Title
Learning Approaches to Analog and Mixed Signal Verification and Analysis

Author
Alt, Samantha

Publication Date
2015

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA

Santa Barbara

Learning Approaches to Analog and Mixed Signal Verification and Analysis

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Electrical and Computer Engineering

by

Samantha Alice Alt

Committee in charge:

Professor Malgorzata Marek-Sadowska, Co-chair

Professor Li-C. Wang, Co-chair

Professor Luke Theogarajan

Dr. Chandramouli Kashyap

March 2015

The dissertation of Samantha Alice Alt is approved.

Malgorzata Marek-Sadowska , Committee Co-chair

Li-C. Wang, Committee Co-chair

Luke Theogarajan

Chandramouli Kashyap

December 2014

Learning Approaches to Analog and Mixed Signal Verification and Analysis

by

Samantha Alice Alt

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisors Prof. Malgorzata Marek-Sadowska and Prof. Li-C. Wang for believing in me and constantly challenging me. Each day I am inspired by their dedication and deep knowledge in their respective fields. My aspiration is to reach their level as mentor, researcher, and teacher. Thank you for your consistent support and your guidance on the amazing journey that you have lead me on.

I would like to thank Prof. Luke Theogarajan and Dr. Chandramouli Kashyap for their time and kindness in serving on my thesis committee. The knowledge I have gained from their insights into analog design and industrial practices I would not have been able to comprehend the full impact of this work.

In addition, I would also like to acknowledge Prof. Luke Theogarajan's lab group. In particular, Le Wang, Mohamed Elzeftawi, and Luis Chen, for without their circuit design and support this project would not have been possible.

I would like to acknowledge Semi-Conductor Research Corporation and Intel Corporation for choosing me to be a recipient of their SRCEA Fellowship which supported me through most of my masters and all of my Ph.D studies. In particular, I would like to thank Shirley Maier and Jeff Parkhurst for providing me with their invaluable guidance which I will continue to cherish for the remainder of my career.

I am truly grateful to the ECE department at the University of California, Santa Barbara for offering me various teaching assistant positions. I would also like to thank Val de Verya and many others in the department for their steadfast assistance through my undergraduate and graduate career at UCSB.

Words cannot express my gratitude to my boyfriend for his constant love, support, and understanding. Without him I would have never been able to complete this journey. I would also like to express my enormous thanks to my family. Without their boundless love and belief in me, I would never have had the strength and courage to pursue my dreams. For that, I dedicate this thesis to them.

VITA OF SAMANTHA ALICE ALT

December 2014

EDUCATION

Bachelor of Science in Electrical and Computer Engineering, University of California, Santa Barbara, June 2007
Master of Science in Electrical and Computer Engineering, University of California, Santa Barbara, June 2011
Doctor of Philosophy in Electrical and Computer Engineering, University of California, Santa Barbara, December 2014 (expected)

PROFESSIONAL EMPLOYMENT

September 2014: Intel Corporation; Rotational Engineering Program

Summer 2012: Intel Corporation; Waveform analysis project (Wavecomp) for analog circuit simulation. Tools were written in TCL and performed provided numerical analysis for verification of the system.

Summer 2011: Intel Corporation; Fault emulation project to utilize a logic emulation system for fault grading purposes. Developed a RTL fault lister and fault partitioner which generates a set of initial faults and partitions then into independent sets based on scanout nodes and primary inputs. Tools were written in C++.

June-December 2010: Intel Corporation; First project: Automatic test generation for architecture simulator validation. Using a SMT solver we were able to get high coverage on virtual instructions. Second project: Test point insertion at the RTL level to increase observability of the design.

June-December 2009: Intel Corporation; I worked with Intel on virtual DUT generation for test program validation. We used pre-Silicon data to automatically generate these vDUT in hope to get full coverage of the post-Silicon test program.

Summer 2007 and 2008: Mentor Graphics; Research on improving the diagnosability of the transition fault using a minimal pattern set. The work resulted in a paper that is currently in submission to ITC '09. The paper focuses on using clock manipulation and a small passing pattern set to greatly increase the diagnostic resolution.

2008-2009: Teaching Assistant, Department of Electrical and Computer Engineering, University of California, Santa Barbara

2006-07: Junior Engineer at Multiplex Engineering. Engineer in charge of product support software. Write and maintain open source software used for testing of automotive computer interfaces which is also used in automotive test software. Other tasks included programming and testing of components and assembly technician.

Summer 2003, 2004, 2005: Provided computer operations support to the Research Aircraft Intergration Facility at NASA Dryden, (RAIF) under contract with Lockheed Martin. Responsibilities included system and network management, system administration and support.

PUBLICATIONS

S. Alt, M. Marek-Sadowska, L-C. Wang, "Circuit Partitioning for Behavioral Full Chip Simulation Modeling of Analog and Mixed Signal Circuits," International Journal of Modeling and Optimization, Vol. 4, Feb 2014.

S. Alt, M. Marek-Sadowska, L-C. Wang, "Circuit Partitioning for Behavioral Full Chip Simulation Modeling of Analog and Mixed Signal Circuits," ICSMO, 2014.

S. Alt, M. Marek-Sadowska, L-C. Wang, "On Analog Behavioral Modeling For System-Level Simulation," Techcon 2012.

S. Alt, M. Marek-Sadowska, K-H. Tsai , J. Rajski, "Frequency Manipulation to Maximize the Diagnostic Resolution of Delay Faults," Techcon 2009.

AWARDS

Recipient of the SRCEA/Intel Foundation Fellowship
Recipient of the Semiconductor Research Corporation Masters Scholarship

ABSTRACT

Learning Approaches to Analog and Mixed Signal Verification and Analysis

by

Samantha Alice Alt

The increased integration and interaction of analog and digital components within a system has amplified the need for a fast, automated, combined analog, and digital verification methodology. There are many automated characterization, test, and verification methods used in practice for digital circuits, but analog and mixed signal circuits suffer from long simulation times brought on by transistor-level analysis. Due to the substantial amount of simulations required to properly characterize and verify an analog circuit, many undetected issues manifest themselves in the manufactured chips.

Creating behavioral models, a circuit abstraction of analog components assists in reducing simulation time which allows for faster exploration of the design space. Traditionally, creating behavioral models for non-linear circuits is a manual process which relies heavily on design knowledge for proper parameter extraction and circuit abstraction. Manual modeling requires a high level of circuit knowledge and often fails to capture critical effects stemming from block interactions and second order device effects. For this reason, it is of interest to extract the models directly from the SPICE level descriptions so that these effects and interactions can be

properly captured. As the devices are scaled, process variations have a more profound effect on the circuit behaviors and performances. Creating behavior models from the SPICE level descriptions, which include input parameters and a large process variation space, is a non-trivial task.

In this dissertation, we focus on addressing various problems related to the design automation of analog and mixed signal circuits. Analog circuits are typically highly specialized and fined tuned to fit the desired specifications for any given system reducing the reusability of circuits from design to design. This hinders the advancement of automating various aspects of analog design, test, and layout. At the core of many automation techniques, simulations, or data collection are required. Unfortunately, for some complex analog circuits, a single simulation may take many days. This prohibits performing any type of behavior characterization or verification of the circuit. This leads us to the first fundamental problem with the automation of analog devices. How can we reduce the simulation cost while maintaining the robustness of transistor level simulations? As analog circuits can vary vastly from one design to the next and are hardly ever comprised of standard library based building blocks, the second fundamental question is how to create automated processes that are general enough to be applied to all or most circuit types? Finally, what circuit characteristics can we utilize to enhance the automation procedures?

The objective of this dissertation is to explore these questions and provide suitable evidence that they can be answered. We begin by exploring machine learning techniques to model the design space using minimal simulation effort. Circuit partitioning is employed to reduce the complexity of the machine learning algorithms. Using the same partitioning algorithm we further explore the behavior characterization of analog circuits undergoing

process variation. The circuit partitioning is general enough to be used by any CMOS based analog circuit. The ideas and learning gained from behavioral modeling during behavior characterization are used to improve the simulation through event propagation, input space search, complexity and information measurements. The reduction of the input space and behavioral modeling of low complexity, low information primitive elements reduces the simulation time of large analog and mixed signal circuits by 50-75%. The method is extended and applied to assist in analyzing analog circuit layout. All of the proposed methods are implemented on analog circuits ranging from small benchmark circuits to large, highly complex and specialized circuits.

The proposed dependency based partitioning of large analog circuits in the time domain allows for fast identification of highly sensitive transistors as well as provides a natural division of circuit components. Modeling analog circuits in the time domain with this partitioning technique and SVM learning algorithms allows for very fast transient behavior predictions, three orders of magnitude faster than traditional simulators, while maintaining 95% accuracy. Analog verification can be explored through a reduction of simulation time by utilizing the partitions, information and complexity measures, and input space reduction. Behavioral models are created using supervised learning techniques for detected primitive elements. We will show the effectiveness of the method on four analog circuits where the simulation time is decreased by 55-75%. Utilizing the reduced simulation method, critical nodes can be found quickly and efficiently. The nodes found using this method match those found by an experienced layout engineer, but are detected automatically given the design and input specifications. The technique is further extended to find the tolerance of transistors to both process variation and power supply fluctuation. This information allows for corrections in layout overdesign or guidance in placing noise reducing components such as guard rings or decoupling capacitors. The proposed approaches significantly reduce the simulation time required to perform the tasks traditionally, maintain high accuracy, and can be automated.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Designing analog circuits that are resilient to power supply noise require the adoption of fully balanced, differential topologies. With differential circuit elements, matching of the transistors is important for robust analog design. Traditional design approaches that aim to achieve fundamental device matching may result in over-design and sacrifice performance such as area, speed, and/or power. One of the most important sources of mismatch within analog and mixed signal circuits is the variation in threshold voltage [1].

1.1 Analog and Mixed Signal Circuits

Today's complex system-on-chip (SoC) designs consist of tight interactions between digital and analog cores in order to achieve higher degrees of performance. While the majority of SoC designs are digital and many analog functions have been replaced with digital counterparts, functions which interact with the world around us, which produces continuous time real value information, will always require analog circuitry. An analog, mixed signal, or RF (radio-frequency) circuit is associated with circuits that have a portion of their operating input, output or both consist of continuous time, continuous amplitude signals.

There are three main functionalities of analog circuits in SoC designs; input to the system, output of the system, and interface between analog and digital components [1]. On the input side of a design, signals are transmitted from components such as sensor, microphone, and antenna and must be sensed, received, amplified, or filtered. On the output side of the design, digital signals need to be converted back to continuous real-valued signals and boosted high enough to drive the external load, i.e. loud speaker. Interfacing between analog and digital components are considered mixed-signal components and provide means of converting continuous time signals to binary, binary to continuous time signals, signal sampling, and timing generation (phase-lock loops). For the three types mentioned, additional stabilization and reference circuitry is required for correct operation, i.e. voltage and current reference circuits or crystal oscillators. Due to the large number of devices we interact with on a day to day basis, i.e. smart phones, it is clear that analog circuits will always be prevalent in SoC designs.

Analog components, while vital, only constitute a small part of the SoC, making up between 5-10% [2] of the design. Of this small percentage it has been reported in [3] that 40-50% of the overall design time is spend on these components and of this time, 70-80% is spent on verification of the components.

1.2 Challenges of Analog CAD

Computer aided-design (CAD) is the use of computer systems to assist in the creation, modification, or analysis of a design [4]. For digital CAD many well developed commercial tools are widely available and have been used in practice for many years. On the other hand analog CAD has not had as much commercial support and/or success due to the complexity, sensitivity, and continuous valued nature of the functionality. The most reliable CAD tools are the low level transistor simulators based on SPICE [5] and hardware description languages (HDL) like VHDL-AMS [6] and VERILOG-AMS[7]. Along with the simulators, there have been recent advances in analog topology selection and synthesis [8] as well as automatic generation of layout. While these areas have seen some success, verification methodologies for analog and mixed signal systems are costly due to the complexity and size of the circuits. Therefore it is imperative that efficient methods for verification are developed to reduce design development time and prevent errors that result in manufacturing re-spins.

1.2.1 Simulation

Simulation time of large analog circuits is the major bottle neck for design, verification, and test. As analog circuits are continuous in nature and not discrete like digital circuits, transient simulations are necessary to understand certain behaviors of the circuits. There have been many attempts to reduce the simulations time, but at the price of accuracy.

Transistor level simulators incorporate transistor models and can perform transient, AC, DC, and steady-state analysis of a system. They are very accurate, but come at the cost of speed, which can be very long depending on the number of non-linear differential equations they must solve. HDLs provide an abstraction of the circuit which results in faster simulation times, but sacrifice accuracy. They are event driven simulators that are often used for functional, behavioral level analysis of the design and SoC. Though these tools are useful at various levels of design and verification, they are often used manually and the search of the state space may not be complete, missing critical behaviors. Symbolic simulators and adjoint network techniques are other methods of early design behavioral analysis; these techniques are examined further in Chapter 5. Modeling based approaches use a subset of simulation data or circuit equations to alleviate some of the simulation time. Modeling is described in more detail in the following sections and in Chapter 3.

1.2.2 Verification

The verification of Analog and Mixed Signal (AMS) designs is concerned with the assurance of correct functionality. The AMS design must be robust with respect to different types of inaccuracies like parameter tolerances and nonlinearities. Variations that occur due to manufacturing and environment lead to incorrect operation of the circuit and therefore must be evaluated. The most common method for verifying correctness of a circuit is to use transistor level simulators. Due to the sheer number of combinations of variable parameters and the amount of time some complex analog circuits take to simulate, it may be impossible to simulate verify the entire space. Various methods have been employed to reduce the number of simulations and will be further discussed in the following chapters.

Functional verification is employed in the early stages of a design to catch errors that arise due to wire or connection mismatches. In this type of verification transistor level simulators may not be the most efficient due to the high accuracy and long simulation times. In this type of verification higher level blocks can be simulated in a HDL language because the loss of accuracy is tolerable.

Circuit types are affected by variation types in different ways. In Chapter 2 each circuit used within this thesis is analyzed and verification challenges for each circuit type are described.

1.2.3 Modeling

Analog modeling, at its core, reduces simulation time while attempting to maintain a high level of accuracy. There are various levels of abstraction that models can be used; transistor, block, system. Each level of abstraction reduces the accuracy, sometimes by a significant amount. At the lowest level of abstraction sits SPICE or transistor model based simulators. The transistor models incorporate as much of the device physics and manufacturing behaviors as possible. This is the most accurate modeling available, is widely used, and considered the golden reference for simulations. Many modeling types that are simulation based utilize the transistor level data to build the models.

There are two major types of modeling used for analog circuits: equation-based and simulation-based. Equation-based models mimic the transfer characteristics of the circuit and elements. This leads to substantial reduction in simulation time due to the highly efficient models. Simulation-based approaches utilize transistor level simulation data to extract meaningful behavioral models. The most common of these approaches are black-box-based models which create input-output relationships without knowledge of the inner workings of the system. Both modeling approaches have their advantages and disadvantages and are further explored in Chapter 3.

1.3 Motivation

Analog circuits are notoriously difficult to automate compared to their digital counterparts. They are extremely complex because they are continuously valued, non-linear, and highly sensitive. Many analog circuits are time dependent or may vary over time. The behavior of these circuits needs to be verified and simulated over time in order to ensure proper operation. Unfortunately, for large and complex circuits the simulation over time can take a very long time, on the orders of hours to days. It becomes infeasible when performing this analysis for many input combinations. Therefore we need a method to explore all the possible behaviors of the circuit without exhaustively testing all of the input combinations.

Increasing the input space to include process variation adds extra levels of complexity to the behavior space. A standard and most simplistic method of estimating the behavior of a circuit is to perform corner case analysis. Each corner represents slow/slow, slow/fast, fast/slow, and fast/fast combinations of nfet/pfet devices. Along with the standard design conditions, these four combinations are simulated over time to estimate the worst case performance of a circuit. For verification purposes and capturing the entire output space, these simulations are not enough. While the number of simulations is low, the estimations are often pessimistic and lead to circuit overdesign. This type of analysis does not ensure that interactions between transistors of various variation combinations do not produce a behavior outside of the performance box created by the four corner case simulations.

The following circuit, Figure 1.1, is used as an example throughout this work. It has two main characteristics of interest; feedback and multiple partitions. Feedback is when a signal at the output is fed back to the input of the system. The current and future events of these systems are influenced by previous events in time. This phenomenon implies circuit behavior analysis needs to be performed in the transient (time) domain as well as the standard frequency domain.

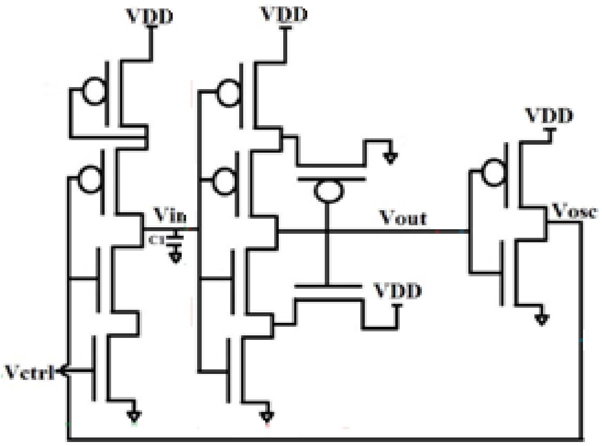


Figure 1.1: Feedback circuit schematic

Most circuits can be broken down into sub-circuits for simpler analysis. Partitioning decomposes a circuit into a set of sub-circuits based on some criteria. In this work we use the Channel Connected Graph-based (CCG) partitioning model. This partitioning scheme is described further in Chapter 3. Partitions of the circuit in Figure 1.1 are shown in Figure 1.2.

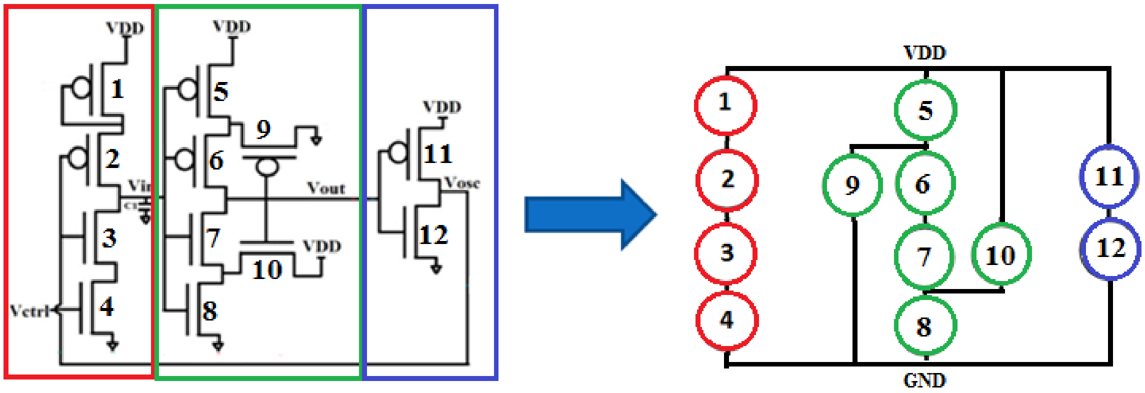


Figure 1.2: Partitioning of feedback circuit using CCG model

Typical behavior of this circuit is shown in Figure 1.3. Intermediate behaviors between partitions for V_{in}, V_{out}, and V_{osc} are also displayed. V_{osc} is the feedback signal which becomes the input to Partition 1 and is also the final output of the circuit. This signal is the most important one because it will connect to external circuitry. For behavior analysis of the circuit, we only care about the final output. The nominal, or typical, behaviors shown here are propagated to the inputs of the next partition and simulated. The typical simulation flow for a transient SPICE simulation is as follows:

1. Compute the initial operating point of the circuit
2. Create linear companion models for non-linear elements such as capacitors
3. Load the currents and conductance's into the nodal matrix
4. Solve the nodal equations
5. Does it converge?
    a. Yes: select time step x(n) and calculate the next time point t(n+1)=t(n)+x(n+1) for all n
    b. No: Select new operating point and go back to step 2

For transient analysis each time step is calculated for the entire circuit. For the circuit in Figure 1.1, since there is feedback, each node is dependent on the previous calculated nodes values, i.e. time dependent. In this circuit the time step is 1nS over a 1000nS total simulation time. The circuit is solved 1000 times, once for each time step.

When the circuit is simulated a single partition at a time, the feedback of the circuit is broken. It can still be solved time dependently by calculating a single time step at a time. For example, given an initial condition for Partition 1, calculate the value for V_{in} at time *t*. Apply V_{in} at *t* as an input to Partition 2 and solve for V_{out}. The same is done for Partition 3 and V_{osc} is calculated then feedback to Partition 1 for the calculation of the following time step *t*+1

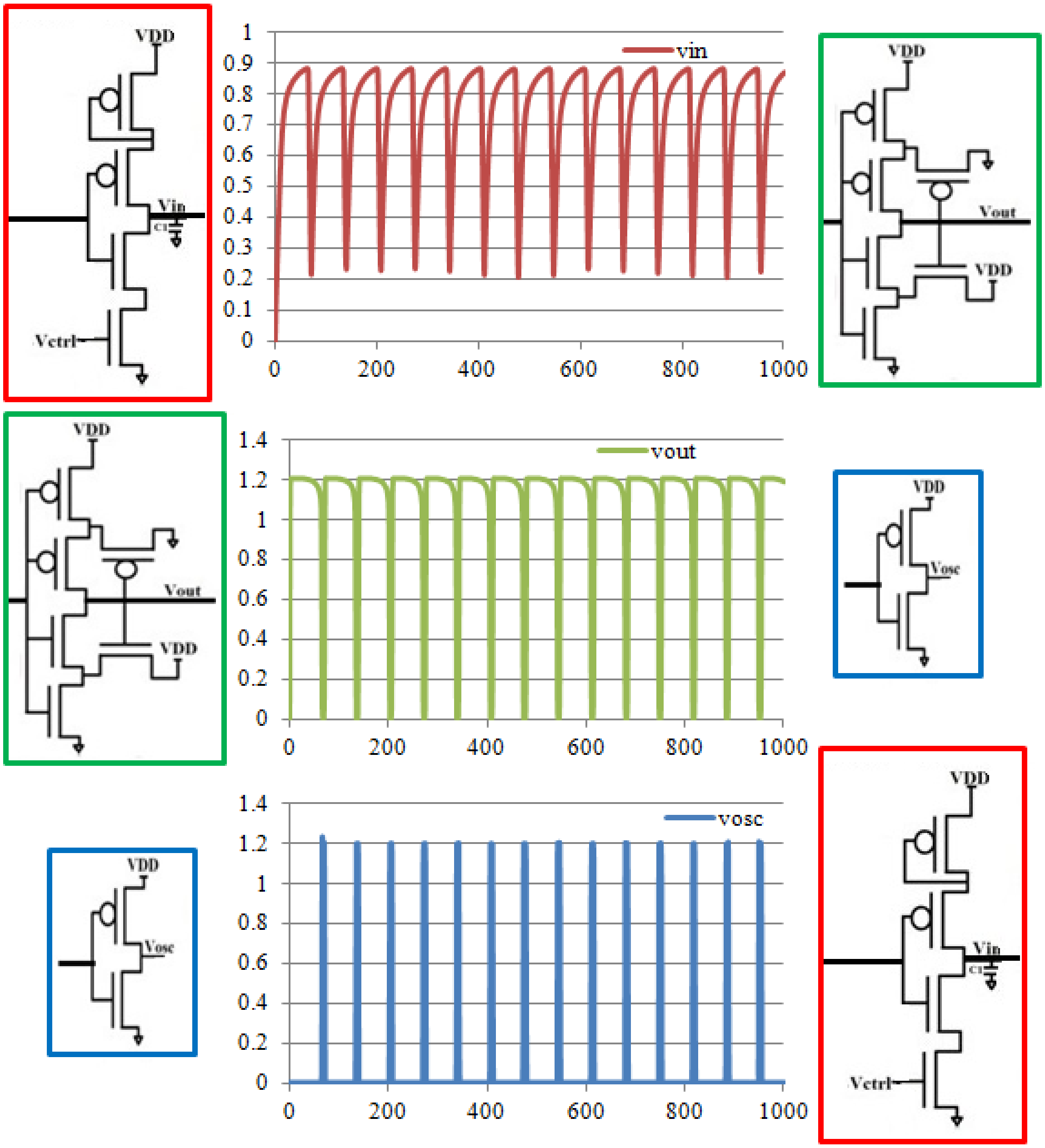


Figure 1.3: Intermediate circuit behavior

The partitioning schemes provide an opportunity to simplify the circuit simulation which in turn reduces the simulation time. Chapter 3 explores a method which utilizes the partitioning to create time based Support Vector Machine behavioral models. These models are built using waveforms generated from full circuit simulation. Each model predicts a single time point at a time and each model is connected for continuous predictions until the entire waveform has been predicted. The models are further extended to include process variations.

There are a few obvious shortcomings to a learning based prediction scheme with respect to analog circuits. SVMs are extremely powerful and created automatically when applied to non-complex partitions which produce regular or not erratic behaviors. For partitions or circuits which are highly complex, creating a learning model requires user intervention and domain knowledge in order to tune the parameters of the learning model. The behavioral models generated are only as good as the data used to generate them. This notion implies that the behavioral models need to be adaptable especially when dealings with new behaviors which are caused by process variation.

With the method explored in Chapter 4 a more efficient way of performing time-based partition simulations was realized. Event-based simulation is a reduced simulation technique that only simulates changes in the waveform. For simplicity let us assume that a waveform can be classified as either repeating (oscillating) or non-repeating. Non-repeating behavior typically is reserved for analog behaviors with the exception being the always high or always low (stuck-at-1 or stuck-at-0 digital behaviors). Repeating waveforms can either be analog (sine wave) or digital (clock). This circuit, Figure 1.1, is a basic clock generator and all three partitions are repeating where only Partition 3 is a digital partition (inverter). The five events from V_{osc} in Figure 1.4 are extracted and modeled as digital events. Event decomposition based on wave type is explained in Chapter 2.

