Computer-Aided Design, 2006. ICCAD '06. IEEE/ACM International Conference on*, pages 868 –875, nov. 2006.

[53] Xin Li. Finding deterministic solution from underdetermined equation: Large-scale performance variability modeling of analog/rf circuits. *Computer-Aided*

Design of Integrated Circuits and Systems, IEEE Transactions on, 29(11):1661 –1668, nov. 2010.

[54] Xin Li, Jiayong Le, L.T. Pileggi, and A. Strojwas. Projection-based performance modeling for inter/intra-die variations. In *Computer-Aided Design, 2005. ICCAD-2005. IEEE/ACM International Conference on*, pages 721 – 727, nov. 2005.

[55] F. Schenkel, M. Pronath, S. Zizala, R. Schwencker, H. Graeb, and K. Antreich. Mismatch analysis and direct yield optimization by spec-wise linearization and feasibility-guided search. In *Design Automation Conference, 2001. Proceedings*, pages 858 – 863, 2001.

[56] D. Liu, S. Sidiropoulos, and M. Horowitz. A framework for designing reusable analog circuits. In *Computer Aided Design, 2003. ICCAD-2003. International Conference on*, pages 375 – 380, nov. 2003.

[57] Stephen Marsland. *Machine Learning: An Algorithmic Perspective*. Chapman & Hall/CRC, 1st edition, 2009.

[58] Chih-Chung Chang and Chih-Jen Lin. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.*, 2(3):27:1–27:27, May 2011.

[59] M.J.M. Pelgrom, A.C.J. Duinmaijer, and A.P.G. Welbers. Matching properties of mos transistors. *Solid-State Circuits, IEEE Journal of*, 24(5):1433 – 1439, oct 1989.

[60] Wei Zhao and Yu Cao. Predictive technology model for nano-cmos design exploration. *J. Emerg. Technol. Comput. Syst.*, 3(1), April 2007.

[61] James Mao. *CircuitBook: A Framework for Analog Design Reuse*. PhD thesis, Stanford University, 2013.