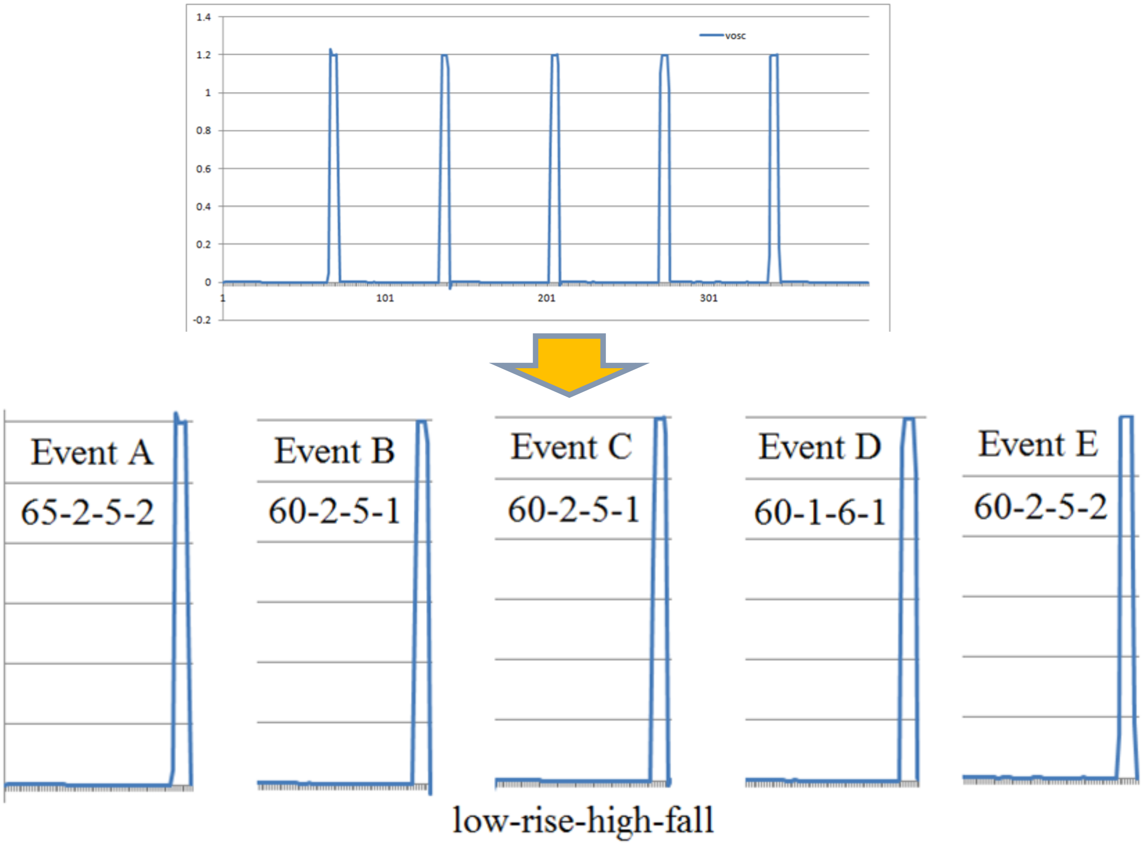


Figure 1.4: Digital event extraction from Vosc waveform

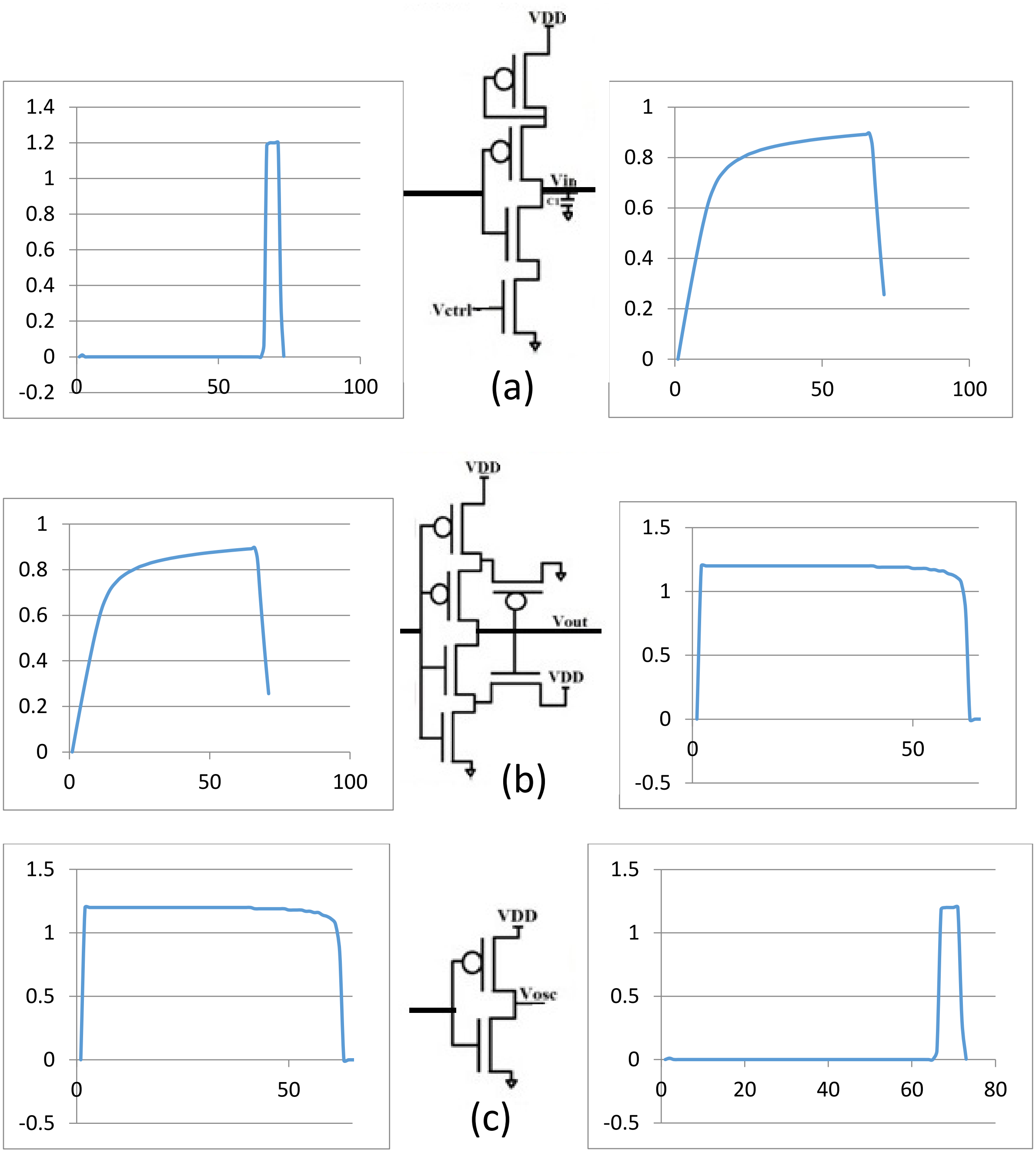


Figure 1.5: Response of partitions to input events (a) Partition 1 response from Partition 3 output (b) Partition 2 response from Partition 1 output (c) Partition 3 response from Partition 2 output.

Regardless of the type of event and how it is modeled, only the original event is propagated to the next partition. Instead of full circuit transient analysis we are now performing partition based transient event analysis. The response of the partition to the single transient event is captured. An example of the propagation of a single event is shown in Figure 1.5(a-c).

To avoid simulating the same event multiple times we need a method of removing similar events. How can we know if the output of Figure 1.5(c) is the same as the input event from Figure 1.5(a)? Similarity measurements between events are performed using a difference metric. The difference metric can be as simple as a pair-wise difference calculation or as complex as the squared correlation coefficient calculation. Two identical waveforms have a similarity measure of 1 or 100% while two complete opposite waveforms have a similarity measure of 0 or 0%. For analog waveforms it is rare that two events will be identical due to the continuous and real-valued nature of the signals. The difference between two values can be as little as 1e-9, but the resulting similarity calculation will never be 100%. These small noise differences can be removed by reducing the definition of similarity to 99%. If two events have a similarity of >99% they can be merged.

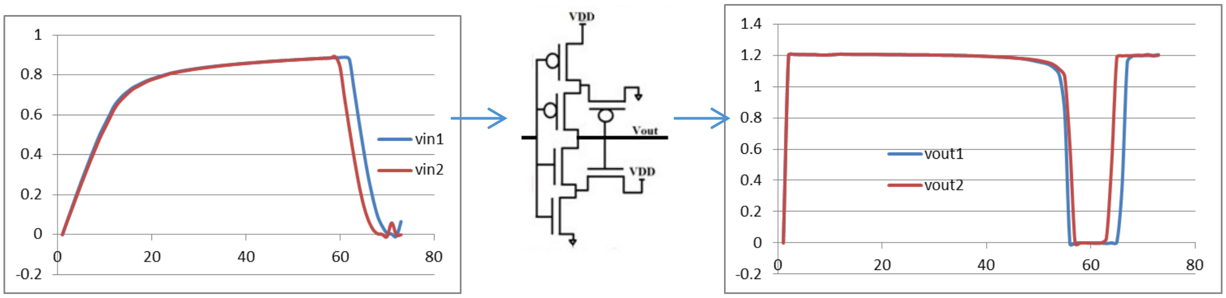


Figure 1.6: Compounding events

Event based decomposition of waveforms and the concept of similarity measures facilitates part of the method described in Chapter 4. One part of the motivational force behind Chapter 4 is the reduction of the input space to contain only inputs which produce unique output behaviors. The concept of merging events based on similarity measures is the main mechanism behind clustering algorithms.

A cluster is a group or set of objects that are similar to one another. The definition of similarity for clusters defines the number of clusters. If the similarity measure is high, meaning two objects need to be very similar in order to be in the same cluster, the more clusters there will be and vice versa. When clustering is performed on the input space a representative event from each cluster is chosen and simulated. These simulated events are the first to populate the output space. An iterative method of completely exploring the output space is in detailed Chapter 4.

Reducing the input space is one way of reducing the simulation time for verification of analog circuits. Another is abstracting the circuit, i.e. behavioral models. From the findings in Chapter 3 it was observed that some partitions were much easier to learn then others. This was due to the complexity and information transfer of the partitions. Therefore a method needs to be developed in order to analyze whether or not the partition should be modeled. If the partition is deemed to be too complex then it is simulated, otherwise it is modeled. These non-complex partitions can also be extended to include process variation.

Reducing the simulations speed for large analog circuits allows for various avenues for analysis to be performed which were otherwise prohibitive. For example, one may perform critical node analysis across wide ranges of inputs and process variation. Or analyze the behavior when the circuit is supplied varying power sources. Chapter 5 expands on the methods developed in Chapter 4 to explore analysis and yield applications.

1.4 Contributions of this Thesis

The work in this thesis addresses multiple issues surrounding analog analysis and verification in the transient domain. We proposed novel methodologies to address modeling, input space, compression, critical node analysis, and power analysis for analog and mixed signal systems. We first propose transient behavioral modeling scheme based on Support Vector Machines and Channel Connected Component Graphs. Circuits are partitioned based on their structure and intermediate behavioral models are built for high sensitivity nets. The models are obtained using Support Vector Machines (SVM), a data dependent black box modeling technique. We demonstrate the soundness of this approach by modeling large circuits such as Sigma-Delta ADC and Phase Lock Loop. Experimental results show that this methodology maintains 95% accuracy behavior predictions while achieving three orders of magnitude speedup over SPICE simulation time.

From the experiences gained in Chapter 3 a new methodology is developed in Chapter 4 to more soundly reduce the simulation time and explore the behavioral output space for verification. A method based on iterative simulation, events, and clustering of the circuit is developed to reduce the input space to only the necessary events to fully capture and characterize the output space. Each partition is analyzed for complexity and information content to determine if the partition is suitable for behavioral modeling. Experimental results show that the methodology significantly reduces the input space and simplifies non-complex portions of the circuit resulting in significant improvement in simulation time and allows for directed and effective circuit verification for very large analog and mixed signal circuits.

In the final chapter, the work is extended to address the critical transistor analysis and environmental variation analysis. Critical transistor analysis involves determining the location of transistors which affect the output of the circuit when perturbed by process variation. Environmental variation analysis is an extension of critical node analysis to include power variation. The analysis techniques utilize the partitioning and event decomposition scheme to initially prune a large number of non-sensitive transistors from the total set of transistors. Each transistor is then simulated at various ranges of process and environment variation to find the sensitivities. The critical transistors found automatically by the tool are compared with those located by an expert designer. We show we can detect a few extra extremely sensitive transistors that need to be redesigned or resized. At the same time we remove some non-critical transistors which get masked by the operation of the circuit.

1.5 Chapter 1 References

[1] G.E. Gielen, R.A. Rutenbar, "Computer-aided design of analog and mixed-signal integrated circuits," *Proceedings of the IEEE* , vol.88, no.12, pp.1825,1854, Dec. 2000

[2] S. Banerjee, D. Mukhopadhyay, D.R. Chowdhury. Computer Aided Test (CAT) Tool for Mixed Signal SOCs. In *IEEE VLSI Design*, pp. 787-790, 2005.

[3] Cadence Design Systems. Using a SoC Functional Verification Kit to Improve Productivity, Reduce Risk, and Increase Quality. White Paper

[4] K.L. Narayan, Computer Aided Design and Manufacturing, *New Delhi: Prentice Hall of India*, 2008

[5] L.W. Nagel, and D.O. Pederson, *SPICE (*Simulation Program with Integrated Circuit Emphasis*)*, Memorandum No. ERL-M382, University of California, Berkeley, Apr. 1973

[6] E. Christen and K. Bakalar, “VHDL-AMS-a hardware description language for analog and mixed-signal applications.” *IEEE Transactions on Circuits and Systems II*, vol. 46, no. 10, pp. 1263–1272, 1999.

[7] K. S. Kundert, The Designer’s Guide to Verliog-AMS, 1st ed. *Boston, MA: Kluwer Academic Publishers*, 2004.

[8] R. A. Rutenbar, G. G. Gielen, and B. A. Antao, Computer-Aided Design of Analog Integrated Circuits and Systems, 1st ed. *New York: IEEE Press*, 2002.

Chapter 2

Overview of Analog Circuits

This chapter focuses on the different circuits used throughout this dissertation. Each circuit discussed has been fabricated and verified by their respective designers. We will discuss the application, structure, and behavior. Each section will conclude with verification challenges for the circuit types. The final section discusses the types of variation applied to the circuits throughout this work. All circuits in this chapter, unless specified, use .13μm CMOS process technology.

2.1 Phase Lock Loops

A phase lock look (PLL) is a feedback system that compares the output phase with an input reference phase [1]. A basic PLL consists of a phase detector (PD), charge pump, and a voltage controlled oscillator (VCO), but also commonly include a divider. The phase detector (PD) is a circuit whose average output is linearly proportional to the phase difference between two inputs. A phase detector can be a XOR gate or a sample hold circuit. The filter receives the output of the phase detector and filters out the high frequency components and presents a DC voltage to the VCO. The filter can be as simple as a RC low pass filter or as complicated as a charge pump. The VCO, ideally, is a circuit whose output frequency is a linear function of its control voltage. The lock term of a PLL implies that the input reference frequency is in exactly the same phase as the generated output frequency. This operation is widely used in both analog and digital systems for synchronization purposes. The main applications are frequency synthesis, clock recovery, and jitter reduction.

2.1.1 Ultra Wide Band Phase Lock Loop

The Ultra-Wideband Phase Lock Loop (UWB-PLL) in Figure 2.1 is used to tune the frequency of the impulse radio ultra wideband transmitter (IR-UWB) [2]. Ultra Wideband communication is based on transmission of very short pulses with relatively low energy over a large frequency bandwidth of several GHz. These devices are extremely useful for biomedical applications due to their low power consumption and can support high data rates.

The UWB-PLL, Figure 2.1, is designed with an IR-UWB transmitter, phase frequency detector (PFD), charge pump (CP), divider, counter and digital to analog converter (DAC). The phase detector takes a reference frequency, F_{ref}, of 31.25MHz and the feedback frequency after it has been divided. The phases are compared and UP/DN signals are generated whose duration is proportional to the phase difference. The CP which is fully differential receives the digital UP/DN signal and generates an analog voltage, V_{cp}. The CP is biased by V_{ctrl}, whereas V_{ctrl} decreases the CP current decreases. The output of the CP, V_{cp}, is fed into the 8-bit up/down/hold counter and then into a DAC to generate V_{ctrl} for the regulator.

The IR-UWB contains a VCO, regulator, pulse positioning modulator and a tunable pulse generator. The Data and Clock are fed into the pulse positioning modulator which triggers a tunable pulse on the rising edge of the clock. The pulse is connected to the enable of the VCO to enable oscillation. V_{ctrl}, is fed into a low dropout regulator, linear voltage regulator, which is biased by V_{ctrl}. The regulator then feds the VCO which generates the frequency, V_{out}, based on the supplied voltage, eventually reaching 2.5GHz. V_{out} is distributed to the frequency divider which consists of seven total stages with two fast stages.

The behavior of the UWB-PLL is shown in Figure 2.2 for 31.25MHz F_{ref} and nominal operating conditions. The partitions CP, PFD, bias, regulator, and divider with signals V_{cp}, V_{pfdup}, V_{pfddn}, V_{nbias}, V_{pbias}, F_{out}, and V_{f_fb} show lock time at approximately1.3µS. The final simulation, F_{out}, shows the frequency generated as it approaches lock at 4.0GHz. The self-biased voltage values, V_{nbias} and V_{pbias}, are feedback from the VCO and are proportional to the VCO's oscillating frequency. The divider, V_{f_fb}, is 128 times slower than the F_{out} in order to compare against the F_{ref} in the PFD.

This circuit takes 16 minutes to simulate and contains 1600 components. With extracted parasitics the circuit contains 60k components and takes 7 hours to simulate.

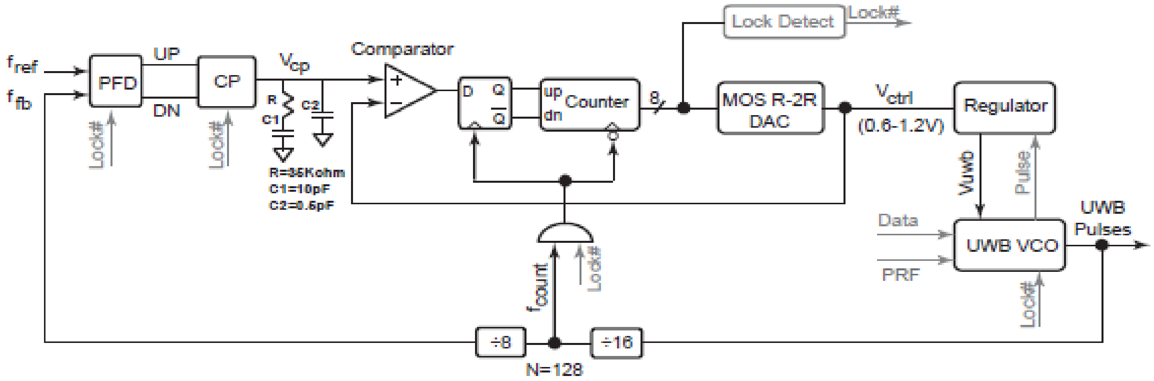


Figure 2.1: Block diagram [2] of the UWB-PLL with embedded digital tracking used to tune oscillation frequency of IR-UWB transmitter

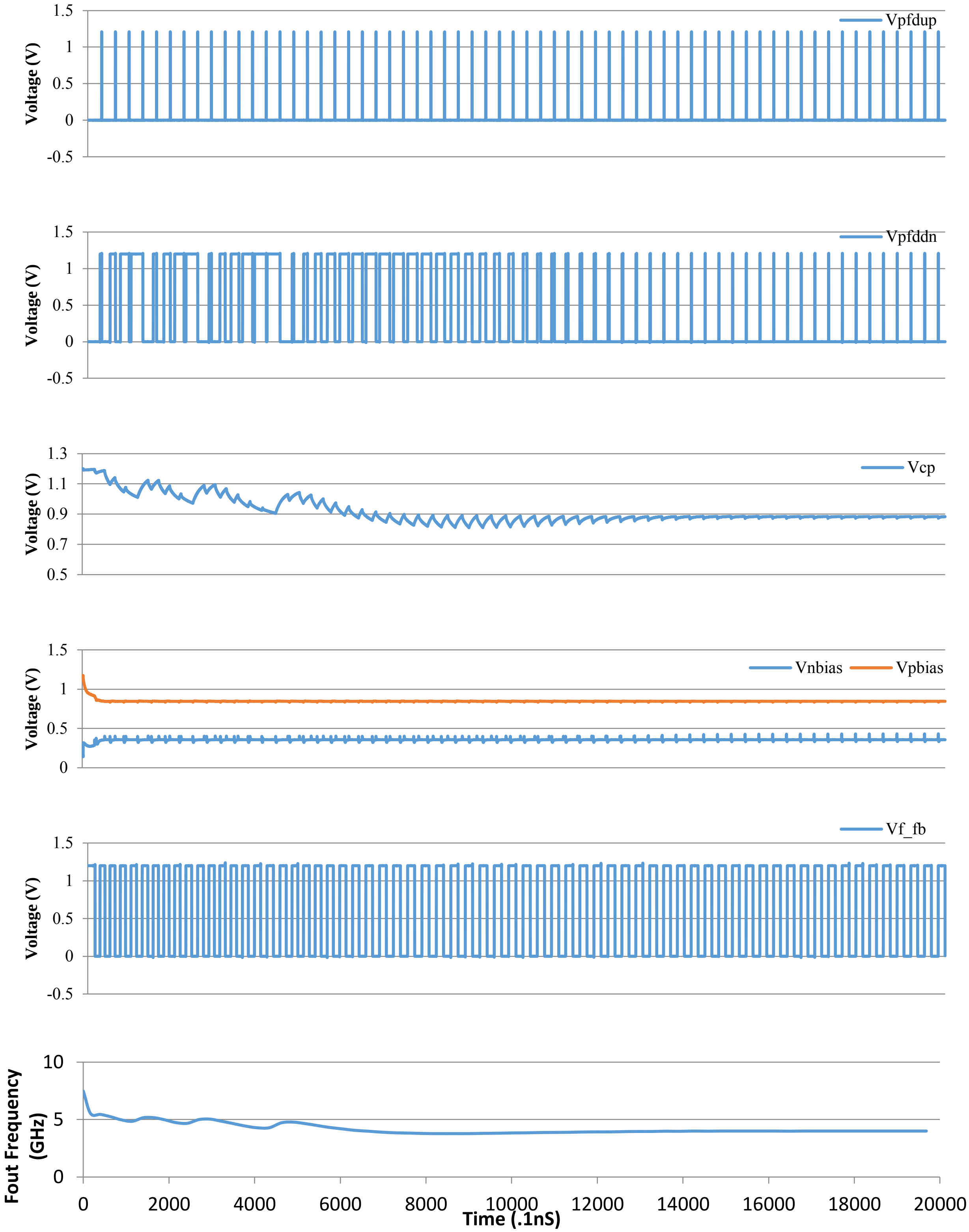


Figure 2.2: Simulation results for PFD, CP, divider and VCO outputs of UWB-PLL over 2μS.

2.1.2 Phase Lock Loop for Clock and Data Recovery

The PLL, Figure 2.3, generates a 2.5GHz clock based on a 31.25MHz reference frequency for clock and data recovery applications [3]. The phase detector and charge pump are the same designs as in the UWB-PLL. On an UP pulse the loop filter is charged and vice versa for the DN pulse. The loop filter smoothes out the abrupt changes cause by the charge/discharge and delivers the signal to a regulator which buffers and regulates the signal. The regulated signal is sent to a pseudo differential ring VCO which creates the oscillation. The frequency is then divided by 16 and feedback to the phase detector for phase comparison.

The behavior of the PLL is shown in Figure 2.4 for 31.25MHz F_{ref} and nominal operating conditions. The partitions CP, PFD, and VCO with signals V_{cp}, V_{up}, V_{dn}, V_{vco}, $V_{vco'}$ and F_{out} show lock time at approximately1µS. The final simulation, F_{out}, shows the frequency generated as it approaches lock at 2.5GHz. The output of the CP, Vcp has some noise in the signal which is removed by the regulator and VCO buffers shown as Vvco' and Vvco.

This circuit takes 12 minutes to simulate and contains 2000 components.

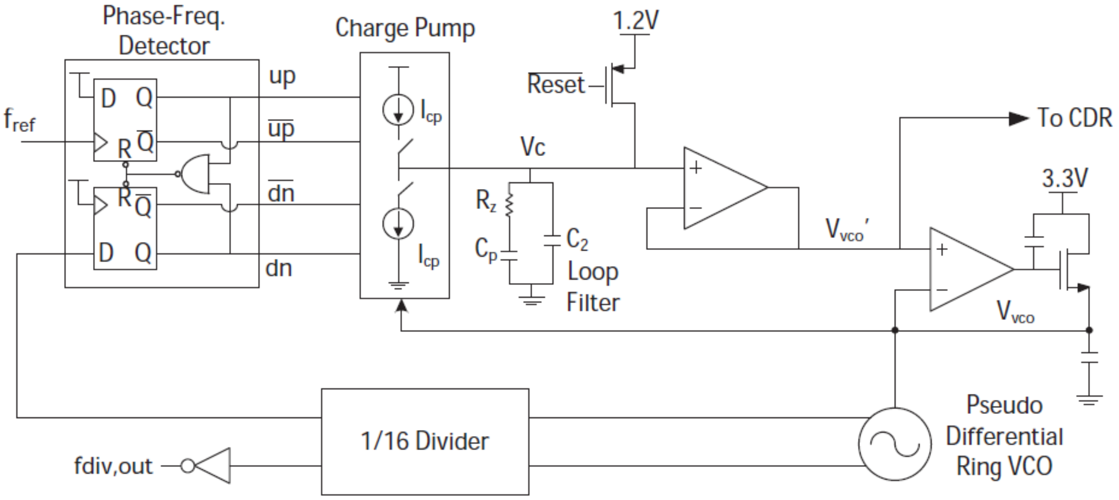


Figure 2.3: Block diagram [3] of a PLL used for clock recovery

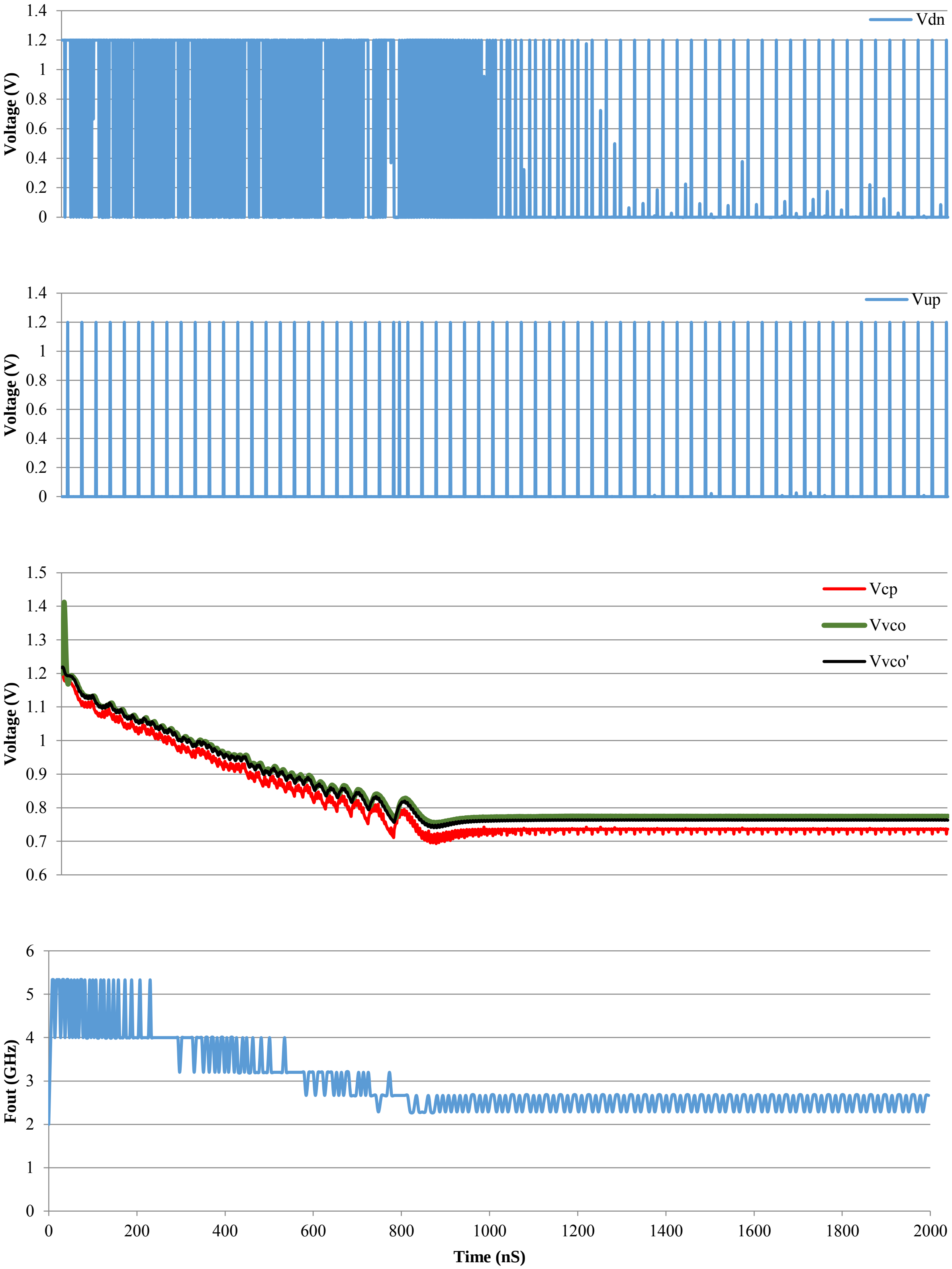


Figure 2.4: Simulation results for PFD, CP, divider and VCO outputs in the PLL over 2μS.

2.1.3 Verification Challenges – PLL

When considering only the external input to the system, F_{ref}, the input space may seem extremely small. This quickly increases when considering various sources of variation. Each transistor can have a range of possible variations increasing the input space by a tremendous amount. Verifying all combinations of variation is obviously impossible due to the sheer number of simulations required. This is further compounded by extremely large and complex circuits which require transient behavioral analysis and take a very long time to simulate. Transient simulations are required to verify the lock time property of the circuit. Lock time is the time it takes to move from one specified frequency range to another specified frequency range within a given frequency tolerance [1]. Essentially the amount of time it takes for the PLL to match the input phase from F_{ref}. The faster the lock time the sooner data can be transmitted.

Jitter introduced by power supply noise or process variation directly affects the sensitivity of the components. Jitter with respect to PLLs is the temporal variation of the phase which is a critical performance metric where too much jitter results in synchronization failures [5].

The feedback nature of a PLL makes verification difficult. Traditionally each block or sub-circuit within the PLL is verified individually and only worst case, or corner case, simulations are done for the whole circuit. Block to block interactions are not typically verified extensively due to the time requirement.

2.2 Converters

There are two types of data converters; a digital to analog converter (DAC) and an analog to digital converter (ADC). A DAC converts a digital signal to an analog signal which is useful when interfacing digital components to analog components such as speakers or amplifiers. An ADC, on the other hand, performs the opposite operation converting analog signals to digital signals which is useful for inputting real world inherent analog signals like sound to be processed by digital components. Converting a continuous-time continuous-amplitude signal to discrete-time discrete-amplitude signal requires a process called quantization which maps a large set of values to a much smaller set of values that introduces rounding errors called quantization errors.

2.2.1 Sigma-Delta Analog to Digital Converter

A low-power high-resolution analog-to-digital (ADC) converter is required to digitize neural data. These ADCs are well suited for low frequency high accuracy measurements. A 2nd order over a 1st-order ∑Δ-ADC, shown in Figure 2.5, is used because the 2nd order does not suffer from "idle-times" or "limit-cycles" for constant inputs and can provide the required 6-bit resolution without consuming as much power [4]. The design consists of two fully-differential self-biased amplifiers, two switched capacitor networks, a 1-bit quantizer, and a non-overlapping clock generator. A ∑Δ-ADC receives an analog input signal and samples each point multiple times, a technique known as oversampling. The sampling is performed much faster than the rate at which the digital signal is outputted. The oversampled data is accumulated over time and averaged. The ∑Δ modulator within the ADC is responsible to the digitalization and noise shaping which pushes low frequency noise up outside the frequency band of interest.

One of the switch capacitor networks and an amplifier create a differential switch capacitor integrator. There are two operations which operate in a time-interleaved manner; the sampling mode and integration mode. The integration mode is active when the switches Φ_2 are shut and sampling is when switches Φ_1 are shut. The single bit quantizer which is inherently linear outputs a stream of digital bits.

The behavior for the ∑Δ-ADC is shown in Figure 2.6 with a sine wave input of 2KHz under nominal operating conditions. The signals shown are the differences between the outputs of each partition for the first 100µS. The simulation starts with the difference between the input sine wave, vin-vip, and propagates through the first switch capacitor network and amplifier, von1-vop1. The output of the first network is the input to the second where the output, von2-vop2, is the input to the quantizer. The output of the quantizer, von-vop, is feedback through the system, but is also the final digital output of the circuit.

This circuit contains 647 components and takes 4.8 minutes to simulate two clock periods using HSPICE.

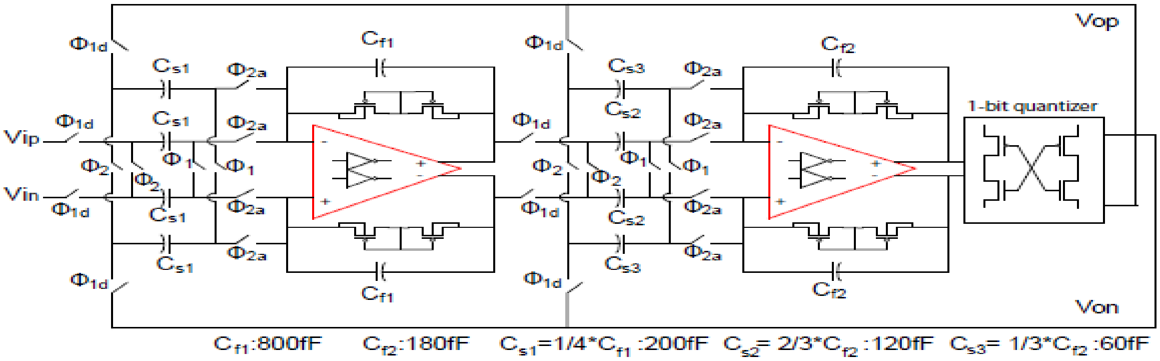

Vop
Von
Vip
Vin
Φ_{1d}
Φ_{2a}
Φ_1
Φ_2
C_{s1}
C_{s2}
C_{s3}
C_{f1}
C_{f2}
1-bit quantizer
C_{f1}:800fF C_{f2}:180fF C_{s1}=1/4*C_{f1} :200fF C_{s2}= 2/3*C_{f2} :120fF C_{s3}= 1/3*C_{f2} :60fF

Figure 2.5: Schematic of the low power high-resolution second order ∑Δ-ADC for digitizing neural data.

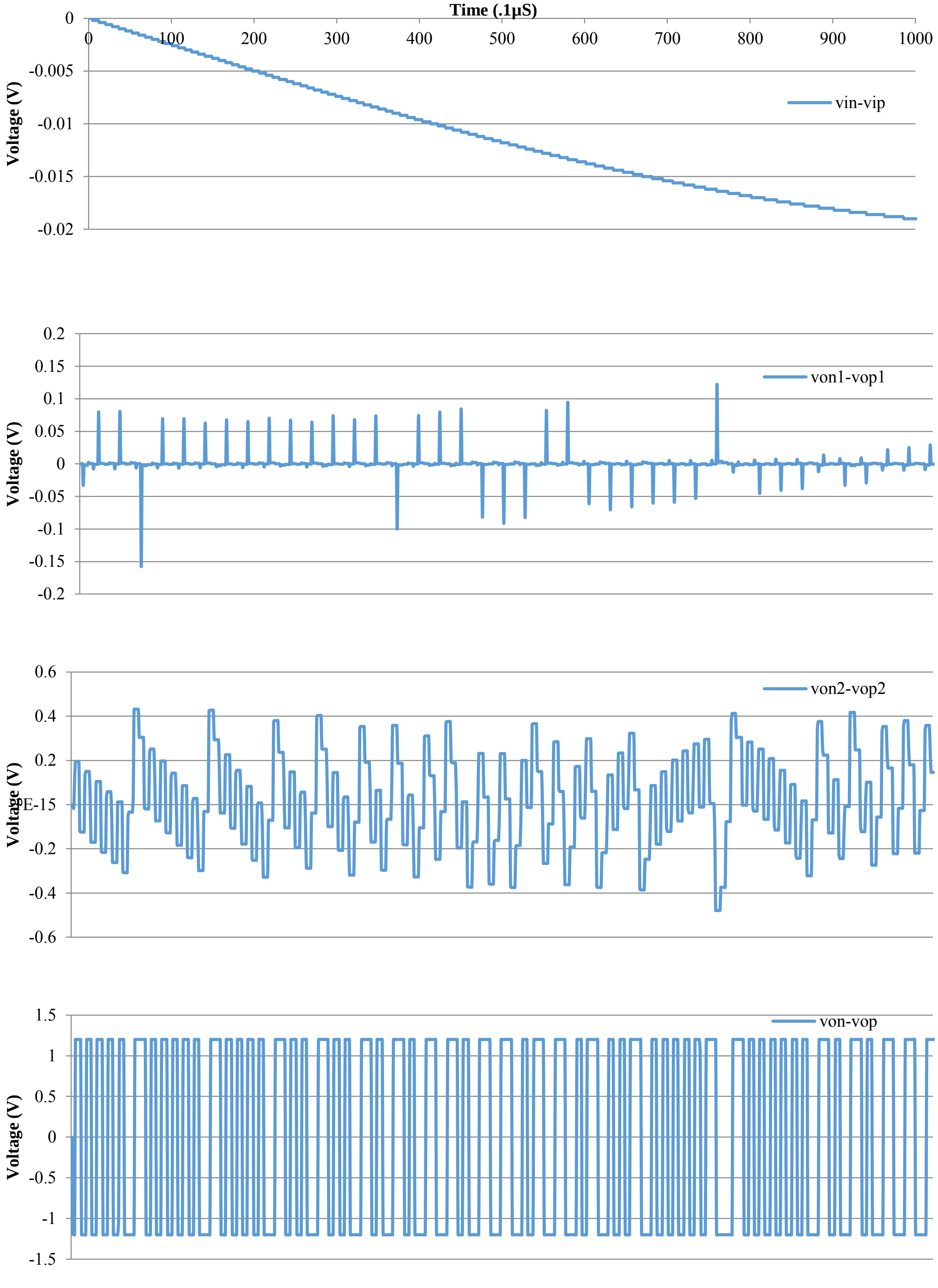


Figure 2.6: Simulation results for outputs in the ∑Δ-ADC over 100µS.

2.2.2 Verification Challenges – Converters

The signal-to-noise-ratio (SNR) is the most important aspect to verify for a converter. SNR is the ratio between the signal and noise where the higher the ratio the less prominent the noise. Noise can be introduced from many sources like process variation, coupling, or power supply variation. The clock is a which source jitter can also effect the SNR in that the sampling duration of the signal may be shorter or longer. The more noise introduced into the circuit the least reliable the conversions are. Therefore converters are designed to be as noise tolerant as possible.

Some converters have the same issue with feedback as the PLLs. Incorect conversions can impact the following conversion stages. They can also be too large to simulate as an entire system and need to be verified by sub-circuits.

2.3 Amplifiers

An amplifier is a device that increases the power of a signal. It is an essential component in almost all analog designs. For example, amplifiers can be used when a signal is too small to drive a load. There are many types of amplifiers which can be categorized as voltage or current amplifiers, transimpedance, and transconductance amplifiers. These are typically composed of single-stage amplifiers, differential amplifiers, or operational amplifiers (opamps). The differential amplifier and opamp are the most widely used amplifiers since differential inputs have higher environmental noise immunity.

Amplifiers are typically designed around a set of performance parameters, i.e. gain, bandwidth, and noise (SNR). Gain is the ratio of the output to input of power or amplitude.

Bandwidth of an amplifier is the difference between the low and upper bound of frequencies at which the amplifier produces acceptable outputs. While there are other performance parameters, these are common throughout all amplifier design.

2.3.1 Transimpedance Amplifier

A transimpedance amplifier (TIA) receives current from the photo detector and converts it to an output voltage. In the circuit in Figure 2.9, the input source is a 5mA current and the output produces a frequency. The input current charges a capacitor which activates a series of inverters connected with negative feedback. The resulting output is an oscillating waveform which dependent on the amount of applied current.

Typical design parameters for a TIA are noise, bandwidth, gain, overload response, and output impedance. The TIA in this section is designed with a frequency output behavior in mind; therefore we will analyze the amplifier in the transient domain. The behaviors of the TIA are shown in Figure 2.10 with an input current source of 5mA. The output of the TIA at the end of the inverter chain, V_{inv}, is shown in the top figure. After the first flip flop, V_{dff1}, shows a smoothed waveform in the middle graph. The final graph shows the final output after the second flip flop, V_{out}, at three different input current levels; 1mA, 5mA, and 10mA.

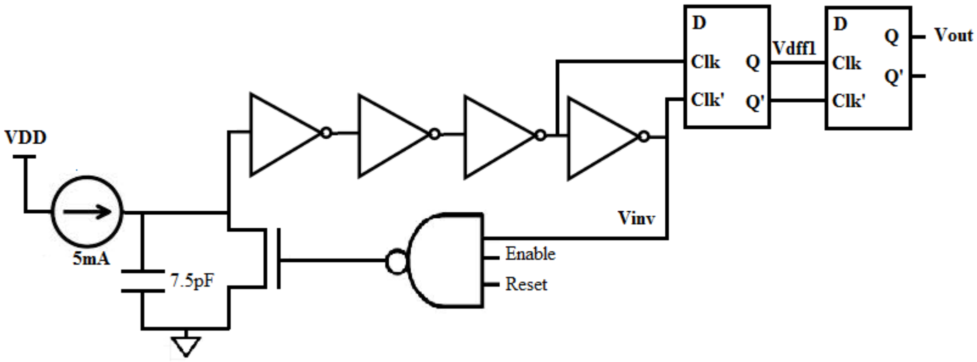


Figure 2.7: Schematic of a feedback TIA

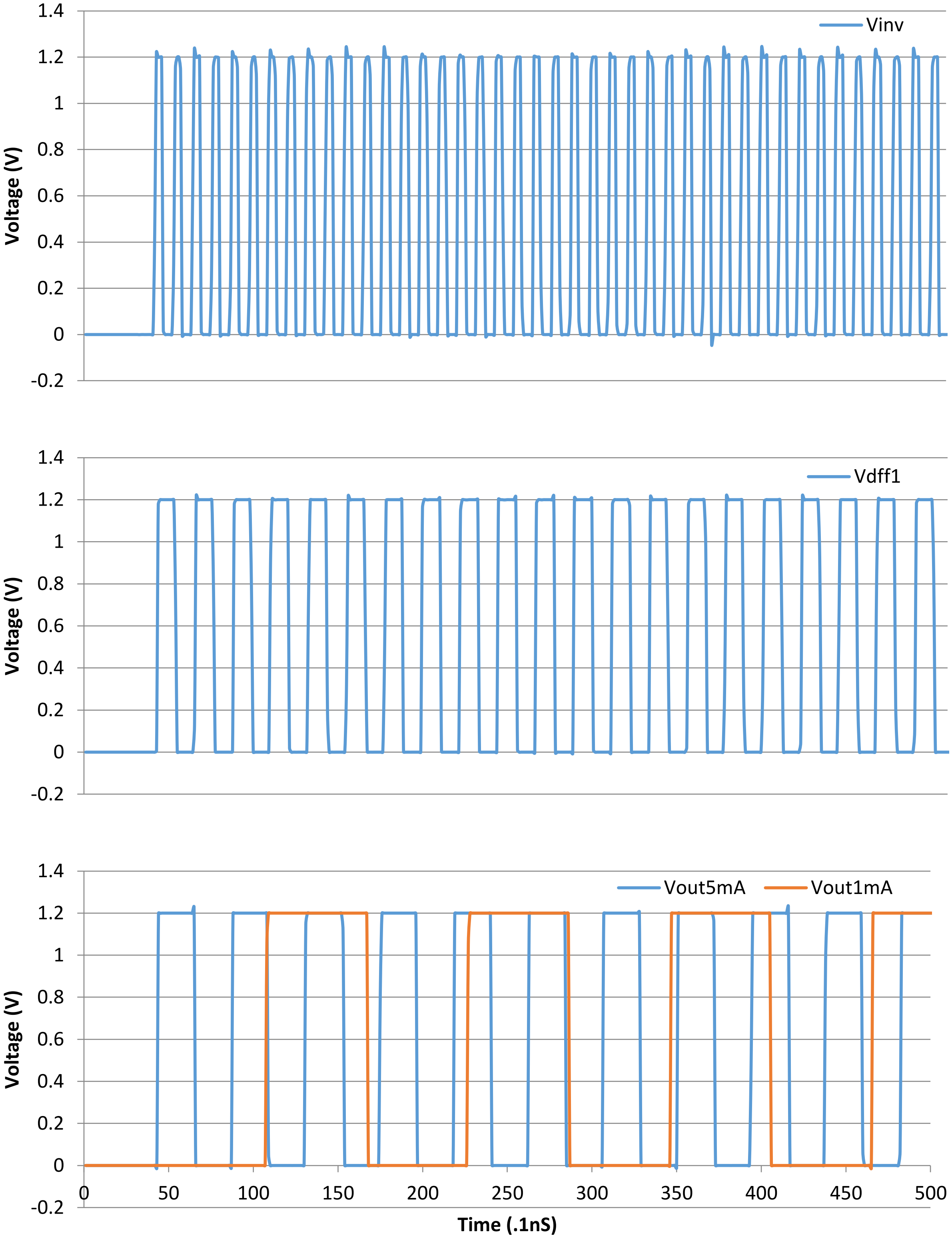


Figure 2.8: Behavior of TIA

2.3.2 Low-Noise Low-Power Neural Amplifier

The low-noise low-power neural amplifier design in Figure 2.11 is a fully differential self biased amplifier [2]. Self biasing is used to create a stable operating condition for the circuit. The bias voltages are generated using the average of two differential signals. The PMOS transistors level shift low voltages more than NMOS, while NMOS level shift high voltages more than PMOS. The input frequency has the range from 10Hz to 10MHz with a typical operating frequency of 1.6KHz.

The behavior in for the self biased differential amplifier is shown in Figure 2.12. The differential inputs are sine waves with amplitude .02V centered on .6V. The nbias and pbias signals show the time it takes for the circuit to settle into normal operating mode. This is also mimicked by the irregular amplified output. The bottom graph shows the difference between Von and Vop for an input frequency of 1KHz and 10KHz.

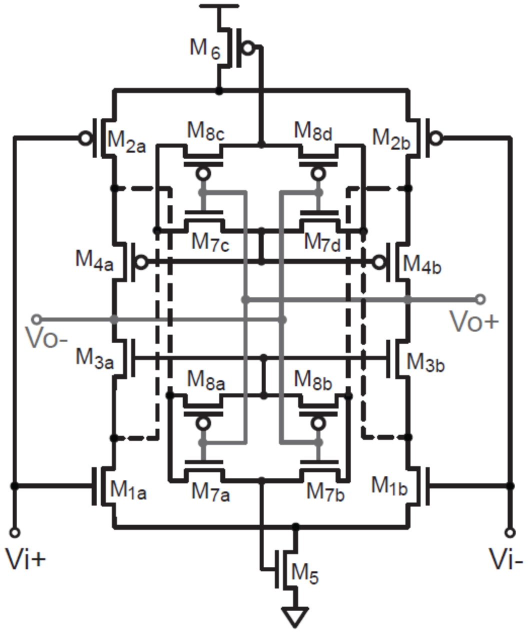


Figure 2.9: Schematic of self biased differential amplifier

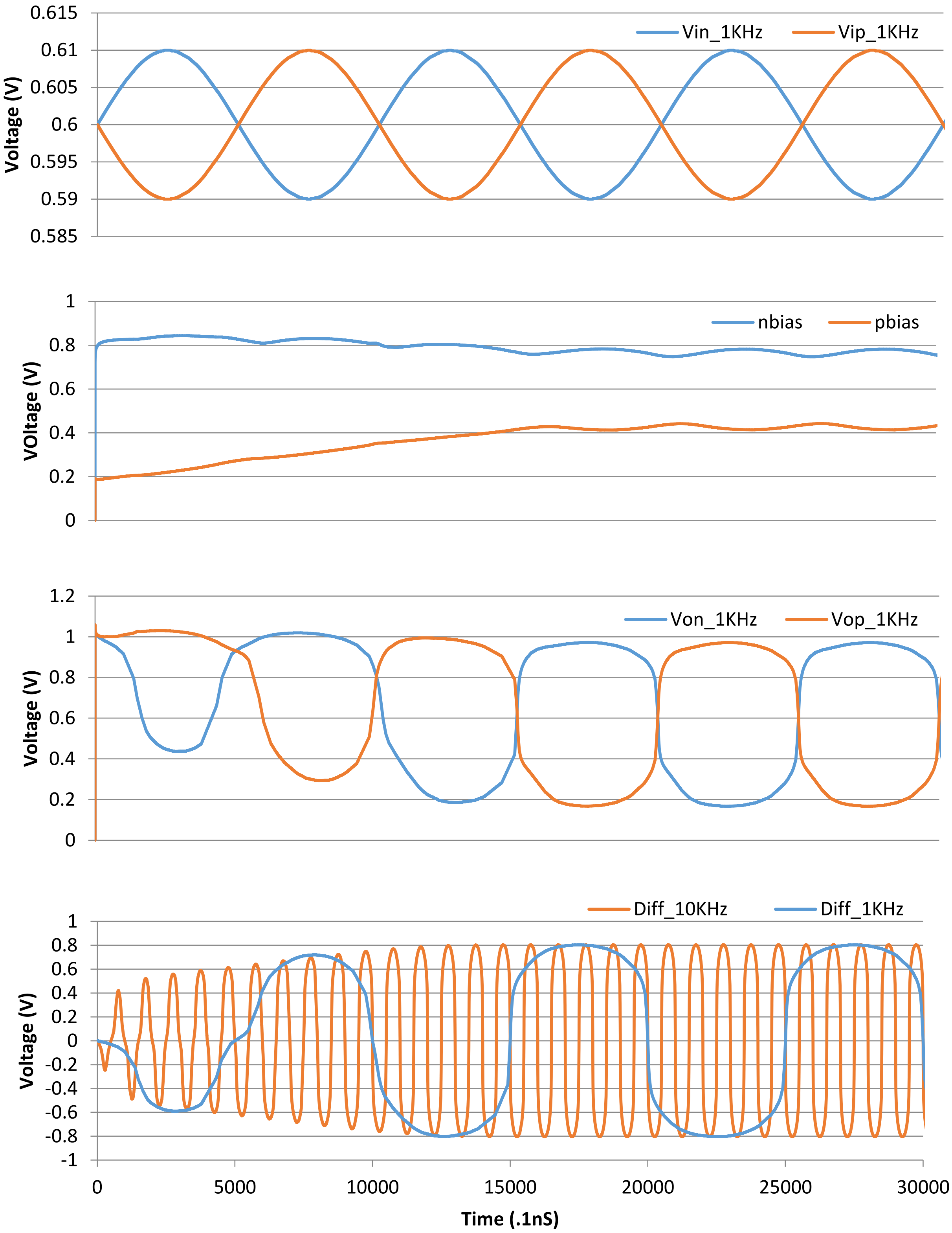


Figure 2.10: Behavior of self biased differential amplifier.

2.3.3 Verification Challenges – Amplifiers

Gain is the major performance specification of an amplifier. It is defined as the ratio of the output to the input of power or amplitude. The gain can be affected by noise sources. Noise, just like with converters, has a significant impact on the performance of the amplifier. Circuits like amplifiers are typically small in scale and therefore all devices use the same power and ground (and possibly substrate in bulk CMOS process) making coupling a serious issue. For a TIA circuit the power supply variation shows up directly at the output, causing the signal to corrupt. Mismatched differential pairs due to process variation also contribute to the noise at the output.

2.4 Voltage Regulator

A voltage regulator is a circuit that generates a fixed voltage of a predetermined magnitude. This voltage values remains constant regardless of the changes to the input voltage or load. There are two types of voltage regulators; switching and linear. Linear regulator consists of a transistor, acting as a pass device, controlled by a high gain differential amplifier. A constant voltage is maintained by comparing the output voltage with a reference voltage and adjusting the pass device accordingly. A switching voltage regulator in contrast uses a transistor as an active device which switches on or off to maintain the required output. Linear regulators are often more efficient in for low noise output, fast response to input changes, and have lower area requirement at low power. Switching regulators are typically more power efficient and at higher levels of power have a smaller area footprint.

2.4.1 Cascode Regulator

The self-biased folded cascade voltage regulator is shown in Figure 2.17 [3]. The circuit is self-biased and used a NMOS transistor as the output transistor which helps guarantee a 1.2V output. The power supply is 3.3V making the transistors think gate devices. This circuit is used as an alternative to a power supply which isolates the connecting circuits from power supply noise. The behavior in Figure 2.18 shows the noisy signal V_{in} and the smoothed signal V_{out}.

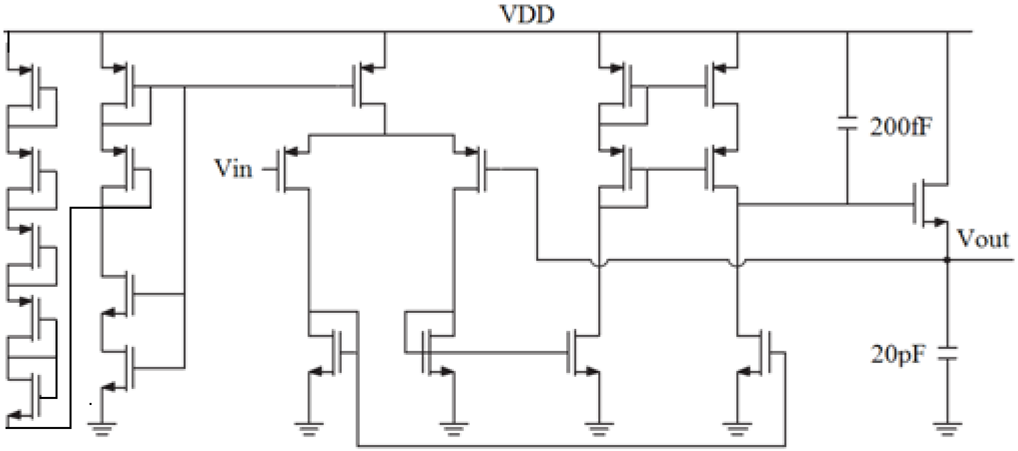


Figure 2.11: Schematic for a self-biased folded cascode voltage regulator

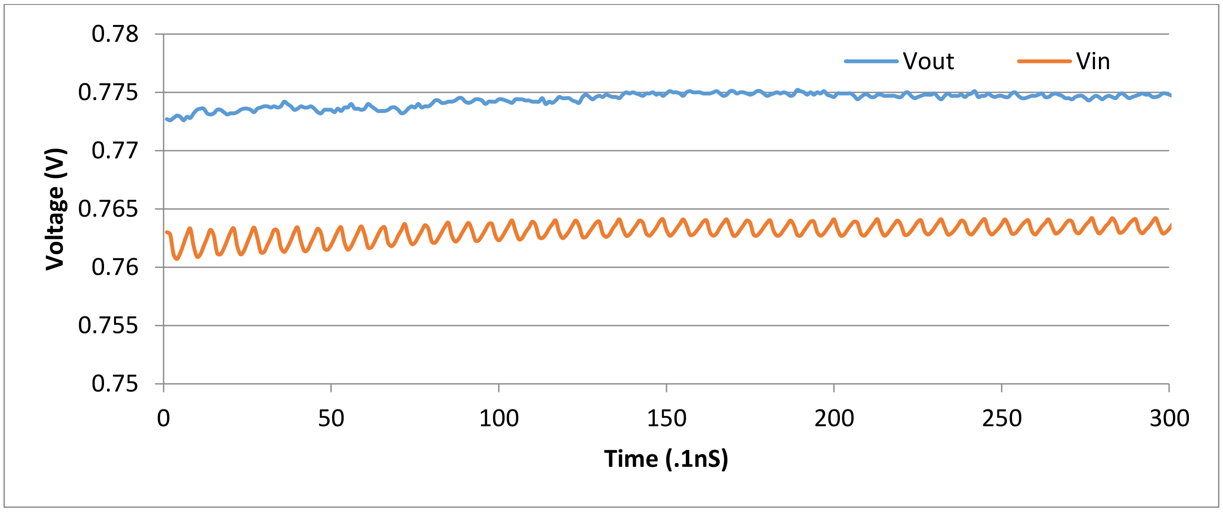


Figure 2.12: Simulated behavior of the voltage regulator.

2.4.2 Verification Challenges - Regulator

Regulators are required to produce a constant reliable voltage output. Any change to that output can cause errors in the connecting circuits. Therefore any large variation in the power supply or process will cause an unstable output voltage.

2.5 Variation

There are three main types of variation addressed in this work. The size of transistors is varied by 10% for the benchmark circuits. Process variation, in particular Vth variation, is applied to the transistors in the remaining circuits. Vth is varied by uniformly from -6σ to +6σ. We focus primarily on Vth variation because of its effect on speed and leakage power. It also has a strong correlation with temperature changes. Environmental variation, in particular, power variation is applied to all circuits in conjunction with Vth variation. The power supply is varied by ±15-20% of the optimal voltage.

2.6 Chapter 2 References

[1] B. Razavi, *Design of Analog CMOS Integrated Circuits*. Boston, McGraw-Hill, 2001.

[2] M. Elzeftawi, *Compact Low-Power Low-Noise Neural Recording Wireless Channel for High Density Neural Implants (HDNIs)*, PhD Dissertation, University of California, Santa Barbara, December 2012

[3] L. Chen, *Integrated CMOS Controller For Fast Optical Switching*, PhD Dissertation, University of California, Santa Barbara, June 2013

[4] L. Wang, *Micro Power Delta-Sigma Analog-to-Digital Converters based on Novel Self-Biased Inverter Amplifiers*, PhD Dissertation, University of California, Santa Barbara, June 2013

[5] B. Razavi. In Phase-Locking in High-Performance Systems: From Devices to Architectures. *New York: Wiley-Interscience*, 2003

[6] B. Kaminska, K. Arabi, I. Bell, P. Goteti, J.L. Heurtas, B. Kim, A. Rueda, and M. Soma, Analog and Mixed-Signal Benchmark Circuits - First Release, *IEEE International Test Conference*, Washington DC, November 1997.

Chapter 3

General Hierarchical Behavioral Modeling of Analog and Mixed Signal Circuits

This chapter focuses on the developed methodology for automatically creating transient voltage behavioral models of analog and mixed signal circuits utilizing circuit partitioning. The models are obtained using Support Vector Machines (SVM), a data dependent black box modeling technique. Larger circuits are partitioned based on their structure and intermediate behavioral models are built for high sensitivity nets. We demonstrate the soundness of this approach by modeling large circuits such as Sigma-Delta ADC and Phase Lock Loop. Experimental results show that this methodology maintains 95% accuracy behavior predictions while achieving three orders of magnitude speedup over SPICE simulation time.

3.1 Introduction

Behavioral models have become a critical step in the analysis of analog and mixed signal circuits which traditionally suffer from long simulation times. In general, behavioral modeling attempts to describe the circuit in a higher level of abstraction then transistor level, while maintaining transistor level simulation accuracy. Due to the critical effects between circuit blocks and second order device effects it is of interest to extract the behavioral models directly from the transistor level, SPICE, descriptions to ensure these interactions are properly incorporated within the behavioral models.

Many behavioral techniques model the circuit frequency domain response. The automation of linear models has been well established for simple circuits including linear time invariant [1-3] and linear time-varying systems [4]. Many fundamental non-linear effects are not captured in these equations and device models, since they are discarded by linear approximations. Addressing the need for non-linear analog macromodels, many modeling approaches have been introduced. For circuits whose non-linearity can be represented by polynomials, Volterra-based series expansions can be employed [5]. For strongly non-linear circuits, piece-wise linear models are used [6-8]. Such models divide the space into segments in which the function is linear or weakly non-linear. Each segment can be further reduced by employing model order reduction techniques. These techniques are practical for modeling transient behaviors, which is the focus of this work. Other black box modeling techniques utilizing various training-based algorithms, i.e. neural networks and support vector machines, have been proposed [9-12]. These algorithms create models by discovering the various relationships between the training data. The models can predict output values for new input values based on these relationships.

In this work, we focus on time-domain modeling for non-linear analog and mixed signal circuits which can be accomplished by equation fitting or black box modeling. Black box modeling describes a circuit or system in terms of its inputs and outputs without any knowledge of its internal workings. For this work we focus on black box modeling because they can be derived strictly from data, measured or simulated, without having to fully understand the circuit. Since they are based strictly on data, they can be generally applied to known and unknown circuit types which equations may not have been developed for.

Time-domain black box behavioral modeling methods have been proposed in the past, but they suffer either from lack of automation, provide minimal speed-up over traditional methods, or they cannot be applied to larger circuits. In [13], the authors propose to extract behavioral models directly from the netlist by processing simulation results into transfer function trajectories; however the method was demonstrated on a small circuit with minimal speedup and the approach is not always automatable. The work in [14] presents a real-time neural-network-based approach for microwave RF devices. The approach is demonstrated on small circuits and it is unclear if it can be applied to large and complex designs.

The methods proposed in this chapter build upon the work described in [15]. The authors use Gaussian Process Regression to build non-parametric models eliminating the need to use and understand complex device models and equations. Non-parametric behavior modeling is extremely attractive because there is no need to learn equations or set parameters since it predicts the device behavior based on similar known behaviors. Unfortunately, this method cannot be generally applied to large highly complex circuits, which is the inspiration for this work. For this reason we propose a partitioning method along with support vector machines (SVM) to create a hierarchy of behavioral models for large strongly non-linear analog circuits. The partitions are derived from a channel connected component graph and hierarchy graph which determines the input/output relationship and hierarchy of each partition. Simulations are run with SPICE on the full un-partitioned netlist in order to ensure continuity between partitions. Once the data has been extracted from the simulations, the black box models are created based on the derived partitions. The models are connected to form a series of predictions that create the transient output waveform. Though we focus on SVM as the primary modeling algorithm, many other black box approaches can benefit from the reduced complexity associated with the proposed partitioning scheme.

Analog circuits vary vastly in their complexity and behavior. For this reason a universal solution has not been developed for automating behavioral models. The existing modeling methods cannot be easily automated because they either (1) require design or designer knowledge to develop sub-circuit equations, (2) sub-circuits are used to speed up simulation of the model development thus the continuity between sub-blocks is difficult to maintain, and (3) input-output relationships of the functional blocks are often too complex or have one-to-many relationship making black box modeling difficult or unachievable. Analog design is usually partitioned into well-known sub-block components like quantizers, VCOs, or charge pumps which can all be modeled by equations or have data fitted to equations. Unfortunately many of these methods fail to capture block to block interactions or they cannot be expanded to include process variation due to the increased complexity of the equations. Such equations will need to be developed for every type of a new circuit or new behavior type encountered.

Building block based models can easily capture strongly correlated relationships of simple sub-circuits automatically, but fail when complex or weak relationships are encountered. To illustrate this point we analyzed a quantizer circuit defined as a single sub-block within a large sigma-delta analog-to-digital converter (ΣΔ-ADC) [16]. This circuit, (Fig.1.) contains 40 transistors and has three inputs and a differential output. Taking two common BB approaches, the response surface modeling (RSM) and support vector machines (SVM), we build transient behavioral models based on 1000 points of transient simulation data for a single period of the input frequency and test them with another set of 1000 points. Both methods predict output transient waveforms with low accuracy: RSM - 49.6% and SVM - 75.82%. This indicates that the input-output relationships are not highly correlated.

To address weakly correlated relationships, we break the netlist into functionally-independent, strongly correlated sub-circuits. Data capture for model building is done from simulations performed on the un-partitioned netlist in order to preserve the continuity between sub-circuits. The graph partitioning methodology and high sensitivity net detection removes the requirement for designer knowledge and allows for modeling unknown, flat netlists, or new circuit structures.

This chapter is organized as follows. Section 3.2 provides background on SVM modeling and data capture techniques. Section 3.3 describes the circuit graph representation, net sensitivity calculations, and modeling-resistant circuit components. Section 3.4 describes circuits used in experiments. Section 3.5 shows experimental results for nominal behavior. Section 3.6 presents variation based models and modeling results. The key learnings are analyzed in Section 3.7 and the chapter is concluded in Section 3.8.

3.2 SVM Background

Within the domain of behaviors studied here, PWL or PWP (piece-wise polynomial), and SVM models are comparable in speed and accuracy. We choose to use SVM-based models because of their ability to handle a large number of model inputs and to discover trends based on small or irregular sample sets. Data mining and learning-based approaches have been developed to predict performance specifications of analog circuits [9-12]. These models use simulation data and employ various techniques such as SVM or Neural Networks. We utilize such modeling techniques in this work, but apply them to transient behavioral modeling.

Our models are built using the supervised learning algorithm, *Support Vector Regression* (SVR) [17]. The objective of supervised learning is to derive a function from a set of input samples (*training set*) and their associated outputs. Support Vector Machines (SVMs) are a family of algorithms for classification and regression applications. SVMs may require long training time for complex data sets, but there exist sampling and data partitioning methods that can be used to create smaller, more compact models which significantly reduces the training time. The work in [18] explores various methods for data sampling to achieve high SVM model accuracy with fewer simulation runs.

3.2.1 SVM-R Theory

Support vector regression [17] can predict real number values which are common in analog circuits. The digital values can be predicted using classification which will not be discussed in this work. Consider a set of training data points $\{(x_i, y_i),_{i=1}^{n}\}$, where x_i represents the input vector and y_i represents the corresponding output value. The support vector regression function can be expressed as

$$y = w * \Phi(x) + \ b \ , \tag{2.1}$$

where b is the bias, w is the weight, and $\Phi(x)$ denotes the feature of the inputs. The optimal regression function is obtained by minimizing the risk function,

$$\frac{1}{2}\left|\left|w\right|\right|^2 + C\sum_{i=1}^{n}(\xi_i + \xi_i^*) \tag{2.2}$$

$$\text{Subject to} \quad y_i - [(w, x_i) + b] \leq \ \varepsilon + \ \xi_i$$

$$[(w, x_i) + b - y_i \leq\leq \ \varepsilon + \xi_i^*$$

$$\xi_i \geq 0, \xi_i^* \geq 0$$

where *C* is the regularization constant, *ε* denotes the *ε*-insensitive coefficient, ξ_i, ξ_i^*denote positive slack variations. By using Lagrange multiples β_i, β_i^* and kernel function *k(x_i,x_j)*, the dual Lagrange form is given as:

$$\text{Max} \sum_{i=1}^{n} y_i(\beta_i - \beta_i^*) - \varepsilon \sum_{i=1}^{n}(\beta_i - \beta_i^*) \tag{2.3}$$

$$-\frac{1}{2}\sum_{i=1}^{n}\sum_{i=1}^{n}(\beta_i - \beta_i^*)\left(\beta_j - \beta_j^*\right)k(x_i, x_j)$$

$$\text{Subject to} \sum_{i=1}^{n}(\beta_i - \beta_i^*) = 0$$

$$0 \leq \beta_i, \beta_i^* \leq C$$

The regression function can now be expressed as,

$$y = \sum_{i=1}^{n}(\beta_i - \beta_i^*)\, k\left(x_i, x_j\right) + b. \tag{2.4}$$

The kernel function, *k*, is used to measure similarity between two vectors in the given feature space. The kernel computes the vector distance between *x* and *x*' without ever explicitly mapping the vectors to the feature space *Φ*, reducing computational costs and allowing data to be linearly separable in the original space

$$k(x, x') =< \Phi(x), \Phi(x') > . \tag{2.5}$$

In this work we use the Gaussian kernel expressed as

$$\mathrm{k}(\mathrm{x},\mathrm{x}') = \exp\left(-\frac{\left\|\mathrm{x}-\mathrm{x}'\right\|^2}{2\sigma^2}\right), \text{where } \sigma > 0\ . \tag{2.6}$$

Other kernels, such as linear, polynomial, or sigmoid can be used, but experimentally, we found that the Gaussian kernel provides the most consistent positive results for the applications considered here.

3.2.2 SVM Creation

The SVM algorithm utilizes data obtained from low level models such as SPICE or differential equations to create a high level model abstraction that captures a specific behavior. A circuit can be completely described by a set of differential equations

$$\frac{d}{dt}\vec{q}\big(\vec{u}(t)\big) + \vec{f}\big(\vec{u}(t)\big) + \vec{B}\big(\vec{x}(t)\big) = 0, \tag{2}$$

$$\vec{y}(t) = \vec{l}^T\vec{u}(t) + \vec{d}^T\vec{x}(t),$$

where $\vec{u} \in \mathbb{R}^n$ are state variables, i.e. capacitor current, $\vec{x}$ are top level inputs, and $\vec{y}$ are the top level outputs. Obtaining a unique output $\vec{y}$ requires access to the state and input variables. At a high level of abstraction, black box, the internal state variables are no longer accessible. Therefore unless the inputs $\vec{x}$ are highly correlated to the outputs $\vec{y}$ the models will produce very low accuracy results. Access to $\vec{u}$ is required to produce high accuracy models. In order to determine which state variables are required we propose a heuristic which partitions the circuit and determines highly correlated intermediate behaviors to the top level output. Simulations are performed in order to capture the behavior relationships between inputs $\vec{x}$ and the intermediate behaviors are captured in SVM models. The output $\vec{y}$ can be determined based on top level input $\vec{x}$ and intermediate models.

3.3 Model Creation

Given the design specifications, input operation ranges for the desired performance, and the circuit netlist, the required data for building behavioral models can be obtained. In this work, we only consider transient voltage behaviors. Before any partitioning is done the entire netlist is simulated as a single entity and the transient voltage behaviors are captured for each net at uniform intervals. We perform uniform sampling within the range of the design specifications and simulate to capture the behavior. For example, if the design should operate between 1V V_{DD} and 1.5V V_{DD} we would run simulations at 0.1V intervals. Once the data is captured, partitioning is performed in order to create a series of simple models to predict the behavioral operation at any V_{DD} between 1V and 1.5V significantly faster than through simulation.

The uniformity of samples may cause model inconsistencies around highly volatile areas which traditionally require increased sampling. It is not known beforehand whether more data is needed for specific areas or for intermediate behavior models. To address this problem, data is captured at finer intervals then required in the design specifications, which does not increase simulation time. The models are initially built based on the original intervals. If testing step discovers inaccuracies in a specific model, the models are rebuilt based on the finer interval data. This data capture method is rarely required; it may be invoked when the output behavior is state dependent and highly volatile. When capturing smaller intervals, the waveform changes become less extreme, making the behavior relationships easier for the SVM algorithm to discover.

Probes are inserted at each edge defined by the partitions in Section III for voltage capture. This way the system level input and output relationships of each partition are maintained without any extra modeling for block coupling. This implies that our models are specific to the simulated design and cannot be utilized in a different design model unless the input and output relationships between the designs are maintained.

3.3.1 SVM Model

Due to the decomposition of the circuit each partition has a strong correlation between the input set and the output resulting in a weight vector having a non-zero value for each feature. The number of partitions per circuit is generally high for large analog circuits. The classification models for the digital partitions remove most of the real number errors in the regression models by predicting the digital values 0 or 1.

State and feedback characteristics of the circuits can easily be incorporated into the vectors. The previous output value of the circuit being modeled is included as a feature in the vector describing the current output value. States are incorporated in the same way, as features within the vector. When applying test vectors to the model, the previous predicted output value is included within the current test vector.

Parameter selection is a difficult problem for many learning algorithms and is crucial for creating a good model. In this work we use the υ-SVM-R algorithm which automatically fits the user-defined values ε and C based on the provided data. The only value the user defines is the variable υ which determines the slack variables. The larger the υ value the more vectors with slack ξ may become support vectors. In this work we use $\upsilon=0.1$ for all regression models.

3.4 Partitions and Intermediate Behaviors

A common approach to addressing slow transistor-level simulation times is to partition the circuit into its digital and analog components. A major challenge in this type of partitioning is to determine when the digital components are in analog modes of operation, which include detecting of the internal and negative feedbacks. The work in [18] provides an algorithm for such partitioning using channel-connected sub-circuits. In [19], the analog sub-circuits are simulated using nonlinear macromodels. The macromodels approximate the charging and discharging behavior of the circuit output nodes and provide one order of magnitude speedup over SPICE simulations for the analog subcircuits. The macromodeling using circuit partitioning proposed in [20-23] decomposes the circuit into a set of building blocks based on the netlist structure. Methods that use subcircuit or building block-based partitions assume that either the user declares the partitions or that there is a set of predefined building blocks. It is not always the case with highly complex or custom built circuits which makes modeling using these methods difficult.

In our approach, the graph model of the circuit and sensitivity analysis of captured data are used for establishing the signal flow, hierarchy, and determining which inputs are essential to predict intermediate behaviors. In this Section we discuss the two types of graphs used in partitioning and explain how the intermediate nodes are determined based on net sensitivity. We then discuss circuits which are resistant to this type of partitioning and propose a method of handling them. Using the partitioning and intermediate nodes we show how accurate this method is at predicting the final output as compared with generic modeling of the component which yielded 75% accuracy for SVM and 50% accuracy for PWL.

3.4.1 Channel Connected Graph and Channel Connected Components

A Channel Connected Graph (*CCG*) describes the source-drain dependencies within a circuit. It is defined as *CCG=(V,E)* where *V* are vertices which correspond to transistors, V_{dd} or *GND,* and *E* are edges which capture the source-drain connections. There is an edge between vertices v_1 and v_2 if their corresponding transistors have drains or sources connected or if one of them represents V_{dd} or *GND* and the other transistor connected to it. Once the V_{DD} and *GND* nodes are removed, *CCG* is fractured into a set of smaller graphs. Each such a sub-graph with V_{dd}/*GND* and connections to them restored corresponds to a channel connected component graph (*CCCG*).

Figure 3.1 shows the *CCG* graph of a basic two stage opamp. The labels on the graph nodes and their corresponding transistors match. The graph has two partitions. Partition 1 consists of nodes 1-6 while partition 2 consists of nodes 7-9. Both partitions include V_{dd} and *GND* nodes along with the corresponding edges.

In analog circuits, when the V_{DD} and *GND* nodes are removed to create the components, and later restored in sub-graphs, it is possible that some resulting components do not contain V_{DD} or *GND.* Such sub-graphs are not *CCCG*s and may occur when a circuit contains floating nets, resistors, capacitors, or inductors that supply connections to the source or drain terminals. Partitions which are not *CCCG*s are discussed in part *D* of this Section.

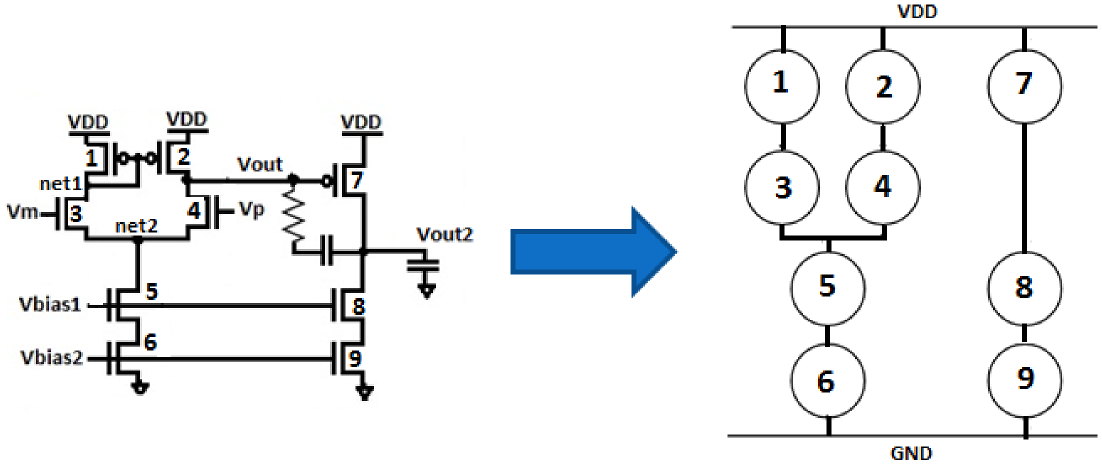


Figure 3.1: Differential opamp schematic and its CCG

3.4.2 Group Hierarchy and Feedback Detection

Each partition within the *CCG* has an associated set of driving gate inputs which can be the top level inputs to the circuit, internal nets, or nets from other partitions. Referring to Figure 3.1, partition1, containing nodes 1-6, has the associated input set $\{V_m,V_p,net_1,V_{bias1},V_{bias2}\}$; partition2, containing nodes 7-9, has the input set $\{V_{out},V_{bias1},V_{bias2}\}$. The inputs $V_m,V_p,V_{bias1},V_{bias2}$ are top level inputs, net_1 is an internal input, and net V_{out} is an external input. The relationships between partitions can be described by a directed graph HG=(N,L) where N is the set of partitions, top level inputs t, or primary outputs; and L are edges. An edge $l(t,b)$ exists in HG, if there is an input to b from the top level t. An edge $l(a,b)$ exists if the output of a partition a is an input to the partition b. Self-loops, edges with the same head and tail nodes, $l(a,a)$, indicate internal inputs.

The graph HG determines the circuit hierarchy. The first level of the hierarchy contains t and those nodes with no in-coming edges other than self-loop. The remaining hierarchical levels can be determined by performing breadth-first traversal of the graph HG. The partitions containing self-loops indicate state dependency due to the feedback cycle between the gate-drain or gate-source connection.

Figure 3.2 depicts the hierarchy graph for the opamp in Figure 3.1. The internal input net1 is shown as edge *l*(*1,1*). V_{out} is edge *l*(*1,2*) indicating an external input. Node 1 contains no external edges, only top and internal inputs, making it the top of the hierarchy. Node 2 is on the second level due to external edge *l*(1,2).

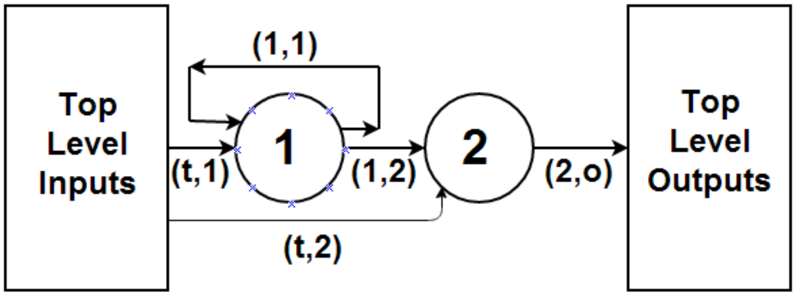


Figure 3.2: Hierarchy graph of differential opamp in Fig. 1

3.4.3 Intermediate Net Detection

Intermediate behavioral nodes are determined based on *CCCGs*, hierarchy graph, and sensitivity analysis performed on initial simulation data. To determine if a partition contains any high sensitivity nets, the calculation of $\Delta V_{net}/\Delta V_{out}$ is performed on the net voltage data generated from circuit simulation. If there are no high sensitivity nets within a specific *CCCG*, then that group is combined with the next group in the hierarchy. The exception is if the group contains an external edge to the top level output *l*(*b*,*o*). If a partition contains a high sensitivity net then the intermediate behaviors are the nets whose edges correspond to external inputs in the hierarchy graph. The opamp design contains one high sensitivity net based on the simulation data, thus the intermediate behavior of V_{out} is modeled. If no high sensitivity nets were found then both partitions would be combined into a single node that utilizes previous state information, due to edge (1,1), and top level inputs.

The high sensitivity nets are determined based on the high correlation to the output behavior where each partition's inputs are highly correlated to the high sensitivity net. The model of each partition is based on simulation data which encompasses the behavior changes occurring due to internal variables $\vec{u}$. Each model then captures behaviors due to internal variables and represents functions dependent on only previous model outputs or top level inputs. The top level output behavior is then predicted based on few highly correlated internal nodes and the top level inputs.

3.4.4 Partition Resistant Components

Analog circuits are composed of many different types of components, all combined into various configurations. In certain instances the circuit or subcircuit may not form a *CCCG;* such a partition may be missing the V_{DD} or *GND* node. Such a circuit is considered resistant to partitioning and must be modeled based on the circuit inputs and outputs without any intermediate nodes. For example, a switch capacitor network does not form a *CCCG* because the source or drain terminals of the nodes are connected only via capacitors which are excluded from the construction of the *CCG*.

The structures resistant to partitioning are components within a library. When at the netlist partitioning stage such circuits or subcircuits are detected, then no further partitioning is performed on them. Their models are created using the circuit inputs and outputs without any intermediate nodes.

The switch capacitor circuits need to be detected from the netlist during partitioning. There is a number of possible topologies that make up these components, Figure 3.3.

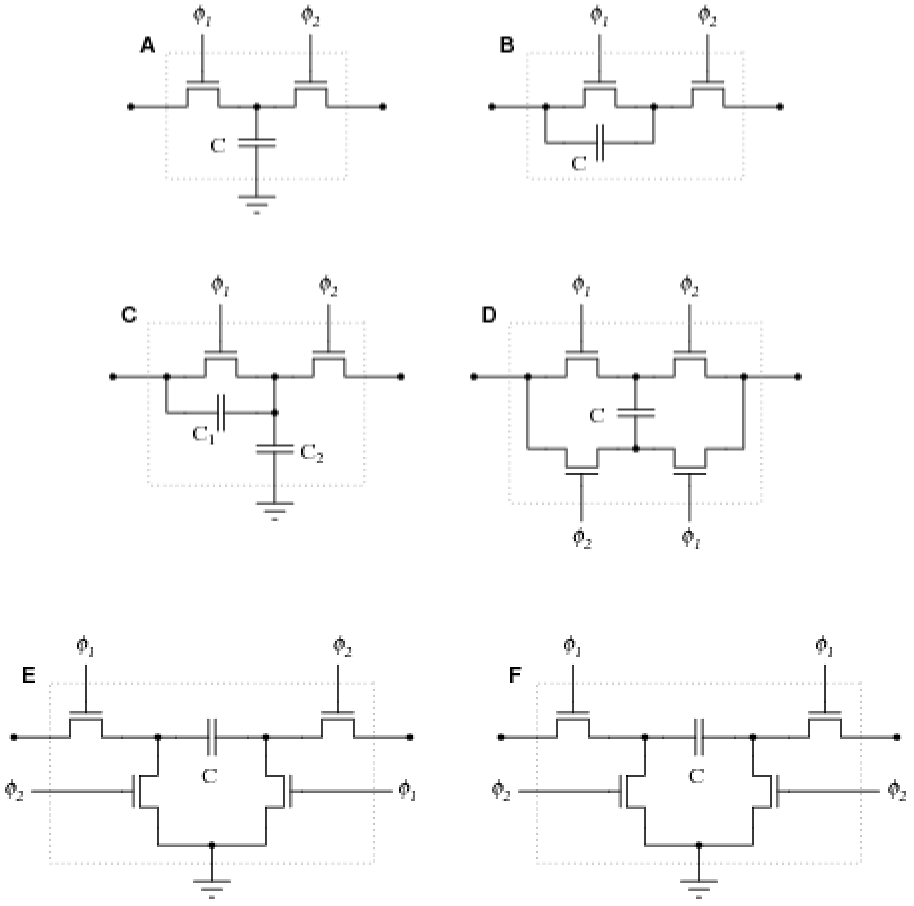


Figure 3.3: Examples of switch capacitor (SC) topoglogies

Algorithm 1: Netlist to SVM Model methodology

Input: Circuit netlist, model files, design inputs D=[d1,...,dn], transient time T

Output: SVM behavioral models

```
1:  Check for known circuit topologies and set SC_flag to true if one or more exist, otherwise false
2:  Create CCG=(V,E)
        a. if SC_flag==true; all SC components are clustered in same group
3:  Create hierarchy HG=(N,L)
4: Modify netlist to include T, D, and data extraction statements for capturing all net voltage behaviors
5:  Run SPICE simulation
6:  Parse simulation output file into training and testing files based on hierarchy graph
        a: Starting from top level of hierarchy,
        b: For each level of hierarchy and top output accuracy <95%
                i: If output net of hierarchy node is digital run SVM-Classification
                ii: Else run SVM-Regression
                iii: if current hierarchy <95% accuracy create intermediate node SVM model
7:  Create dependent model  chain
8:  Apply top level inputs
9:  Analyze predicted top level output
```

3.4.5 Quantizer Example

Figure 3.4 shows the *CCG* for the quantizer circuit and highlights the sensitive nets and components. The top of the hierarchy, as shown in Figure 3.5a, is the input level to the circuit and leafs contain the outputs. In Figure 3.5b, we show the reduced hierarchy graph for the quantizer in which the levels labeled Group 1 and Group 2 are merged together. Group 1 does not contain any high sensitivity nets and it is combined with the next lowest level. Such modifications of the hierarchy graph reduce the number of models needed to be created. The high sensitivity nets are modeled and then used as input variables to the next group lower in the hierarchy.

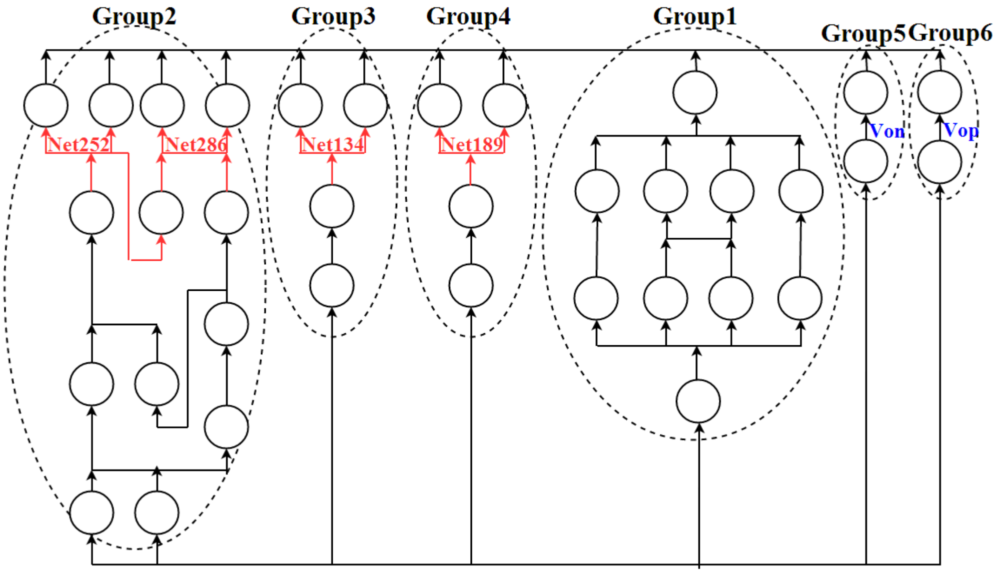


Figure 3.4: Channel Connected Compont Graph of the quantizer circuit. Six group partitions with intermediate behaviors

By partitioning the circuit into a set of connected subcircuits we can predict the output using intermediate signals which leads to the final V_{on} and V_{op} digital values. Table 3.1 shows the six models created for the quantizer circuit and their prediction accuracy. The models were trained with 1000 vectors and tested with 1000 different vectors. For the intermediate nodes or final outputs producing real values, support vector regression was used. The input vector consists of circuit inputs and previous group outputs where applicable as shown in Figure 3.5(a) and (b). Table 3.2 shows which outputs were predicted within each group whether high sensitivity nets or final outputs. Nets 286 and 252 are inputs to the groups predicting nets 134 and 189 which are in turn used as inputs to the groups predicting V_{on} and V_{op}. The resulting outputs have 100% prediction rate which is significantly better then the results prior to partitioning at 75.82%.

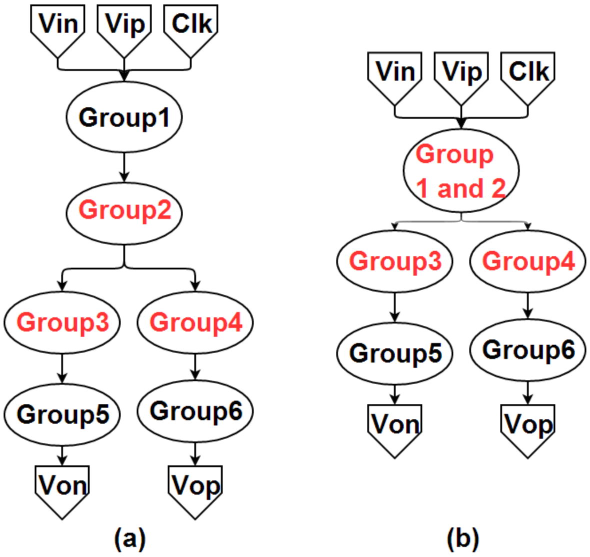


Figure 3.5: (a) The initial hierarchy graph of the quantizer. (b) Hierarchy graph with combined non-sensitive groups Group 5 and 6 contain output edges.

TABLE 3.1: RESULTS FOR THE QUANTIZER PARTITIONS

	Training Time	Prediction Accuracy
Net286	.24s	99.4599%
Net252	.234s	99.54%
Net134	.18s	99.94%
Net189	.18s	99.94%
Von	**.18s**	**100%**
Vop	**.17s**	**100%**

3.5 Experiments

The partitioning, data capture, and net sensitivity calculations described in the earlier sections of this chapter have been implemented as a fully automated tool and demonstrated on the UWB-PLL and the ∑Δ-ADC described in Chapter 2. The data capture is done via HSPICE [23] and the SVM models are created using LIB-SVM [24]. Each circuit has an associated set of design specifications which indicate the input parameters correlated with the desired operational behavior. In each experiment, the transient voltage behavior is modeled. The model accuracy is determined by the predicted values compared to the actual simulated values.

Each netlist is modified to include the transient statements and voltage capture for each net. In parallel with simulation, the hierarchy of the defined subcircuits is determined and partition graphs are created for each subcircuit. The data file generated from the simulation run is parsed and a subset of the data is used to quickly determine if the subcircuits require further partitioning starting from the topmost level of the hierarchy until the desired accuracy is achieved. Once the level is determined then the models are created using the full simulation data. The models are then chained together to form the full path to predict the final circuit outputs. Those models within the same level of hierarchy can be determined in parallel.

Table 3.2 shows the models created for the ∑Δ-ADC and the amount of training time, number of support vectors, and the prediction accuracy when the models were tested independently and dependently (within the circuit). The models were trained with 50,000 vectors, one input period, and tested with 50,000 different vectors. When the models are tested for their accuracy independently of any other models the desired accuracy is very high. When the models are chained together and are dependent on one another, there is some loss in accuracy due to miss-predictions being propagated through the chain. This error is minimized by the weights discussed in Section IIC. The total amount of time to perform 500,000 predictions, 50,000 per model, is 184 seconds. The speedup over HSPICE is only 21X because of the high dimensionality of the amplifier models, with each model having 15 dimensions. Figure 3.6 shows a subset of the dependent predictions for the Amp_V_{op2} model where V_{op2A} is the actual simulated waveform and V_{op2} is the predicted waveform.

Table 3.3 shows the models created for the PLL with the amount of training time, number of support vectors, and the prediction accuracy when tested independently and dependently. The models were trained with 10,000 vectors and tested with 20,000 different vectors. As with the ∑Δ-ADC the independent model predictions are very high. The training time is significantly shorter for the PLL due to its continuous nature and smaller dimensionality. Each partition in the PLL contained fewer elements making the intermediate and input-output relationships less complex. The total amount of prediction time for 380,000 vectors is 15.76 seconds which is a 445X speedup over HSPICE simulation. Figure 3.7 shows the waveform comparison between the simulated F_{out} and the predicted waveform F_{out_P} for the first 200 predictions, 20ns.

Significant speedup can be observed when modeling designs that include layout parasitics. For this experiment the UWB-PLL in Figure 2.1 includes parasitics extracted from layout. The simulation time in HSPICE for this netlist with 60830 components took 7 hours and 8 minutes. Table 3.3 shows the model creation for the UWB-PLL data and the prediction accuracy for independent and dependent models. To perform all 380,000 predictions, 20,000 test vectors per model, the total prediction time was 14.54 seconds a 1766X speedup over HSPICE. Figure 3.8 shows the waveform comparison between the simulated F_{out} and the predict waveform F_{out_P}

TABLE 3.2: BEHAVIOR MODELS FOR THE ∑Δ-ADC

Model Name	*# Support Vectors*	*Training Time - Minutes*	*Prediction Accuracy -Independent*	*Prediction Accuracy - Dependent*
Amp_Von1	10599	8	99.13%	98.01%
Amp_Vop1	10793	8	99.15%	98%
Amp_Von2	9383	8	99.21%	97.57%
Amp_Vop2	9097	8	99.01%	97.53%
Net286	683	2	99.67%	99.06%
Net252	683	2	99.78%	99.01%
Net134	43	1	99.48%	98.77%
Net189	41	1	99.47%	98.98%
Von	**19**	**1**	**99.98%**	**98.82%**
Vop	**20**	**1**	**99.98%**	**98.82%**

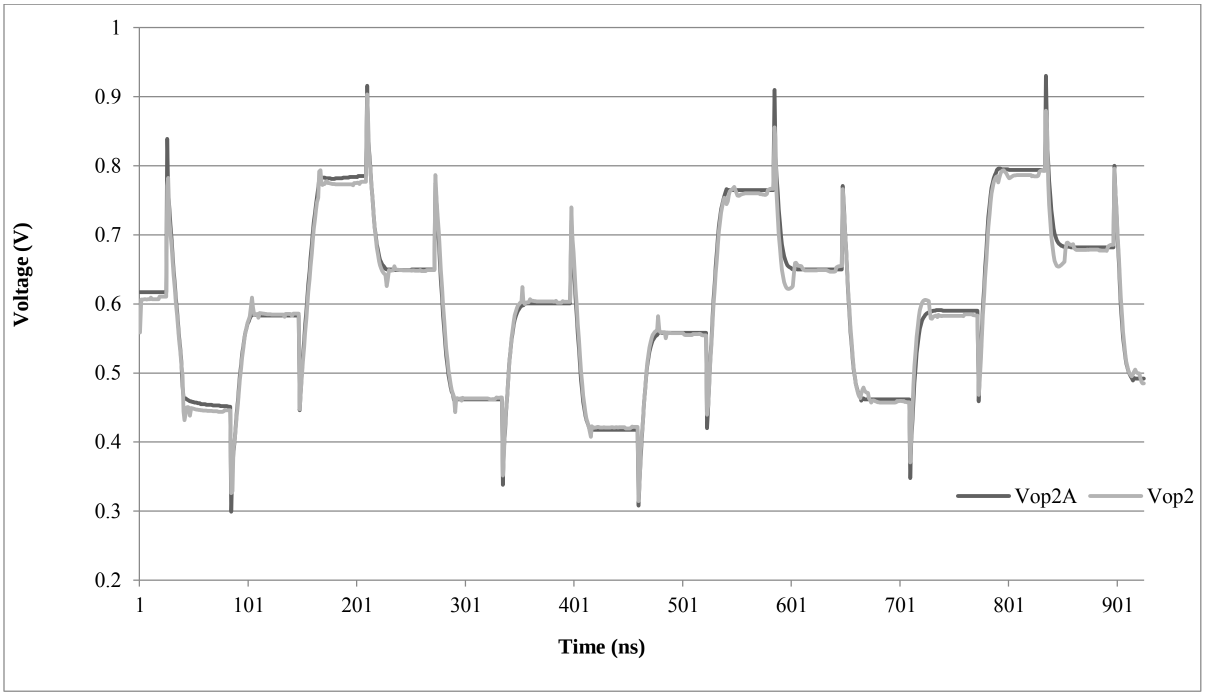


Figure 3.6 Waveform comparision between simulation waveform, Vop2A, and predicted waveform, Vop2.

TABLE 3.3: BEHAVIOR MODELS FOR THE UWB-PLL

Model Name	*# Support Vectors*	*Training Time -Seconds*	*Prediction Accuracy -Independent*	*Prediction Accuracy - Dependent*
PFD_UP	115	.21	98.77	99.64%
PFD_UPi	111	.28	99.19	99.63%
PFD_DN	106	.18	99.677	99.19%
PFD_DNi	107	.24	98.95	99.11%
CP	489	2.65	99.9	95.31%
DIVN_S1	180	.18	100	98.55%
DIVN_S2	162	.17	100	99.3%
DIVN_S3	1176	.81	94.02	98.86%
DIVN_S4	1154	.78	95.2	99.42%
DIVN_S5	490	.36	97.58	99.11%
DIVN_S6	245	.21	99.24	99.12%
DIVN_S7	131	.14	99.53	99.32%
REG_VUWB	1003	6.99	99.9	94.95%
VCO_NET41	639	2.7	98.7	96.17%
VCO_Vn	1511	3.4	99.45	96.68%
VCO_Vp	1512	3.3	99.48	96.68%
VCO_Von	1512	3.3	98.6	97.56%
VCO_Vop	1509	3.4	98.7	97.58%
VCO_fout	**1008**	**2.5**	**98.7**	**97.12%**

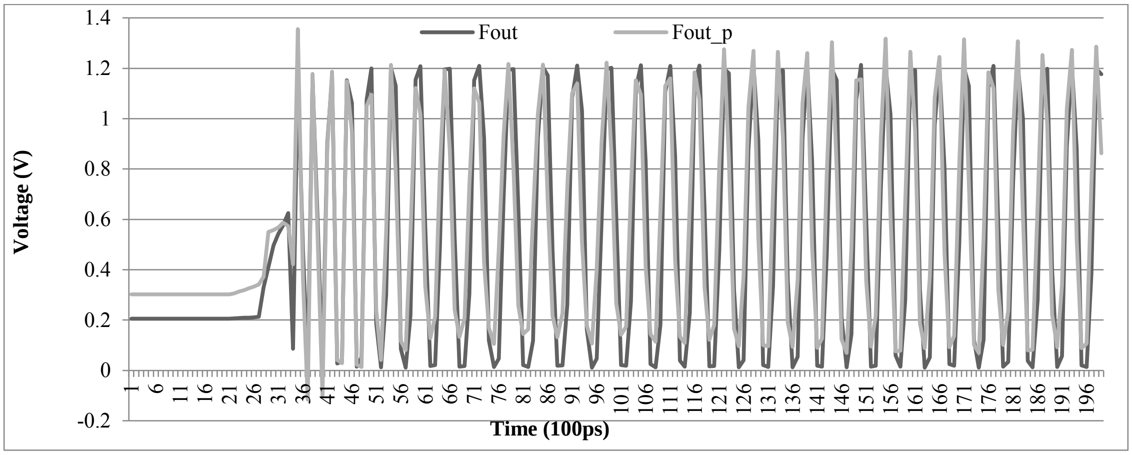


Figure 3.7: Waveform comparision between simulation waveform, Fout, and predicted waveform, Fout_P for the PLL_UWB

TABLE 3.4: BEHAVIOR MODELS FOR THE UWB-PLL WITH PARASITICS

Model Name	*# Support Vectors*	*Training Time -Seconds*	*Prediction Accuracy -Independent*	*Prediction Accuracy - Dependent*
PFD_UP	88	.266	99.63	98.98
PFD_UPi	94	.192	99.76	99.01
PFD_DN	94	.186	99.64	98.78
PFD_DNi	90	.185	99.79	98.78
CP	518	4.41	99.8	96.8
DIVN_S1	180	.18	100	98.66
DIVN_S2	162	.162	100	99.23
DIVN_S3	886	.637	96.16	99.33
DIVN_S4	366	.294	98.5	99.63
DIVN_S5	230	.203	99.32	99.47
DIVN_S6	123	.137	99.27	99.48
DIVN_S7	71	.103	99.63	99.45
REG_VUWB	1018	6.7	99.8	98.7
VCO_NET41	1027	6.81	99.5	98.66
VCO_Vn	1519	4.33	99.35	96.2
VCO_Vp	1523	4.32	99.32	96.2
VCO_Von	1511	4.1	97.78	97.12
VCO_Vop	1516	4.2	97.8	97.24
VCO_fout	**1012**	**.81**	**96.35**	**95.1**

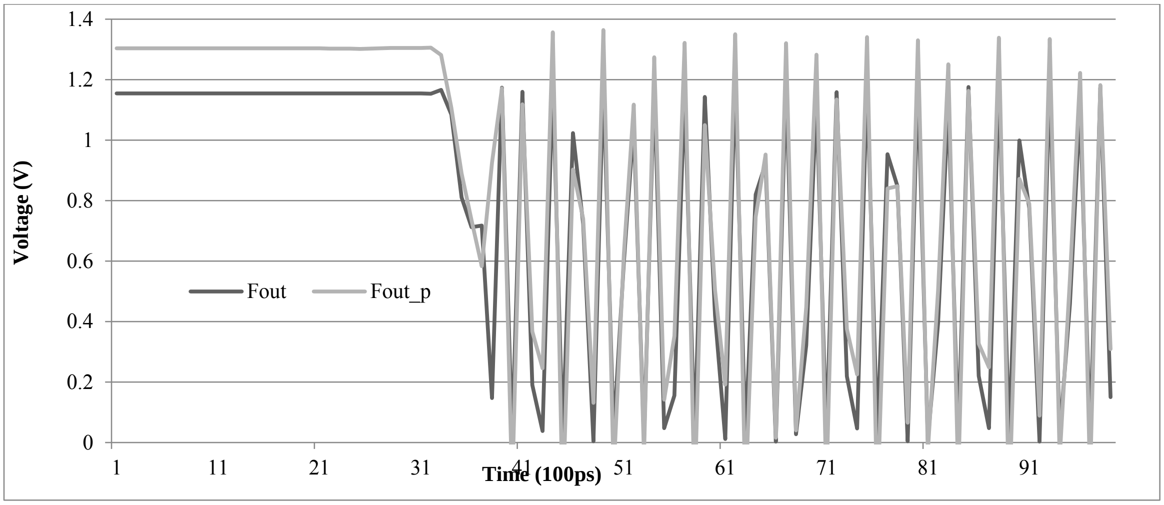


Figure 3.8: Waveform comparision between simulation waveform, Fout, and predicted waveform, Fout_P for the UWB-PLL with parasitics

3.6 Variation Models

Variation based behavioral models are an extension of the behavioral modeling algorithm described in the previous sections. Automatic decomposition and partitioning of the circuit remains the same, while the training set is expanded to include new variation data. The netlist file is altered to include Monte Carlo sweep statements for each process parameter being varied. For each combination of voltage supply and temperature values, 20 Monte Carlo simulations are performed. Supply voltage is varied from .9V to 1.4V with .1V increments and temperature is varied from 10°C to 40°C with 5°C increments. If voltage is being varied, temperature is being kept at the constant at the nominal 25°C and vice versa. The process parameters, V_{th}, T_{ox}, L_{eff}, are varied randomly by 3σ. Ten of the twenty Monte Carlo simulations are used as training data and the other ten simulations are used as testing data.

To reduce the model building time it is essential that the number of training vectors is reduced as much as possible. Each single Monte Carlo simulation contains 20,000 training vectors and after removing all the duplicate vectors we run SVM to create a set of support vectors that describe the training set. These training sets are small so it takes just a total of 6 minutes to create all the support vector sets. These sets are combined to form a single training set that describes the variation for a given temperature and voltage pair. This is repeated for each voltage and temperature pair. To create a single model, all the support vectors are extracted again to form a single training set which contains all simulated pairs of temperature and voltage and their corresponding sets of process variation.

Reducing the size of the model also greatly impacts the amount of time performing predictions takes. The prediction time is proportional to the number of support vector comparisons and number of elements therefore reducing the number of support vectors dramatically reduces the prediction time.

Table 3.5 shows waveform prediction results for the PLL using the single model described above. For each temperature and voltage pair the table shows the number of waveforms predicted each with a random set of process variation, the average prediction accuracy, and average lock time and power prediction accuracy. The non-bolded entries represent pairs that have been trained, but the process variation is different. The bolded entries represent new pairings that were not used to train the model. Most of the waveforms can be predicted with at least 95% accuracy, where most of the inaccuracies occur at the peak and valley amplitudes. Figure 3.9 shows a sample period of two of the predicted waveforms which highlights the mis-prediction in amplitude. The dotted lines are the predicted waveforms and the solid lines are the simulated waveforms. The most important aspect, being that of the frequency generated, is still predicted with high accuracy. The lock time, which is extracted from each predicted waveform, represents the time (ns) that the PLL stabilizes the frequency. From the results in Table 3.5 for lock time prediction accuracy represent how accuracy the simulation lock is in comparison to the predicted lock time. Even if the waveform prediction shows about 95% accuracy, that accuracy is lost in the amplitude and not the frequency lock time which is why the lock time prediction is so accurate.

The average time to predict each waveform is 28 seconds. Each waveform has 20,000 prediction points spanning the full 20µS capture time. This is compared to the 26 minutes it takes to do a single Monte Carlo simulation of the same 20µS. The prediction time would take less time if the model file contained fewer support vectors, i.e. if the design range was more refined. Since the design range is large the model file will in turn be large increasing the time it takes to perform predictions.

A SVM model is created to predict the power for a given waveforms based on voltage, temperature, and lock time. Figure 3.10 shows the power predictions for the test data.

Table 3.5: Predicted PLL waveform data

Voltage	Temperature	Number of Waveforms Predicted	Average Waveform Prediction Accuracy %	Lock Time Prediction Accuracy
0.9	25	10	94.8	99.89
1.0	25	10	95.2	99.96
1.1	25	10	95.3	99.97
1.2	25	10	95.5	99.82
1.3	25	10	96.3	99.97
1.4	25	10	96.5	99.97
1.2	10	10	94.95	99.94
1.2	15	10	94.87	99.75
1.2	20	10	94.96	99.76
1.2	25	10	95.12	99.86
1.2	30	10	95.31	99.92
1.2	35	10	95.33	99.95
1.2	40	10	95.53	99.91
0.95	**13**	**5**	**92.4**	**98.92**
0.95	**32**	**5**	**93.3**	**99.11**
1.05	**25**	**5**	**94.87**	**99.03**
1.15	**25**	**5**	**97.63**	**99.62**
1.25	**20**	**5**	**96.22**	**99.45**
1.2	**18**	**5**	**95.54**	**99.57**
1.33	**37**	**5**	**96.54**	**99.56**
1.5	**50**	**5**	**97.79**	**99.66**
1.15	**22.5**	**5**	**95.22**	**99.28**

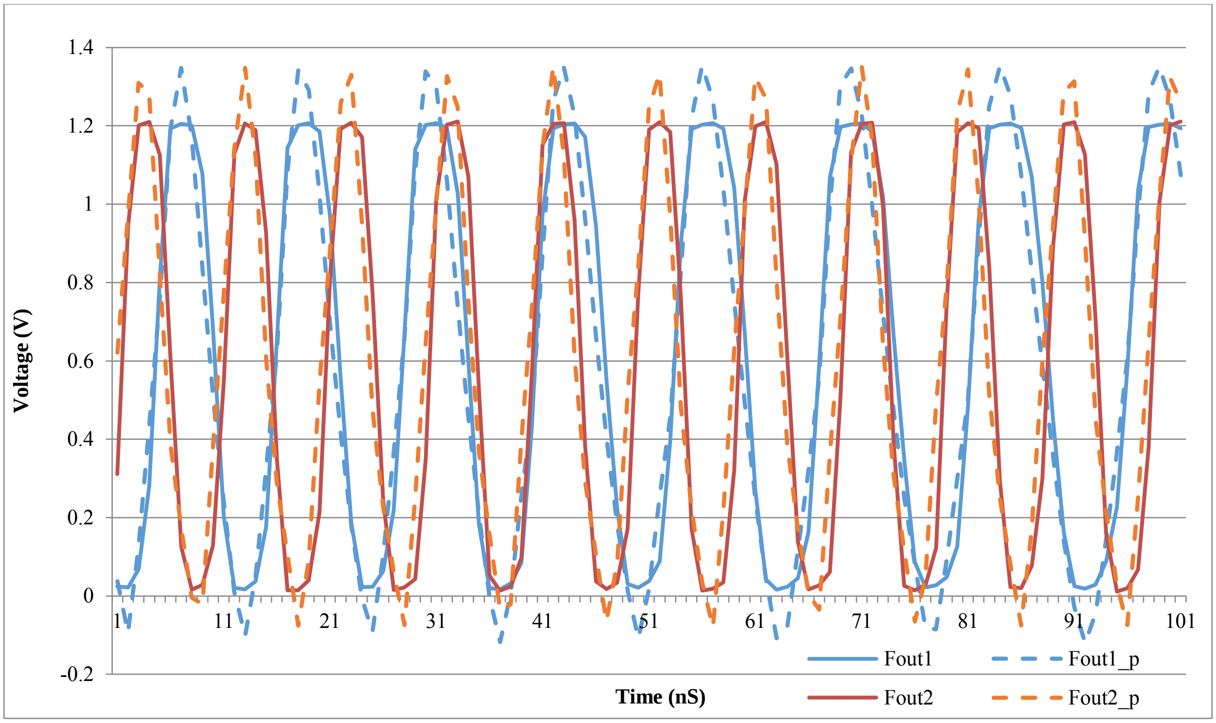


Figure 3.9: Two Monte Carlo simulations at 1.2 and 25°C (solid line) with the respective predicted waveforms (dashed line), shown is a snapshot of 100nS of simulation and prediction time for the waveforms.

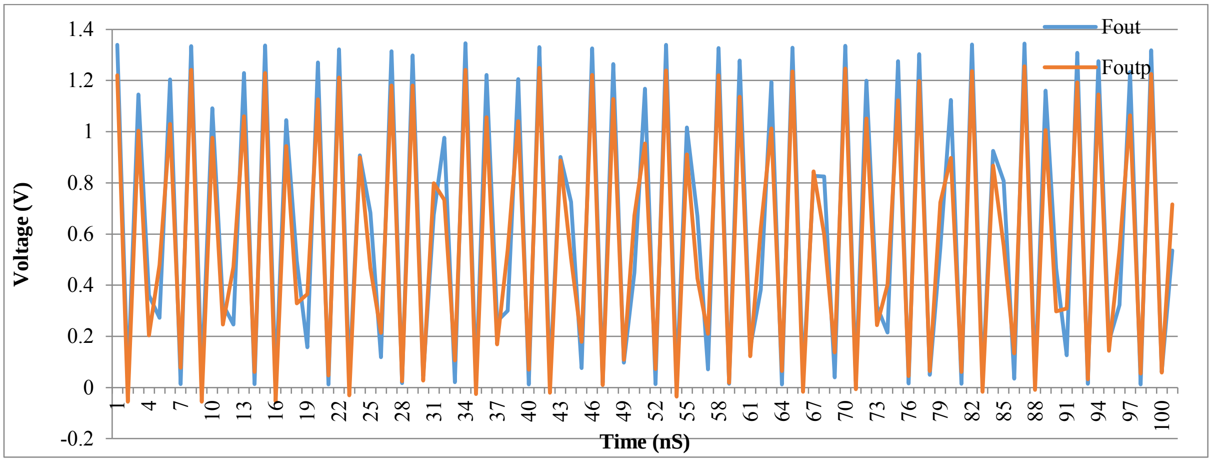


Figure 3.10: Predicted waveform for untrained pairs 1.33V and 37°C. Foutp is the predicted waveform and Fout is the simulated waveform. Snap shot of 100nS out of 20µS

3.6.1 Tradeoff Curves

Waveforms quickly generated from the SVM models can provide a designer with a fast way to analyze tradeoffs and determine the ideal operating conditions. For example, from the predicted PLL waveforms we can extract the lock time given the set of inputs, voltage, temperature, and process variation. The lock time in combination with the power SVM model described above can be used to create a power verse stability graph. In this section we use the UWB-PLL for proof of concept.

Figure 3.11 displays the tradeoff between power and lock time. The graph contains only simulated points where the red squares are fault free simulations while the X's represent simulations resulting in functional faults. The triangle represents the nominal simulation without any process variation. The line is the best fit boundary between the faulty and fault-free simulations. The X's by the 0 points in the axis represent simulations that fail to lock within the required 20μS time frame. The simulations with lock times close to 20μS lock too late in the simulation to be considered fault-free because there was not enough time to the circuit to stabilize at the lock time.

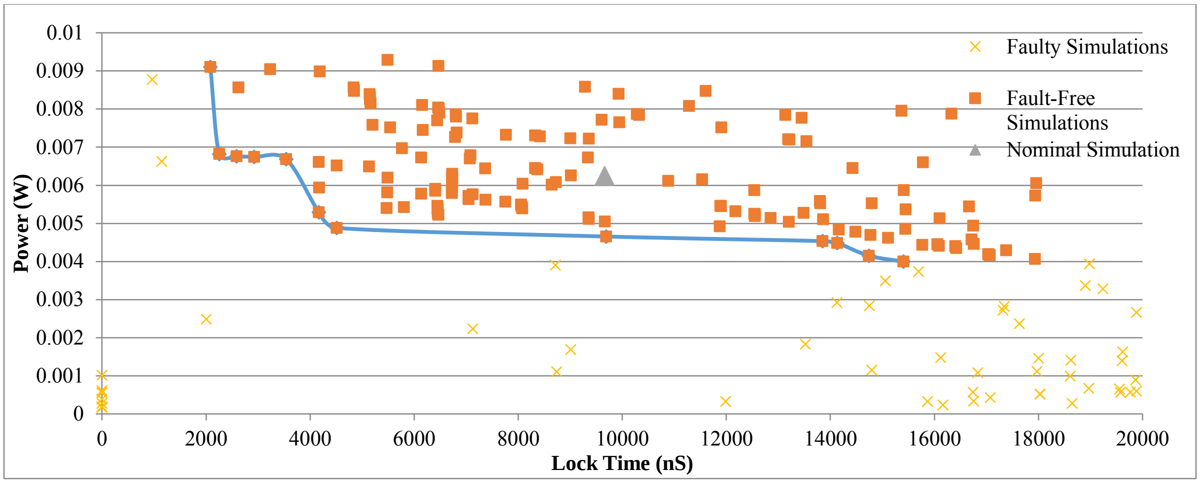


Figure 3.11: Graph of simulations comparing power and lock time. Boundary separates the faulty from the fault free simulations. The triangle is the nominal design point without variation

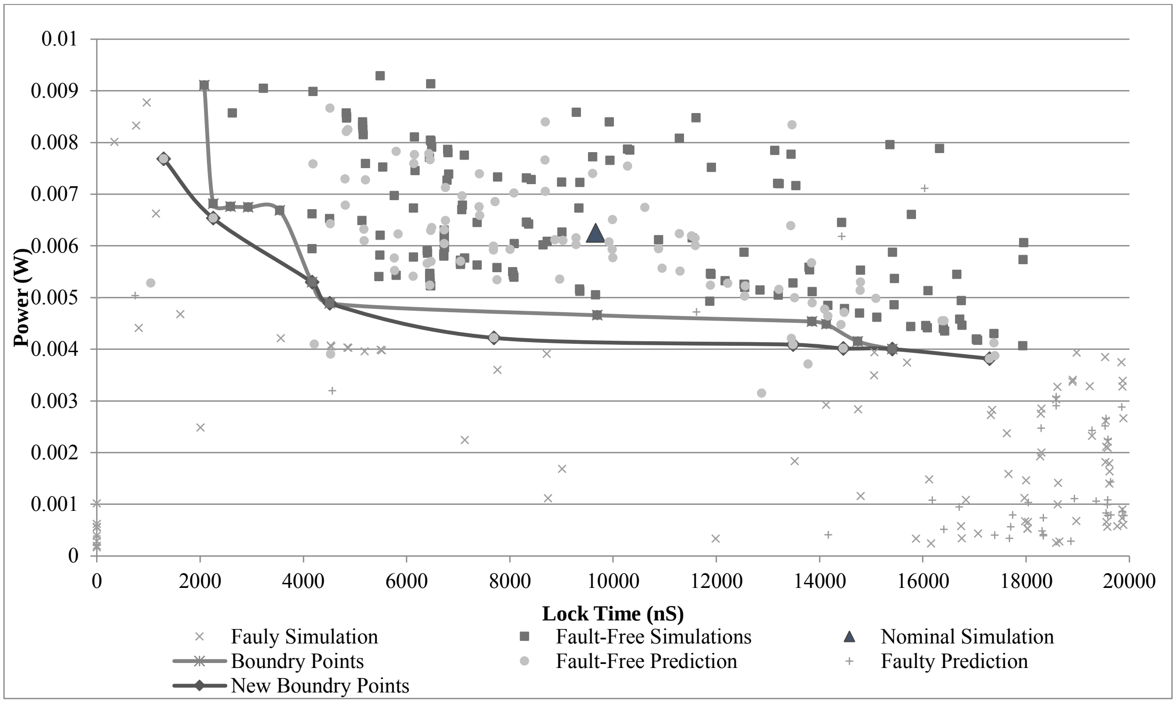


Figure 3.12: Graph of lock time verse power for simulated waveforms and predicted waveforms.

Creating more waveform data in order to fill out the graph in regions such as low power and fast lock time would be beneficial in determining where the design should be centered. More simulations can be performed in order to target key areas, but this can be time consuming. Using the variation models generated in the last section, we can create predicted waveform data quickly. Figure 3.12 shows 140 new waveform points generated by the SVM models. The new fault-free predictions points are represented as circuits and the faulty points are represented as pluses, +. As a result of the new data a new boundary line can be created separating the sets. There are some fault free predicted simulations in the faulty region and vice versa, these are outliers and ignored when creating the boundary. To create all 140 waveforms took 53.2 minutes, averaging 22 seconds per waveform, compared to 61 hours it would take to run HSpice.

3.7 Key Learnings

The method described in this chapter shows it is possible to create behavioral models in the time domain for analog circuits. For verification purposed it is difficult to ensure the soundness of the models, this is especially true for models with variation. A key component to learning methods is that the models are only as good as the data used to train them. With very large variation and input spaces it becomes difficult to ensure the model encapsulates enough behaviors in order to properly predict new behaviors. Another key issue is the complexity of some of the analog circuits. The switched capacitor network is a component which cannot easily be partitioned and predicted. With design/domain knowledge and user intervention the models can be hand tuned in order to capture the desired accuracy. This abolishes the idea of a general automated behavioral modeling method.

3.8 Conclusion

In this chapter we have presented a new approach for creating behavioral models of analog and mixed signal circuits based on partitioning. This methodology addresses the need for an automatic approach for behavioral modeling of any type of analog and mixed signal circuits. We developed a tool that can automatically create a set of partitions and detect intermediate behaviors based on netlist and transistor level simulation behavior. SVM models are created to predict intermediate behaviors which lead to the prediction of the final output behavior. We have shown the generality and feasibility of this approach on large circuits such as a PLL and ∑Δ-ADC. Our results indicate that we can obtain three orders of magnitude speedup over transistor level simulations while maintaining over 95% accuracy.

3.9 Chapter 3 References

[1] L. Pillage and R. Rohrer, "Asymptotic Waveform Evaluation for Timing Analysis," *IEEE Trans. Comp.-Aided Design Integr. Circuits Syst.*, vol. 9, no. 4, pp. 352–366, Apr. 1990.

[2] R. W. Freund, "Krylov-Subspace Methods for Reduced-Order Modeling in Circuit Simulation," *J. Comp. Appl. Math.*, vol. 123, no. 1/2, pp. 395–421, Nov. 2000.

[3] A. Odabasioglu, M. Celik, and L. Pileggi, "PRIMA: Passive Reduced Order Interconnect Macromodeling Algorithm," in *Proc. Int. Conf. Comp.-Aided Des.*, Nov. 1997, pp. 58–65.

[4] J. Roychowdhury, "Reduced-Order Modelling of Time-Varying Systems," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 46, no. 10, pp. 1273–1288, Nov. 1999.

[5] J. R. Phillips, “Automated Extraction of Nonlinear Circuit Macromodels,” in *Proc. IEEE Custom Integr. Circuits Conf.*, 2000, pp. 451–454.

[6] P. Li and L. T. Pileggi, “NORM: Compact Model Order Reduction of Weakly Nonlinear Systems,” in *Proc. IEEE Des. Autom. Conf.*, 2003, pp. 472–477.

[7] M. Rewienski and J. White, “A Trajectory Piecewise-Linear Approach to Model Order Reduction and Fast Simulation of Nonlinear Circuits and Micro-machined Devices,” in *Proc. Int. Conf. Comput.-Aided Des.*, Nov. 2001, pp. 252–257.

[8] S. K. Tiwary and R. A. Rutenbar, “Scalable Trajectory Methods for on Demand Analog Macromodel Extraction,” in *Proc. IEEE Des. Autom. Conf.*, 2005, pp. 403–408.

[9] H. Liu, A. Singhee, R. Rutenbar, and L. Carley, “Remembrance of Circuits Past: Macromodeling by Data Mining in Large Analog Design Spaces,” in *Proc. IEEE Des. Autom. Conf.*, 2002, pp. 437–442.

[10] X. Ren and T. J. Kazmierski, “Behavioral-level Performance Modeling of Analog and Mixed-signal Systems Using Support Vector Machines,” in *Proc. IEEE Int. Behavioral Model. Simul. Conf.*, 2006, pp. 28–33.

[11] M. Ding and R. Vemuri, “A Combined Feasibility and Performance Macromodel for Analog Circuits,” in *Proc. IEEE Des. Autom. Conf.*, 2005, pp. 63–68.

[12] T. Kiely and G. Gielen, “Performance Modeling of Analog Integrated Circuits Using Least-squares Support Vector Machines,” in *Proc. Des., Autom. Test Eur. Conf. Exhib.*, 2004, pp. 448–453.

[13] D. De Jonghe *et al*, "Extracting analytical nonlinear models from analog circuits by recursive vector fitting of Transfer Function Trajectories," *DATE,* 2013 , pp.1448-1453.

[14] Y. Cao *et.al*, "Dynamic Behavioral Modeling of Nonlinear Microwave Devices Using Real-Time Recurrent Neural Network," *IEEE Trans. El. Dev.,* vol.56, no.5, pp.1020-1026, 2009

[15] D. Drmanac *et. al.*, "A non-parametric approach to behavioral device modeling," *ISQED*, 2010, pp.284-290.

[16] M. Elzeftawi, "Compact Low-Power Low-Noise Neural Recording Wireless Channel for High Density Neural Implants (HDNIs)," PhD dissertation, Dept. Elect. Eng., Univ. California, Santa Barbara, 2012.

[17] B. Scholkopf, A.J. Smola, "*Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*," MIT Press, Cambridge, MA USA, 2001.

[18] H. Li, M. Mansour, S. Maturi "A new Sampling Method for Analog Behavioral Modeling," *Int. Symp. on Circuits and Systems (ISCAS),* pp.2908-2911, May 30 2010-June 2 2010

[18] D. Overhauser, I. Hajj, Y.F. Hsu, "Automatic Mixed-Mode timing simulation," *Proc. International Conference on Computer-Aided Design,* pp.84,87, 5-9 Nov 1989

[19] L. Yang, C.-J.R. Shi, "FROSTY: a Fast Hierarchy Extractor for Industrial CMOS Circuits," *Proc. Int. Conf. on Computer Aided Design, ICCAD-2003*, pp. 741- 746, 9-13 Nov. 2003

[20] X. Li, X. Zeng, D. Zhou, X. Ling, "Behavioral Modeling of Analog Circuits by Wavelet Collocation Method," *Proc. Int. Conf. on Computer Aided Design,* pp.65,69, 2001

[21] S. Lungu, D. Pitica, A. Rusu, "An Analogue Behavioral Macromodel Construction," *24th Int. Spring Seminar on Concurrent Engineering in Electronic Packaging,* pp.146-149, 2001

[22] S. Basu, B. Kommineni, R. Vemuri, "Variation-Aware Macromodeling and Synthesis of Analog Circuits Using Spline Center and Range Method and Dynamically Reduced Design Space," *Int. Conf. on VLSI Design,* pp.433,438, 5-9 Jan. 2009

[23] Y. Wei; A. Doboli, "Structural Macromodeling of Analog Circuits Through Model Decoupling and Transformation," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems,* vol.27, no.4, pp.712-725, April 2008

[27] HSPICE User Guide: Simulation and Analysis," www.synopsys.com

[28] C.Chang, C.J. Lin, "LIBSVM: A Library for Support Vector Machines,"ACM Transactions in Intelligent Systems and Technology, Vol. 2, Issue 3, 2011.

Chapter 4

Improving Circuit Verification Efficiency with Unsupervised and Supervised Learning

Circuit simulation is a common approach for verifying the analog behavior of a circuit. This chapter studies the application of statistical learning techniques for improving the circuit verification efficiency. To enable the application, circuit simulation is modeled as an event propagation process through a system consisting of primitive elements. Then, the efficiency improvements are achieved with two approaches. By assuming that the output space can be represented by a set of selected events, unsupervised learning is applied to search the input events that correspond to the selected output events. Only the selected input events are simulated, resulting in saving of the simulation time. During the simulation, low-complexity primitive elements with low information content are modeled by supervised learning models. Event propagation through these primitive elements is achieved by model prediction rather than by actual simulation, resulting in further saving of the simulation time. This chapter explains the statistical learning concepts and the techniques to implement the two approaches and demonstrates their effectiveness with experimental results in the context of voltage domain analysis of several analog circuit designs.

4.1 Introduction

Circuit simulation is indispensable for verifying the analog behavior of a design. For assessing the uncertainty of design behavior over process variations, Monte Carlo circuit simulation is one of the most popular approaches. However, circuit simulation can be time consuming. Hence for a large and complex design, Monte Carlo circuit simulation can become prohibitively expensive.

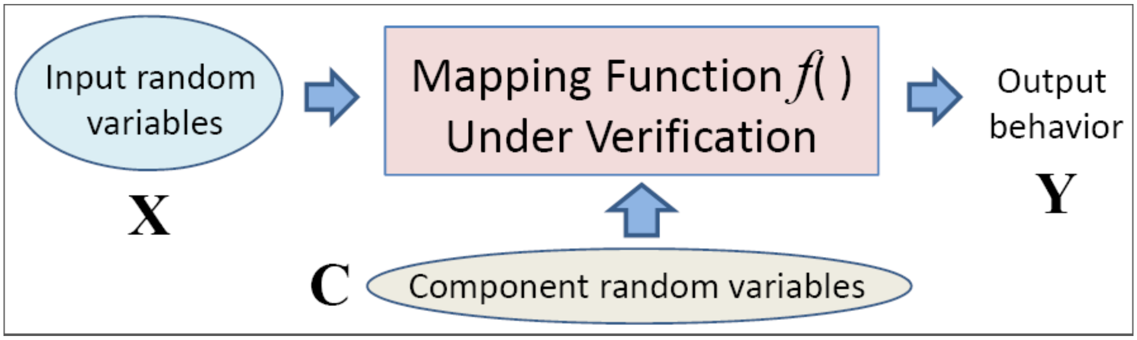


Figure 4.1: Functional view of uncertainty analysis of a statistical system

Figure 4.1 depicts a functional view for the underlying problem in uncertainty analysis of a statistical system. In this view, the circuit to verify is seen as a mapping function *f()*. Inputs to the function comprise two sets of random variables. First, there are variations on the input space **X**. Furthermore, there are variations on the components constituting the circuit. The component space is denoted as **C**. The function *f()* is a mapping from (**X**, **C**) to the output space

$\mathbf{Y}$, i.e. f: $(\mathbf{X}, \mathbf{C}) \rightarrow \mathbf{Y}$. In verification, there is a specification on the expected output behavior. Hence, the problem becomes, giving $(\mathbf{X}, \mathbf{C})$, is there any output behavior out of the specification?

In a typical setting of Monte Carlo circuit simulation, a particular input x from the input space $\mathbf{X}$ is selected. Uncertainty analysis concerns the output behavior with respect to the component variations, i.e. assessing the output subspace $\mathbf{Y}_x=f(x, \mathbf{C})$. The idea of improving the efficiency of uncertainty analysis in this setting is not new.

For example, one notable area of research is static statistical timing analysis (SSTA) [1][2]. For example, in SSTA each delay element is modeled as a random variable according to process variations. Circuit timing is a function of a set of delay random variables under the worst-case assumption on the input pattern space. The analysis is static because variation on the input pattern to the circuit is not considered. In SSTA, efficiency is obtained by propagating the random variables directly through the circuit. It does not involve random sampling and hence avoids the high cost of Monte Carlo simulation. Delay elements in SSTA are usually assumed to be pin-to-pin delays of a cell [2].

The same idea of propagating random variables can be applied to lower-level circuit analysis where the random variables are based on basic circuit elements such as resistors and capacitors. In lower-level circuit analysis, the operators involved are no longer restricted to addition and maximization as used in SSTA. Hence, the problem becomes more complex. For example, the work in [3] applied Polynomial Chaos Theory (PCT) [4] to low-level circuit analysis. In a PCT framework, random variables are modeled by orthogonal polynomials to facilitate their propagation through circuit equations [3].

In contrast, the work in [5] retains the idea of random sampling. To improve efficiency, supervised learning techniques are applied to identify the irrelevant subspace (in **C**) to ignore. Because the inputs to the circuit under analysis are well defined and limited, they are not required to be treated as coming from a random space. In more recent works [6][7], advanced learning techniques are applied to develop an efficient statistical analysis framework for uncertainty analysis of circuit performance parameters. Similarly, the underlying sources of uncertainty for the analysis are from the components **C** constituting the mapping function.

Because traditional Monte Carlo analysis is cost restricted to analyze small circuits, it is usually not required to explicitly treat the input space as a random variation space. Hence, from the objective of improving the efficiency of Monte Carlo analysis, the focus is usually on modeling the behavior with respect to the component variations **C**. However, this view can change when the circuit to verify becomes large.

An example to consider variation in the circuit input space X is in the context of delay testing. In delay testing, the mapping function *f()* is a gate-level circuit of *n* inputs. Hence, there can be up to 2^n input patterns to apply. For this problem, component variations in Figure 4.1 are based on two sources: (1) variations on the delay elements due to process variations, similar to the setting of SSTA, and additionally (2) variations in the delay defect sizes and locations. The output behavior is divided into two classes: passing and failing.

Since it is not feasible to apply all 2^n input patterns, one crucial aspect of the uncertainty analysis is to identify the important input patterns such that the result of uncertainty analysis across all input patterns can be approximated by the result of uncertainty analysis across the important input patterns. Hence, the saving is obtained by avoiding the analysis on the unimportant input patterns.

The work in [8] is an example of how to approach this delay testing problem. However, the work relied on Monte Carlo simulation of the entire circuit to be analyzed and hence, no saving could be obtained with respect to the component random space **C**.

In Figure 4.1 if one takes an extreme view that the mapping function is a processor or SoC, then the input space **X** becomes extremely large. In this case, considering component variations is no longer practical. Typically, for verifying a SoC, RTL simulation is used. Simulation cost is due to the large input space **X** to cover. In this context, an input pattern is a functional test, i.e. a sequence of input vectors.

Suppose functional tests can be represented by *n* random variables each varying across a domain of values. Then, the input space **X** can be viewed as a *n*-dimensional random space. With such a view, the work in [9] proposed a framework for saving simulation time by identifying and simulating only the important functional tests. Unsupervised learning technique, the Support Vector Machine one-class [10], was applied to learn and model the unimportant input subspace to facilitate the selection of the important tests. The work in [11][13] extended the idea to the analysis where each functional test was an assembly program.

In view of the prior works discussed above, observe that the idea of improving simulation efficiency with respect to either the **X** space or the **C** space is not new. However, what has not been explored is to improve simulation efficiency in the context of Figure 4.1 where both the input variations and component variations are considered in the uncertainty analysis. This motivates the study described in this work to develop a framework that tackles the new problem setting.

With respect to Figure 4.1, such a framework aims to obtain saving from two ends: (1) Simulation cost is reduced by avoiding the simulation of unimportant inputs sampled from **X**. This objective is similar to the works in [8][9][11][13].

(2) Simulation cost is reduced by predicting a substantial number of events arising in the Monte Carlo simulation. In other words, instead of using the actual simulation to obtain those events, those events are predicted by learning models and hence, the total simulation time is reduced. This objective can be thought of as similar to the works [5][6][7] discussed above.

It should be noted that the objective of this work is not to develop a framework that is more general than the prior works. The proposed framework is to be applied in a different application context in which the variations in both the **X** space and the **C** space in Figure 4.1 need to be considered in the uncertainty analysis. Hence, the proposed framework should not be viewed as an alternative to the prior works in their respective application contexts. It can be viewed as an addition which addresses a different application context.

The rest of the chapter is organized as the following. Section 4.2 explains the two essential problems to solve in this work, one for each objective just discussed. Section 4.3 discusses basic machine learning concepts and keys to enable the application of statistical learning techniques in the respective two problem settings. Section 4.4 presents our approach to tackle the first problem where unsupervised learning is applied on the input space **X** to achieve the objective (1) mentioned above. Then, section 4.5 presents our approach to solve the second problem where supervised learning techniques are used to achieve the objective (2). Section 4.6 puts all pieces together into a unified framework. Section 4.7 demonstrates the effectiveness of this framework with experimental results. The experiments were conducted based on a number of analog circuits with the focus on voltage domain analysis. Section 4.8 concludes.

4.2 Two Essential Problems to Solve

4.2.1 The Underlying Question to Ask

The problem stated with Figure 4.1 in theory can be thought of as a pre-image computation problem illustrated in Figure 4.2. Let the combined input space to the function *f()* be denoted as **X**·**C**. Let the resulting output space be denoted as **Y**. Typically, some portion of the **Y** space is considered acceptable while other portion is considered not acceptable. Let **Y** denote the acceptable subspace. The theoretical problem of the verification can be thought of as computing the pre-image $f^{-1}(\mathbf{y})$.

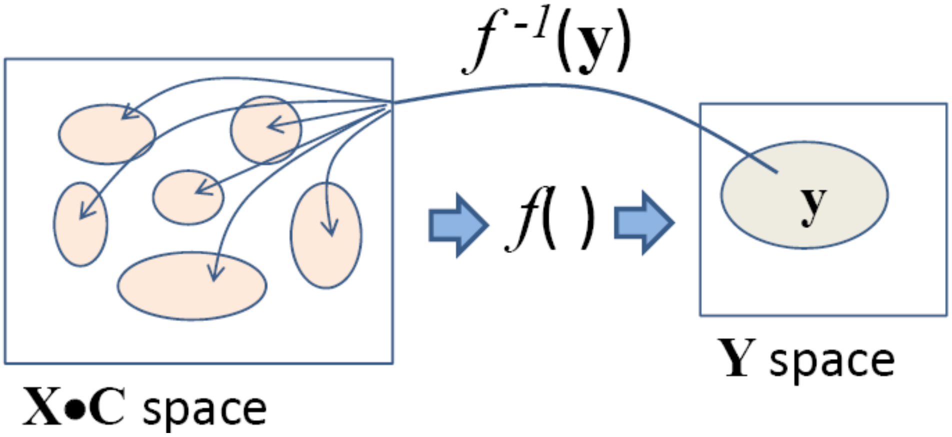
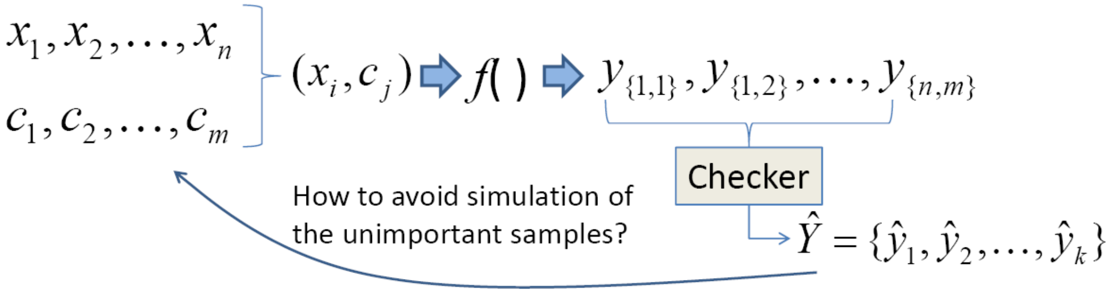


Figure 4.2: Theoretical view of the fundamental problem

Figure 4.2, while theoretically simple to understand, is a practically difficult problem to solve. For example, while *f()* is computable with a simulator, $f^{-1}()$ is usually not. Moreover, input variation space **X** and the component variation space **C** are fundamentally two different spaces. For example, a sample in **X** can be a waveform over a simulation period [0,*T*]. A sample in **C** can be a sample vector ($s_1,...,s_p$) where each s_i is the size sampled for the transistor *i* in the circuit, according to a process variation model. Therefore, even assuming that one can formally model the acceptable space **Y** with a set of rules or equations, it remains difficult to compute the corresponding rules or equations in the combined **X·C** space.

More practically, the verification problem illustrated in Figure 4.2 is approached by asking a different question. This question is illustrated in Figure 4.3.



$x_1, x_2, ..., x_n$
$c_1, c_2, ..., c_m$
(x_i, c_j)
$f(\ )$
$y_{\{1,1\}}, y_{\{1,2\}}, ..., y_{\{n,m\}}$
Checker
$\hat{Y} = \{\hat{y}_1, \hat{y}_2, ..., \hat{y}_k\}$
How to avoid simulation of the unimportant samples?

Figure 4.3: Practical view of the problem

In the setting of Figure 4.3, *n* samples $x_1,...,x_n$ are drawn from the **X** space. For example, these can correspond to *n* different input waveforms over a simulation period [0,*T*]. Moreover, *m* samples $c_1,...,c_m$ are drawn from the **C** space. For example, each c_i is a vector capturing the sizes of *p* transistors and together, they represent *m* samples drawn from a process variation model for statistical Monte Carlo simulation. Every combination (x_i, c_i) is simulated to produce the output waveform $y_{\{i,j\}}$ over the period [0,*T*]. In verification, a checker is applied to these *n* X *m* outputs to decide which are acceptable and which are unacceptable.

One key concern in Figure 4.3 is obviously the total simulation time over the *n* X *m* combined input samples. Suppose every output $y_{\{i,j\}}$ is crucial for the checker to verify the design correctness. In this case, there is not much one can do but simulating all the n X m input samples.

Suppose, however, that in order for the checker to be used to prove correctness of a design, it does not need all the n X m output samples. Instead, a rather smaller set $\hat{\boldsymbol{y}}$ of k *representative* samples $\{\hat{y}_1, \hat{y}_2,...,\hat{y}_k\}$ is sufficient, i.e. $k << n$ X m. Then, given these k output samples, ideally one only needs to simulate the corresponding k samples in the combined input space. In this context, the underlying question becomes: How to avoid simulating the unimportant inputs that do not change the representative output set $\hat{\boldsymbol{y}}$?

Suppose i represents the information contained in a representative output set $\hat{\boldsymbol{y}}$, which is used to prove correctness. Alternatively, one can ask a different question: How to select a minimal number of combined input samples to simulate and obtain an output set that contains an equal amount of the information as i and is sufficient to prove correctness?

Conceptually, one can treat $\{x_i, c_j\}$ as a single unified sample and try to solve the problem illustrated in Figure 4.3. However, this can be difficult because as mentioned above, x_i and c_i come from two fundamentally different domains. A more effective way is to consider these two domains separately. Below we will discuss how to approach the problem by solving two essential problems.

4.2.2 First Essential Problem

In the first essential problem, we consider a fixed c_j. Figure 4.4 illustrates the problem. Given a particular c_j, assuming that simulation of all input samples $x_1,...,x_n$ results in coverage of a subset of l output samples, without loss of generality, denoted as $\hat{L}=\{\hat{y}_1, \hat{y}_2,..., \hat{y}_l\}$ for $l \leq k$.

In Figure 4.4, these l output samples are colored red. There are other output samples colored green. They are output samples not reachable by simulating with the component sample c_j.

The l output samples correspond to partitioning the n input samples into l clusters. Therefore, ideally if at least one representative sample is selected from each cluster and simulated, all the l output samples would be covered.

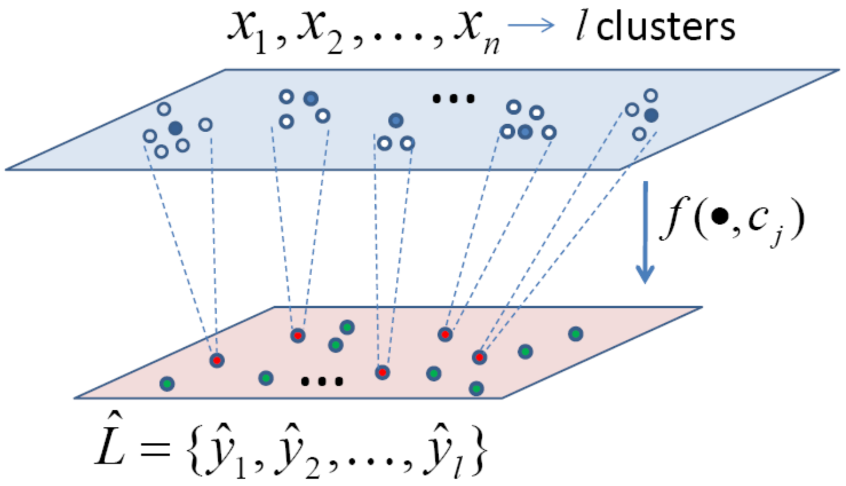

$x_1, x_2, \ldots, x_n \rightarrow$ l clusters
$f(\bullet, c_j)$
$\hat{L} = \{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_l\}$

Figure 4.4: Discovery of representative input samples

Let the set of l clusters be denoted as $U=\{u_1,\ldots,u_l\}$. Let the set of selected inputs to simulate be $\hat{X}=\{\hat{x}1,\ldots,\hat{x}q\}$. We call them the *representative input samples*. We have $q \geq l$. We say that U is covered by $\hat{X}$ if for every cluster u_i, there exists an input $x_j \in \hat{X}$ such that $x_j \in u_i$. With these definitions, the first essential problem can be stated as: How to discover a minimal-size set $\hat{X}$ that *covers* U without knowing the output sample set $\hat{L}$ in advance?

In the context of functional verification, the output set $\hat{L}$ can be thought of as a set of coverage points while each input x_i as an assembly program [13]. In this context, the work in [11] approaches the problem by assuming that the set of coverage points is known in advance. The learning approach takes advantage of this information to select representative inputs to simulate. In contrast, the work in [12] approaches the same problem without assuming that the set of coverage points is known in advance. Consequently, the approach in [11] is more effective for a specific application task than the approach in [12] (e.g. see discussion in [13]).

Both approaches follow an unsupervised learning paradigm. Although they are designed for a completely different context, conceptually they provide two hints to approach the first essential problem stated above: (1) The problem could be approached by unsupervised learning; and (2) Without knowing $\hat{L}$ in advance is a harder problem than that with $\hat{L}$ given.

4.2.3 Considering Component Variations

Suppose we have a method that can find a desired set of representative input samples. An intuitive question arises: Can the method be used to find a desired set of representative samples from the component variation space **C** as well? Figure 4.5 illustrates the difficulty following this thought.

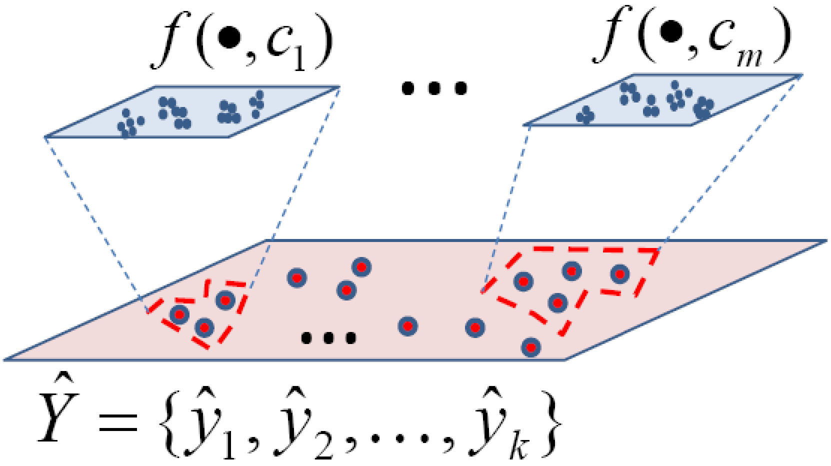


Figure 4.5: An intuitive way to consider component variations

Consider the outcome by simulating all inputs $x_1,...,x_n$ based on a component sample c_j.

The outcome is a subset of $\widehat{Y}$. Let $f(\cdot, c_j)$ denote this subset. To save simulation, one does not want to simulate all component samples $c_1,...,c_m$. Rather, a subset of component samples is selected for simulation.

From this perspective, we see that the basic problem to solve is, to discover a minimal set from the m subsets $f(\cdot, c_1),...,f(\cdot, c_m)$ such that a complete coverage of the output samples in $\hat{Y}$ can be obtained. However, even with the m subsets and $\hat{Y}$ known in advance, finding a minimum set of subsets for the complete coverage is the same as the Set Cover Problem, a well-known NP-complete problem. Hence, the problem illustrated in Figure 4.5 can be a very difficult problem because neither the m subsets nor the $\hat{Y}$ are known in advance.

As pointed out in [14], statistical learning techniques do not make an NP-hard problem easier. Consider the well-known Boolean Satisfiability (SAT) problem. From the statistical learning perspective, SAT can be reduced to the problem of learning a Boolean function. As discussed in [14], although some Boolean functions can be learned with a good accuracy [15], learning a Boolean function in general can be a computationally difficult problem, i.e. the learning problem itself is NP-hard [16]. Therefore, the problem formulated with Figure 4.5 is not a problem suitable for application of statistical learning techniques. We need to pursue an alternative.

4.2.4 Second Essential Problem

The discussion with Figure 4.5 above assumes that the only decision that can be made to save simulation time is whether to skip the simulation based on each component sample c_i

or not. However, this is not the only way to save the total Monte Carlo simulation time. An alternative to save simulation time can be based on a different set of decisions: whether to skip the simulation on some parts of the circuit. Figure 4.6 illustrates this alternative.

Suppose the circuit under verification can be partitioned into a set of *primitive elements* (PEs). Some PEs are simple. For example, an inverter is a simple PE. Some PEs are complex. Then, the basic idea is that simple PEs are replaced with statistical learning models and complex PEs are simulated. Hence, the saving comes from skipping the simulation on the simple PEs.

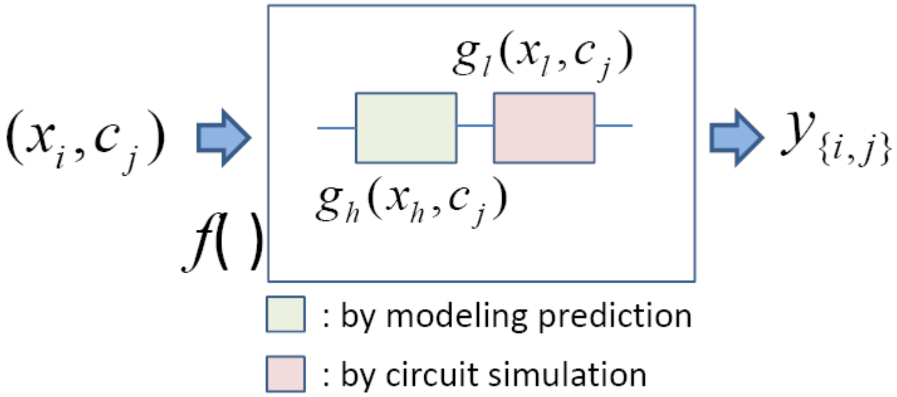


Figure 4.6: A more practical way to consider component variations

Note that modeling a simple PE is not solving the traditional analog behavior modeling problem. First, in traditional behavior modeling, the component variations are usually not considered. In recent works [17][18] some experiments were conducted to pursue behavior modeling with component variations. However, the attempts were to derive a model for the entire circuit. The works in [17][18] reached limited success because some aspects of circuit behavior could be too complex to capture with a learning model. For simple circuits, however, a learning model could work well [18].

Therefore, assuming that a learning method is given to model the behavior of a PE over the combined space **X·C**, the essential problem illustrated with Figure 4.6 is to decide which PEs are simple enough that a learning model can be used to predict their behavior. In other words, the focus of the problem is not on how to learn a model for a PE. The focus is on how to define what a simple PE is with respect to a learning method. In other words, the focus is on deciding which PEs are *learnable* and which PEs are not. Solving this problem requires (1) a method to partition a circuit into a set of PEs, and (2) a method to evaluate the complexity of learning the behavior of a PE.

4.3 Keys to Learning Problem Formulations

4.3.1 Basic Machine Learning Concepts

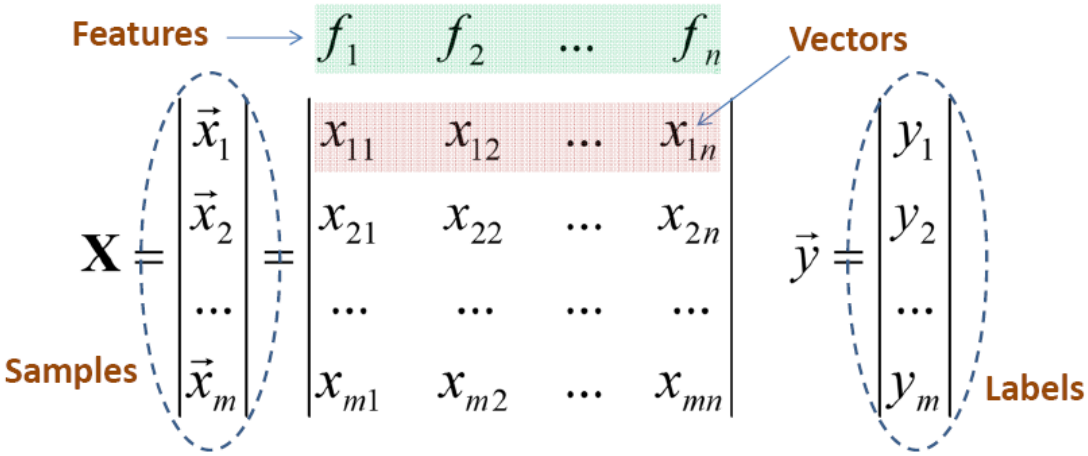

Features
f_1 f_2 … f_n
Vectors
$\mathbf{X} = \begin{vmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \cdots \\ \vec{x}_m \end{vmatrix} = \begin{vmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \end{vmatrix}$
$\vec{y} = \begin{vmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{vmatrix}$
Samples
Labels

Figure 4.7: Typical dataset seen by a learning algorithm [14]

Figure 4.7 illustrates a typical dataset seen by a machine learning algorithm (For more discussion, see e.g. [14]). When $\vec{y}$ is present and there is a label for every sample, it is called *supervised* learning. In supervised learning, if each y_i is a categorized value, it is a *classification* problem. If each y_i is modeled as a continuous value, it is a *regression* problem.

When $\vec{y}$ is not present and only **X** is present, it is called *unsupervised* learning. When some (usually much fewer) samples are with labels and others have no label, the learning is then called *semi-supervised* [19].

In Figure 4.7 each sample from **X** is encoded with n features $f1,...,f_n$. Hence, the characteristics of each sample are described as a vector $\vec{x_i}$. To apply a learning algorithm to analyze a set of samples, an intuitive way is to decide on a set of features to encode the samples.

4.3.2 The Importance of Similarity Measure

Many modern machine learning algorithms follow the paradigm of *kernel-based learning* [20][21]. Figure 4.8 illustrates the basic concept of kernel-based learning.

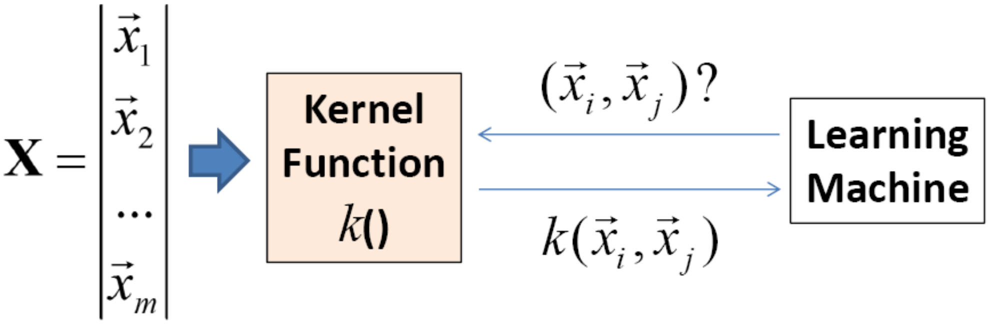

$\mathbf{X} = \begin{vmatrix} \vec{x}_1 \\ \vec{x}_2 \\ \dots \\ \vec{x}_m \end{vmatrix}$
Kernel Function $k()$
$(\vec{x}_i, \vec{x}_j)$?
$k(\vec{x}_i, \vec{x}_j)$
Learning Machine

Figure 4.8: Kernel function vs. learning machine

In kernel-based learning, the learning machine, i.e. the learning algorithm such as a Support Vector Machine (SVM) algorithm [20], is not required to access the samples $\overrightarrow{x_1}$,...,$\overrightarrow{x_m}$ directly. Instead all the information required for the learning is coming from a *kernel function k()*. The kernel function measures the *similarity* between two samples based on some definition of what similarity means.

Kernel-based learning provides great flexibility to apply learning techniques in EDA and test applications [14], especially when the samples to be analyzed are not provided in matrix form like Figure 4.8. This is because the representation of a sample is not important. What is important is to define a proper similarity measure function.

4.3.3 Keys to Enable the Application of Learning Techniques

While the concepts discussed up to this point are more general, for experimental purpose this work focuses on a rather specific application context. In this context, each input sample x_i from the **X** space is an input waveform to a circuit over a period [0,T]. Each sample c_j from the **C** space is a vector of transistor sizes sampled from a process variation model.

Consider the first essential problem. In this problem, the input samples are waveforms. Hence, to enable learning, one has to devise a method to measure the similarity between two waveforms. This is the key to apply unsupervised learning techniques to approach the first essential problem.

Consider the second essential problem. This problem can be thought of as deciding the complexity of learning the function *f()* in Figure 4.1 above, where *f()* is based on a PE (primitive element). From this perspective, we see that the input samples to the PE are still coming from the combined space **X**·**C**. Hence, each sample can be represented as a 2-tuple (x'_a,

c'_b). Here, we use the notation (x'_a, c'_b) to emphasize the difference from the notation (x'_i, $c'j$) used earlier. Each x'_a is a waveform to a PE. Each c'_b is a vector of transistor sizes for the transistors in the PE. In contrast, each c_j earlier is a vector of sizes for all transistors in the circuit. In other words, the dimensionality of c'_b is much smaller than the dimensionality of c_j.

While defining a proper kernel function for samples (x_i, c_j) can be hard, it is more feasible to define a kernel function for samples (x'_a, c'_b) because of their reduced dimensionality. It is important to note that in a 2-tuple (x'_a, c'_b), x'_a is a waveform and the 2-tuple is an object across two different domains. Therefore, to enable learning, one has to devise a method to measure the similarity between pairs of 2-tuple samples. This is one of the keys to apply learning techniques for the second essential problem.

For predicting the behavior of a PE, supervised learning techniques would be applied (explained later). In Figure 4.8, an output $\vec{y_i}$ is a scalar value. In contrast, the output of a PE is a waveform. This raises another important question: How to apply supervised learning when the outputs to be predicted are not scalar values? Answering this question is another key for the second essential problem.

4.4 Unsupervised Learning for Important Input Subspace Modeling

Suppose n samples x_1,...,x_n are given. Recall that the goal is to cover k representative output samples $\hat{Y}$={$\hat{y}_1$,...,$\hat{y}_k$}. Without loss of generality, the assumption is that these k output samples are dissimilar to each other. In other words, suppose k_y() is a similarity measure function applicable to the output samples. We have $0 \leq k_y() \leq 1$, and $k_y()=1$ means the two samples are most similar and $k_y()=0$ means the two samples are least similar (or most dissimilar). Then, for any pair $\hat{y}_{\mathrm{i}}, \hat{y}_{\mathrm{j}}$, we have $k(\hat{y}_{\mathrm{i}}, \hat{y}_{\mathrm{j}}) \leq \delta$ for some δ.

Suppose the simulation process is broken into iterations. In each iteration, *l* samples are applied. Figure 4.9 depicts what might happen from one iteration to the next. In this example, there are 14 samples, i.e. n=14. In each iteration, 3 samples are applied, i.e. l=3.

For iteration i=0, no sample is applied yet. Hence, the coverage on the **Y** space is unknown. For iteration i=1, 3 samples are selected and applied. The picture depicts that these 3 samples are selected from three clusters of samples. In other words, a clustering algorithm is applied to group samples. Then, one sample is selected from each cluster. Note that in order to apply a clustering algorithm, another similarity measure function $k_x()$ is needed to measure the similarity between two input samples. Then, similar samples can be grouped together where the similarity is defined based on $k_x()$.

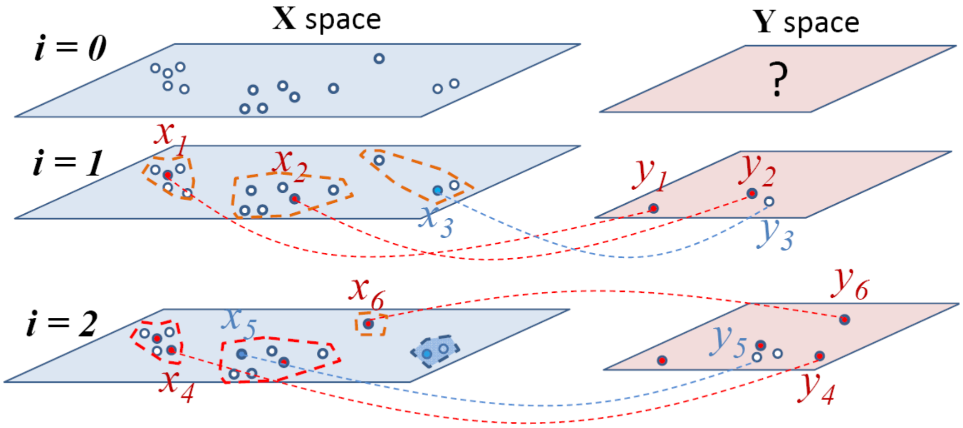


Figure 4.9: Iterative search for important inputs in **X**

Application of three input samples x_1, x_2, x_3 results in three output samples y_1, y_2, y_3. Suppose based on a predefined method (e.g. $k(\hat{y}_{\mathrm{i}}, \hat{y}_{\mathrm{j}}) \leq \delta$), y_1 and y_2 are determined to be representative output samples, i.e. $y_1, y_2 \in \widehat{\boldsymbol{Y}}$, but y_3 is not. In the **X** space, x_1 and x_2 are marked as important samples (solid red) and x_3 as unimportant samples (solid blue).

The question is: for the next iteration, what would be a good strategy to select input samples?

Suppose the input samples are distributed in such a way that if two samples are dissimilar in the **X** space, they both are likely to be important samples. In other words, $k_x() \approx k_y()$. In this case, finding important input samples can be reduced to finding representative input samples by clustering. This is because conceptually two representative samples from two clusters are most dissimilar samples. Hence, in this case a simple clustering algorithm could solve most of the problem. In other words, the strategy would be to form k clusters in the input space and then, in each iteration, select k representative input samples, one from each cluster. Then, the process repeats by removing those samples that have been applied.

A more realistic assumption is that $k_x()$ and $k_y()$ are very different, resulting in some input subspaces more important than others. In this case, a strategy is needed to direct the search to (1) identify the important subspaces and (2) select more samples from the important subspaces.

Following this idea, for iteration i=2, the sample close to x_3 is blacked out. Clustering is then applied to the rest of the samples. Three samples x_4, x_5, x_6 are selected. Samples x_4 and x_5 are selected from the same clusters as x_1 and x_2 before. This is because x_1 and x_2 are identified as important samples. Hence, regions close to them are treated as important subspaces in this iteration. Notice that x_6 is at a distance from x_3 and hence, it is not deemed unimportant because of x_3. In iteration i=2, x_6 by itself forms a cluster and is selected.

The above example illustrates the search strategy. A known unimportant sample results in an unimportant subspace. A known important sample results in an important subspace. There can be other samples not in either of the subspaces. Clustering is applied in each iteration to find representative samples in the "adjusted" **X** space.

4.4.1 Adaptive Similarity Measure

Figure 4.10 illustrates how the idea discussed above can be accomplished without changing the clustering algorithm and the similarity measure function $k_x()$ from one iteration to the next. Suppose originally each input sample is encoded with two features f_1 and f_2. Hence, the samples are distributed in a 2-dimensional space defined by f_1 and f_2. This is shown as the left plot in the figure.

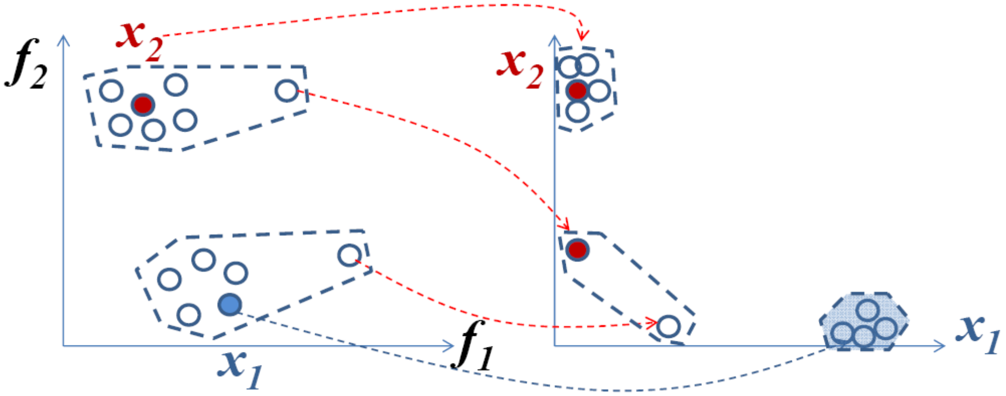


Figure 4.10: Adaptive learning space for similarity measure

Clustering is applied to form two clusters and two representative samples x_1 and x_2 are selected. The trick is that in the next iteration, a new space is created based on x_1 and x_2.

The coordinate of a sample x is calculated as ($k_x(x, x_1)$,k_x (x, x_2)). Therefore, the samples close to x_1 (and far from x_2) will all be close to the point *(1,0)* and form a cluster. This cluster is blacked out because x_1 is not an important sample. The samples close to x_2 (and far from x_1) will be close to the point *(0,1)* and form another cluster. One representative sample is selected because x_2 is an important sample. Furthermore, the remaining two samples form a cluster and another representative sample is selected.

The simple example illustrates that without changing the similarity measure function $k_x()$ and without changing the clustering algorithm, by projecting the samples into a different space, the idea discussed above can be realized.

4.4.2 Issue of Large Dimensionality

The idea discussed above has one major drawback. As the number of the applied input samples grows, the dimensionality of the adjusted space grows accordingly. To avoid this growth, the actual implementation can be the following. Without loss of generality, assume that l samples $x_1,...,x_l$ have been applied. Samples $x_1,..., x_i$ are deemed important samples while $x_{i+1},..., x_l$ are deemed unimportant.

1) Start with all input samples not yet being applied.

2) For an input sample x, for each j, $i+1 \leq j \leq l$, if $k_x(x, x_j)$ is greater than $\max\{k_x(x,x_h) \mid \forall h, 1 \leq h \leq i\}$, then x is removed from consideration in the current iteration. In other words, x is more similar (or "closer") to an unimportant sample than to *any* of the important samples.

3) For all the input samples not removed, project them into the space defined by $x_1,...,x_i$. Run clustering in the new space to find representative samples.

In the above implementation, the dimensionality of the adjusted space is bounded by *k*, the number of representative output samples. A sample is removed from consideration in a particular iteration if the sample falls into an unimportant subspace. An unimportant subspace contains those samples closer to unimportant samples than to any of the important samples.

4.4.3 Property on the Removed Samples

It is important to note that a removed sample in one iteration can still have a chance to be selected in a future iteration. Figure 4.11 illustrates this point. Figure 4.11 depicts a selection process of three iterations (from left to right) for two cases. In the iteration 1 of case 1, x_1 is deemed important while x_4 is deemed unimportant. Since x_3 is closer to x_4 than to x_1, i.e. $k(x_3, x_4)>k(x_3, x_1)$, x_3 is removed in the next iteration. In contrast, x_2 is kept. In the next iteration, x_2 is applied and deemed important. In iteration 3, since x_3 is closer to x_2 than to x_4, i.e. $k(x_3, x_4)<k(x_3, x_2)$, x_3 is kept and selected.

The case 2 depicts a different scenario where $k(x_3, x_4)>k(x_3, x_2)$. In this case, the process stops at iteration 3 where x_3 is not selected because it is closer to x_4 than to x_2.

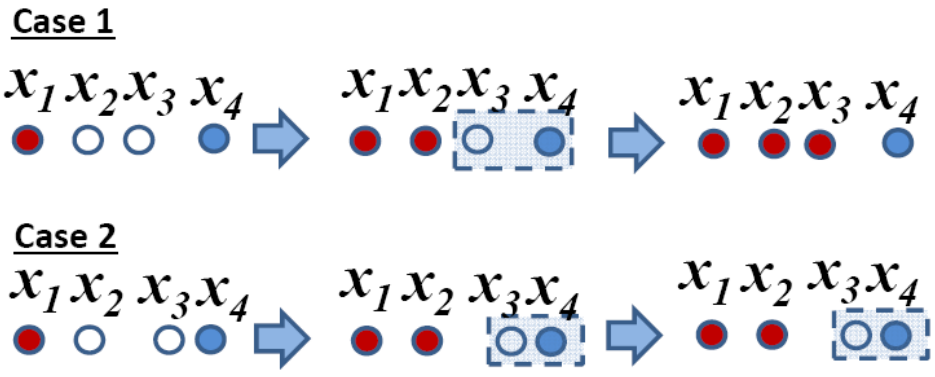


Figure 4.11: A removed sample in one iteration can be selected in the next

4.4.4 Implementation Considerations

Suppose that given a set of output samples, the user has a method to determine the representative samples. Then, there are two remaining questions to consider for implementing the search strategy for finding important input samples.

The first question is how to define the similarity measure function $k_x()$. Because for the application discussed in this work, each input sample is a waveform over a period $[0,T]$, a sample x can be represented as a vector $[x_0,...,x_T]$ assuming that time is incremented by 1 unit. Then, given two input sample vectors, the popular Gaussian kernel function [21] can be used as the similarity measure function.

In general, the input waveforms can be digital or analog, periodic or non-periodic, stationary or non-stationary. The user might have an idea what important characteristics are to be analyzed by the system under simulation. For example, for analyzing periodic non-stationary waveforms, the user might desire to encode waveforms with wavelet transform [22]. Hence, the user can apply a particular transform to the waveform before applying the Gaussian kernel. How to encode/transform waveforms to reflect the important characteristics under analysis is subjected to user choice. From the perspective of providing a tool, the assumption is that input waveforms are already encoded as vectors.

The second question is how to choose a learning algorithm. As discussed in [14], from experience the learning algorithm is usually not as important as the methodology to utilize the learning approach. In the above search strategy, the idea of re-projecting samples into a new space based on the important samples can be thought of as the methodology to utilize the clustering approach to solve the overall important input search problem. For clustering, the definition of the space to conduct the clustering impacts the clustering result more than the cluster algorithm. Hence, if the space re-projection idea does not work, then no clustering algorithm would be helpful. If the idea is viable, then while two algorithms might give different results, both results should show some effectiveness to indicate that the idea is working.

Because the focus of this work is on developing the search strategy, optimization of the learning algorithm is not discussed. Such optimization can also be application case dependent. One algorithm can be better for one case while another algorithm is better for another case. Unless a comprehensive set of test cases is determined, it is not really meaningful to discuss optimization of the learning algorithm.

For clustering, the Python machine learning library [23] provides several options to choose from. We selected the Hierarchical clustering algorithm because it was easy to apply and from experience the result was usually more robust than the popular k-mean algorithm [23].

4.5 Supervised Learning for Event Prediction in Monte Carlo Simulation

Recall the discussion with Figure 4.6 above. The idea can be summarized into two parts: (1) Partition a circuit into a set of primary elements (PEs) and (2) decide if a PE is *predictable*.

From a tool provider's perspective, partitioning a circuit can be subjected to the user. For example, suppose the circuit is given as a transistor netlist. The starting point can be to partition the circuit into a set of channel connected components. The user might want to merge certain components because conceptually, merging components result in a PE whose input-output behavior is more intuitive to understand. Hence, while partitioning a circuit is a required step to run the tool, it is not the main focus for the tool development. The focus is on developing a methodology to determine the predictability of a PE, and if it is predictable, to construct a prediction model.

Suppose a set of n PEs $\{g_1,...,g_N\}$ is given. Our goal is to develop a method to evaluate the *predictability* of each PE. There can be two approaches to pursue this development.

First, predictability obviously depends on the power of the predictor, i.e. a model built by a learning machine by learning from a given dataset. For example, the learning machine can be the Support Vector Machine (SVM) regression called Support Vector Regression (SVR) [20]. The predictability of a learning machine can be evaluated experimentally by applying it to a set of PEs. In other words, the experimental approach starts with an idea to implement the learning machine and then determines the predictability of a PE based on a specific learning machine.

The experimental approach is feasible when one has a good idea to implement an effective learning machine. However, as pointed out in Section 4.3.3 before, traditional supervised learning algorithms assume that the outputs to be predicted are scalar values (refer to Figure 4.7). Hence, learning machines such as SVR are not readily applicable.

The alternative approach is to develop a concept to capture the meaning of predictability. Then, a learning machine for constructing the predictor can be designed based on the concept. In this work, we take the second approach.

4.5.1 Intuition behind Predictability of a SVM

Before considering the predictability of a PE, we first illustrate the intuition behind the predictability of a SVM learning model. Given a set of data points (x_1, y_1),...,(x_n, y_n) to be learned (note: each x_i can be a vector, and each y_i is a scalar value), a SVM model is of the form [20]:

$$f(x) = \left(\sum_{x_i} \alpha_i \, k(x_i, x)\right) + b \tag{3.1}$$

SV is a subset of the samples. Without loss of generality, let SV ={x_1,...,x_l}. Each x_i ∈ SV is called a *support vector*. The kernel function *k()* measures the similarity between a pair of *x* samples. Each *k(x_i,x)* measures the similarity between a support vector x_i and the input vector *x* to be predicted. The coefficient α_i is the weight associated with *k(x_i,x)*.

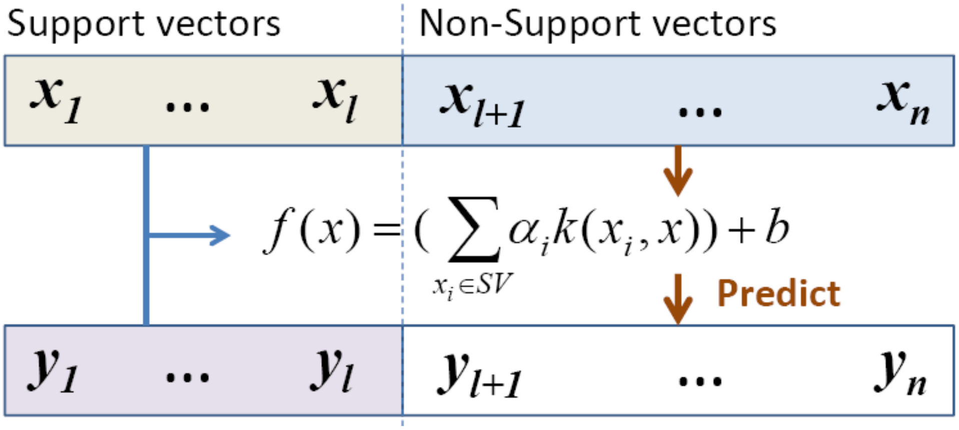


Figure 4.12: Intuition behind SVM predictability

Because the non-support vectors x_{l+1},...,x_n are not used in the calculation of *f(x)*, they can be seen as the samples used to *validate* the model *f(x)*. Figure 4.12 illustrates this point.

Suppose a prediction error function is given. For example the error function can be the sum of square errors: $SSE = \sum_{i=1}^{n}(y_i - f(x_i))^2$. Then, with the optimization objective to minimize the SSE, an SVM algorithm determines the following three things: (a) the set SV, (b) the coefficients α's, and (c) the constant b in Equation (1) above.

Conceptually, the quantity $\frac{n-l}{n}$ can be thought of as a measure for the predictability of the model *f(x)* [24]. This is because the smaller the l (the size of the set *SV*) is, the larger the number of the *validation* samples is. In other words, if the model can predict more validation samples within a given error, then the dataset is more predictable.

Therefore, we make two observations: (1) Learning is to decide which samples can be predicted by other samples. In a sense, it can be seen as a compression process. (2) The more samples that can be predicted by others, the higher the predictability is for the given dataset.

4.5.2 Illustration of SVM Models

Consider $|SV|=1$. The SVM model is simplified to $f(x) = \alpha_1\, k(x_1, x)+b$. The learning problem is simple because one sample x_1 is sufficient to predict the behavior of all other samples within a given error.

Assume that the kernel $k(x_1,x)$ is a Gaussian kernel $k(x_1,x)=e^{\{-(x1-x)^2}$. Without loss of generality, also assume $\alpha_1=1$ and $b=0$. Figure 4.13 illustrates the model on a 2-dimensional plane. All points with the same distance (the quantity $(x_1-x)^2$ is the same) to x_1 have the same predicted y value. This is illustrated in the figure with three circles where the points on those circles have the same predicted values y_a, y_b and y_c, respectively.

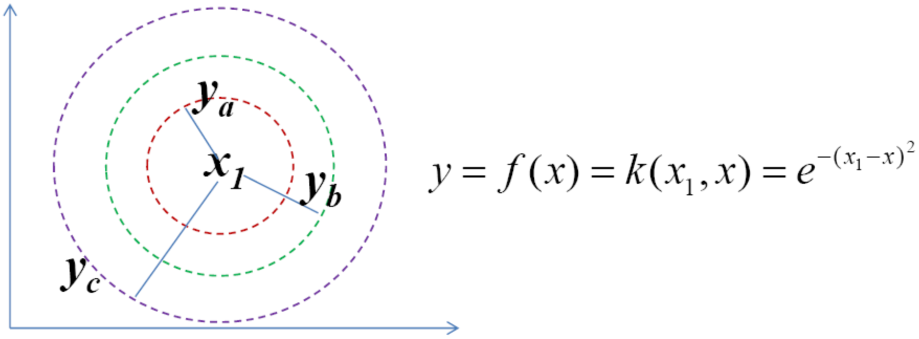


Figure 4.13: SVM model based on one support vector x_1

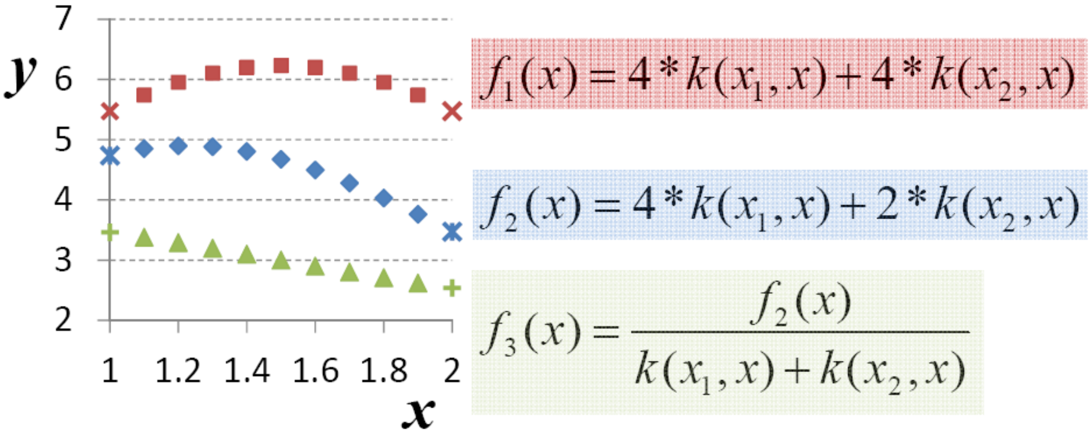


Figure 4.14: SVM models based on two support vector x_1, x_2

Now consider the case with two support vectors x_1, x_2. The model becomes $f(x)= \alpha_1 k(x_1,x)+ \alpha_2\ k(x_2,x)$ (assuming b=0). Figure 4.14 shows three cases for x_1=1, x_2=2, and x=1.1,1.2,...,1.9. Again, we use $k(x_i,x)=e^{\{-(xi-x)^2}$.

In the first case $f_1(x)$, we have α_1=4 and α_2=4. The predicted y values are shown. Observe that the predicted y value of a x_i is closer to $f(x_1)$ if x_i is closer to x_1. Similar observation can be made for x_2. For example, x=1.5 is the most dissimilar point to both x_1=1 and x_2 =2 and hence, $f_1(1.5)$ has the largest y value.

In the second case $f_2(x)$, we have α_1=4 and α_2=2. Notice that in this case the largest y value no longer occurs at x=1.4. This is due to the two non-equal weights α_1, α_2.

Observe that by changing α_1, α_2, the model is capable to capture a variety of convex functions. In the third case $f_3(x)$, we take $f_2(x)$ and normalize it with the similarity sum $k(x_1,x)+k(x_2,x)$. This results in a linear function. Hence, if we desire to model a linear behavior between two samples, we can use the normalization method.

Figure 4.14 illustrates that a variety of behavior, linear or non-linear, can be captured by a simple two-SV learning machine. Later, we will use this simple learning machine as the basis to implement a learning machine for PE behavior prediction.

4.5.3 The Predictability of an Inverter

Inverter perhaps is one of the simplest circuit elements. Therefore, we use an inverter to illustrate the intuition behind predictability of a PE.

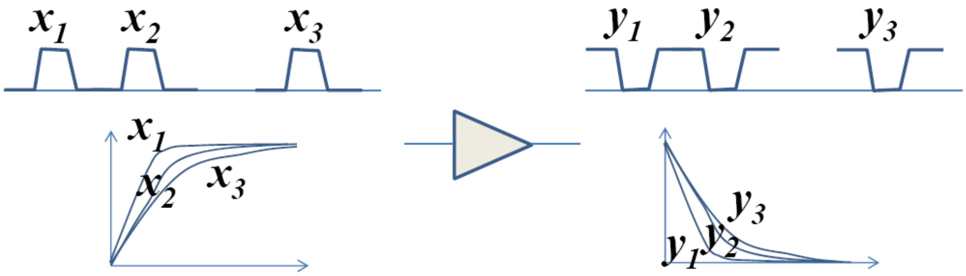


Figure 4.15: An inverter is simple for prediction

Figure 4.15 depicts an inverter with two cases. In the first case, the inverter is fed with digital pulses at different times. In the second case, the inverter is fed with rising waveforms.

Consider the first case. Intuitively, the output pulse y_2 of the input x_2 can be easily predicted by the two samples (x_1, y_1) and (x_3, y_3). This is because the distance between y_1 and y_2 is almost the same as the distance between x_1 and x_2. Similar observation applies to x_3 and x_2 as well. Furthermore, the shape of y_2 is almost the same as the shapes of y_1 and y_3.

Suppose we have a kernel function *k()* that measures similarity between two pulses based on their time distance. Then, intuitively y_2 can be predicted as:

$$y_2 = f(x_2) = \frac{y_1 k(x_1, x_2) + y3\, k(x_3, x_2)}{k(x_1 x1, x_2) + k(x_3, x_2)} \tag{4.2}$$

If we refer to the discussion of the model f_3() in Figure 4.14 earlier, we see that Equation 4.2 essentially is doing linear interpolation. In other words, the inverter is simple and predictable because the output behavior of an input can be predicted by two samples with linear interpolation.

Consider the second case in Figure 4.15. Observe that x_2 is "between" x_1, and x_3 and y_2 is also between y_1 and y_3. Intuitively, the linear interpolation of Equation 4.2 may still work. Or we can consider a more complex model:

$$y_2 = \mathrm{f}(x_2) = \alpha_1 y_1 k(x_1, x_2) + \alpha_3 y_3 k(x_3, x_2) \tag{4.3}$$

Essentially, we can find the coefficient values α_1, α_3 to best fit the output waveform y_2. Equation (3) is still following interpolation.

Based on the discussion above, observe that there are two properties that make an inverter easy to predict by the equations: (1) the outputs follow the same "ordering" of the inputs. (2) The similarity between two outputs is reflected in the similarity between the two corresponding inputs. The first property allows interpolation to work. The second property allows similarity-based prediction to work.

In the following, we will use the first property to explain the notion of *complexity* and the second property to explain the notion of *information*. We want to define the notions of *complexity* and *information* in such ways that only a PE with low complexity and low information is considered *predictable*.

4.5.4 Intuition Behind Complexity and Information Measures

First, it is important to note that given a set of samples $\{(x_1, y_1),...,(x_n, y_n)\}$ where x_i and y_i are waveforms, what we have is a kernel function *k()* that measures the similarity between a pair of inputs $k(x_i, x_j)$. We assume that learning follows the kernel-based learning depicted in Figure 4.8 before. Hence, we do not need to know the actual representation of a waveform. All we need are the similarity measure values.

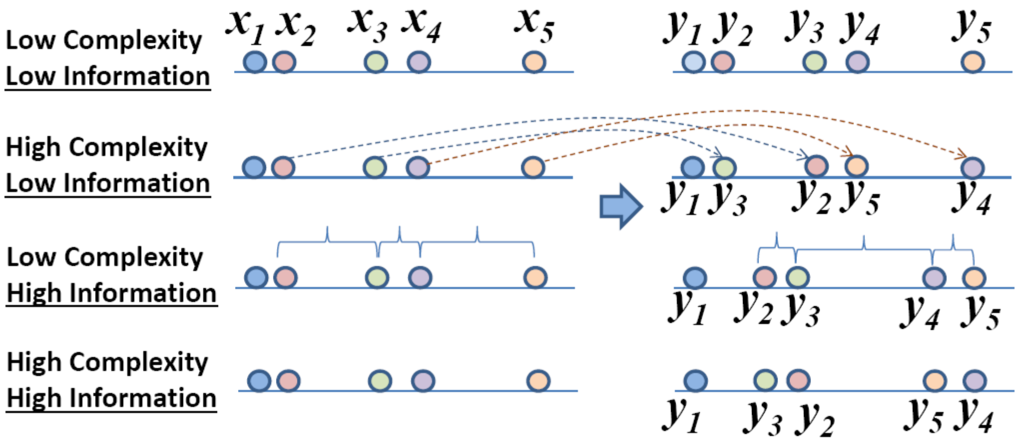

Low Complexity
Low Information
High Complexity
Low Information
Low Complexity
High Information
High Complexity
High Information
x_1 x_2 x_3 x_4 x_5
y_1 y_2 y_3 y_4 y_5
y_1 y_3 y_2 y_5 y_4
y_1 y_2 y_3 y_4 y_5
y_1 y_3 y_2 y_5 y_4

Figure 4.16: Intuition behind complexity and information measures

Figure 4.16 depicts the intuitions behind the complexity and information measures to be developed later. Keep in mind that although samples are shown along a 1-dimensional line, what we focus on is the similarity between them. Hence, those samples can reside in any space.

For a kernel that is distance-based (such as the Gaussian kernel used in Figure 4.13 and Figure 4.14 above), when two samples are shorter in distance to each other, they have a larger similarity value, and vice versa. Hence, we can use the distance $dist(x_i,x_j)$ in the discussion to make it more intuitive.

The first case in Figure 4.16 illustrates the case of low complexity and low information. First, the ordering of five *y* outputs is the same as the ordering of the five *x* inputs. This gives low complexity. Second, the distance between any pair of outputs is proportional to the distance between the corresponding pair of inputs. This gives low information.

In the second case of high complexity and low information, observe that the ordering changes among the *y* outputs. However, if we *discard* the *y* labels, the (relative) locations of the five inputs are almost the same as the (relative) locations of the five outputs. In this case, we say the information content of the *y* outputs is almost the same as the information content of the *x* inputs.

The reason that information measure discards sample labels is the following. If we view the samples as distributed in a space following a probability density function $p(z)$, the information can be measured by the differential entropy $h(Z)\ _Z =-\int_z p(z) \log p(z)dz$. Basically, we view the samples in the *x* space following a density function $p_x()$ and the samples in the *y* space following a density function $p_y()$. The low information means that the information content based on $p_y()$ is not changed much from the information content based on $p_x()$. Hence, in this measure, labels are not required.

Given $p_x()$ and $p_y()$, the relative information between the *x* space and the *y* space can be measured by the well-known Kullback-Leibler Information:

$$I\left(p_x(), p_y()\right) = \int p_x() log\left(\frac{p_x()}{p_y()}\right) dx \tag{4.4}$$

which measures the information loss from the distribution in the *x* space to the distribution in the *y* space. Hence, if we can estimate the two density functions $p_x()$ and $p_y()$ (this will be discussed later), we can calculate this information loss. From this perspective, we say that a PE is with low information if the information loss is low.

Consider the inverter example discussed above. The information loss by going through an inverter is low. This is because an inverter is simply inverting an input waveform to obtain its output waveform. This intuition can also be easily understood for an inverter operating in a binary space. For example, the information contained in a binary sequence fed into an inverter is the same as the information contained in the binary sequence output by the inverter.

The third case in Figure 4.16 shows low complexity and high information. We see that the ordering does not change from *x* to *y*. However, the distribution (the relative locations of the five samples) changes from *x* to *y*. Hence, some information is lost. The fourth case then shows high complexity and high information where both the ordering and the distribution changes from *x* to *y*.

4.5.5 Complexity Measure

To measure complexity, the above discussion points to the idea of measuring how much the ordering is changed from the samples in *x* space to the samples in *y* space. While the concept of ordering is easy to perceive in Figure 4.16 because samples are projected on a 1-

dimensional line, in general the ordering is not well defined if we are working in an unknown space where only similarity values between pairs of samples are available. In fact, it is likely that a total ordering does not exist for a given set of samples.

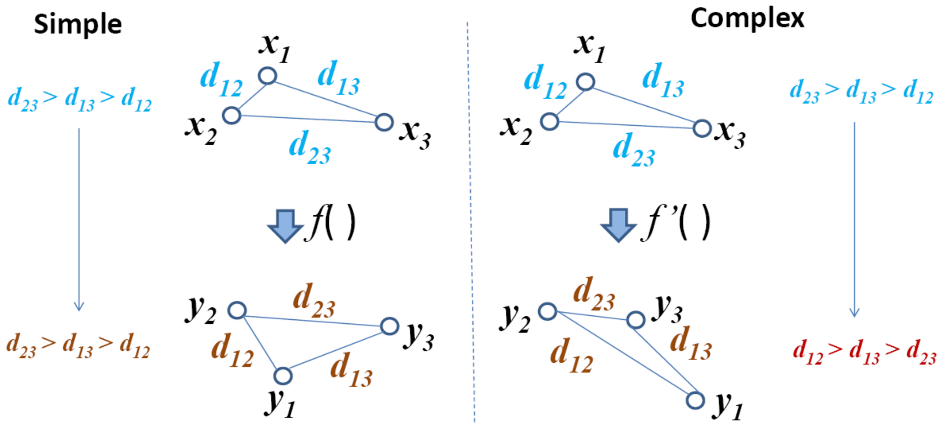


Figure 4.17: Intuition for complexity measure

Figure 4.17 depicts the idea employed in this work to measure complexity of a PE. Suppose a set of samples are given (x_1, y_1),...,(x_n, y_n) for a PE. For any three samples, for example (x_1, y_1),(x_2, y_2), (x_3, y_3), we check if their relative distances (or similarities) have changed from x to y.

For example, on the left plot of Figure 4.17, we have d_{23}>d_{13}>d_{12} in the x space. The corresponding y space also has d_{23}>d_{13}>d_{12}. In this case, we say the ordering has not changed for the three samples. On the right plot, we have d_{12}>d_{13}>d_{23} in the y space. In this case we say the ordering has changed, or more specifically the relative positions among the three samples have changed.

The check can be applied to all combinations of three samples (or a randomly selected number of combinations if n is too large) to estimate a total number of ordering changes. The higher the number of changes, the higher the complexity of the PE is. It is easy to see that if we apply the complexity measure to the low complexity cases in Figure 4.16, the number of changes would be zero.

4.5.6 Information Measure

Figure 4.18 illustrates the information measure in a 2-dimensional space. The top case shows information unchanged. The *y* space is obtained by flipping the *x* space vertically and horizontally, and by shrinking the scale of the image. If we view those dots as samples from a probability distribution, we see that the *x* space and *y* space depicts the same distribution. For example, there is a cluster (dense region) given by x_1, x_2, x_3 in the *x* space. The same dense region is given by y_1, y_2, y_3 in the *y* space.

The bottom case shows that the distribution of samples in the *x* space is different from the distribution of samples in the *y* space. This is easy to see because the dense region given x_1, x_2, x_3 in the *x* space is no longer there in the *y* space.

Given a set of samples $(x_1, y_1),\ldots,(x_n, y_n)$ for a PE, to measure the information (loss) of the PE, we will perform the following three steps:

1) Estimate a probability density function $p_x()$ based on samples $x_1,\ldots,x_n$ and kernel function *k()*.

2) Estimate a probability density function $p_y()$ based on samples $y_1,\ldots,y_n$ and kernel function *k()*.

3) Measure the information loss using Kullback-Leibler Information in Equation 4.4.

To perform steps 1) and 2), we need a method to estimate density function from a set of samples based on their similarity values. This is a classical unsupervised learning problem. For example, we can use the SVM density estimation method proposed in [25]. Due to space limitation, the detailed discussion on SVM density estimation is omitted.

Note that steps 1) and 2) use the same kernel function *k()*. In theory, this does not have to be the case. However, in this work we do not consider using two different kernels. We leave that consideration to future work.

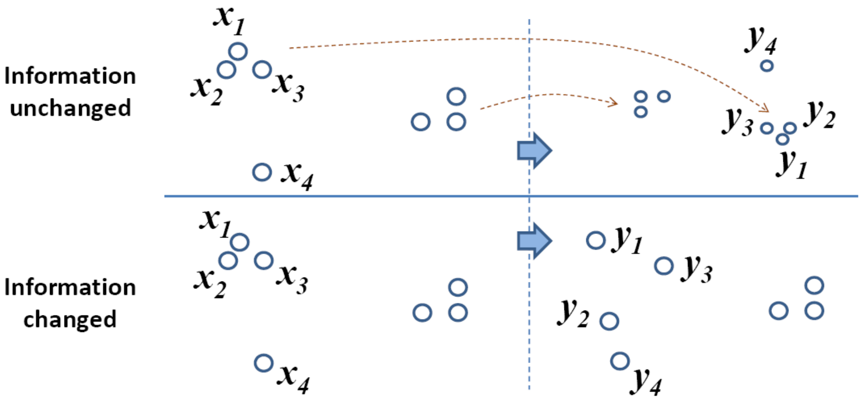
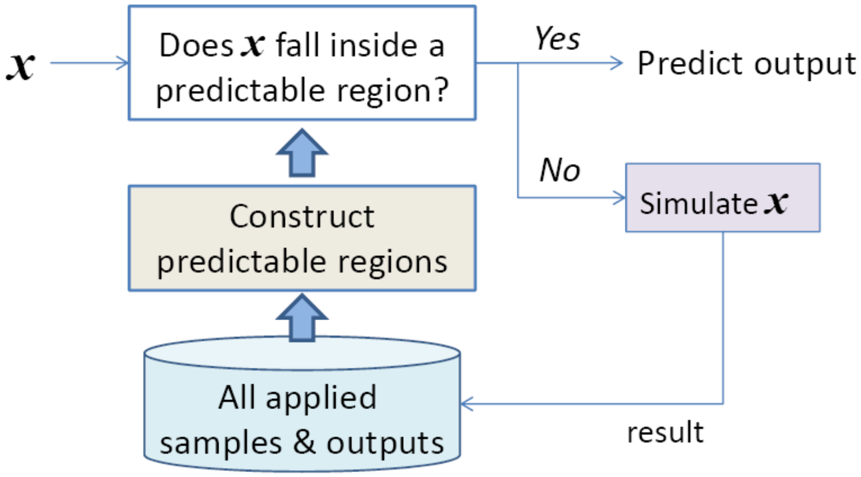


Figure 4.18: Intuition for information measure

4.5.7 Supervising Learning Based on Local Prediction

The complexity and information measures discussed above enables us to assess the predictability of a PE. Suppose a PE is deemed predictable after the evaluation. Next, we will discuss how to construct a predictor.

Suppose a set of samples (x_1, y_1),...,(x_n, y_n) are available for learning. These samples are simulated samples obtained during the *entire* circuit simulation. One can try to learn a single model *f(x)* → *y* but this would be difficult because both *x* and *y* are waveforms. Hence, instead of learning a single model, we try to find predictable regions based on a pair of samples. Figure 4.19 illustrates our learning strategy.



x
Does x fall inside a predictable region?
Yes
Predict output
No
Simulate x
Construct predictable regions
All applied samples & outputs
result

Figure 4.19: Supervised learning flow based on local predictability

The general idea is based on the discussion in Section 4.5.1 before. Essentially, learning is to identify samples that can be predicted by other samples. In our learning, we restrict the prediction to be based on *only* two samples. This can result in many *local* predictors rather than a single *global* predictor for all samples. When we do that, we need to define what a predictable region is based on a pair of two samples. This is because there can be multiple predictors and for a future sample, we need to decide (1) whether the future sample can be predicted by any of the predictors and (2) if yes, which predictor should be applied.

4.5.8 Defining a Potential Predictable Region

Figure 4.20 gives an example to define a potential predictable region based on two input samples x_a, x_b. The region is defined as: $\forall x$: $k(x_a,x) \leq k(x_a, x_b) \wedge k(x_b,x) \leq k(x_a, x_b)$.

Suppose *k()* is distance-based, e.g. a Gaussian kernel as discussed before. On a 2-dimensional plane, the region can be visualized as the intersection of the two circles as shown in Figure 4.20 (also refer to the discussion with Figure 4.13 before).

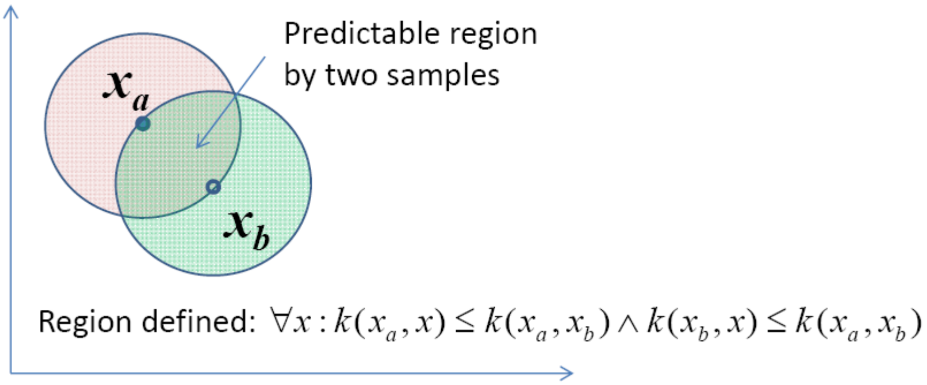

Predictable region
by two samples
x_a
x_b
Region defined: $\forall x : k(x_a, x) \leq k(x_a, x_b) \wedge k(x_b, x) \leq k(x_a, x_b)$

Figure 4.20: An example of potential predictable region by two samples

4.5.9 Deciding a Predictable Region

$$\begin{array}{ccc} \begin{array}{cc} \boldsymbol{x_a} & \boldsymbol{x_b} \end{array} & & \\ \left| \begin{array}{cc} k(x_a, x_1)y_a & k(x_b, x_1)y_b \\ \vdots & \vdots \\ k(x_a, x_i)y_a & k(x_b, x_i)y_b \end{array} \right| & \rightarrow & \begin{array}{c} y_1 \\ \vdots \\ y_i \end{array} \end{array}$$

Figure 4.21: Illustration of a learning dataset based on two samples x_a, x_b and *j* other samples falling into their potential predictable region

Given two input samples x_a, x_b and their potential predictable region, Figure 4.21 illustrates the local learning problem to decide if the potential region is actually a predictable region.

Suppose there are i samples falling inside the potential region. Figure 4.21 shows how a dataset or learning can be constructed for this region. Then, we can apply the simple models shown in Equation 4.2 and Equation 4.3 above to see if a predictor can be constructed to predict those i samples. If they can be predicted, then the potential region becomes a predictable region. Observe that Figure 4.21 is a much easier learning problem than learning a global model to predict all samples. The dimensionality is restricted at 2 and typically if x_a and x_b are close, we do not expect that i is large.

4.5.10 Learning and Model Application

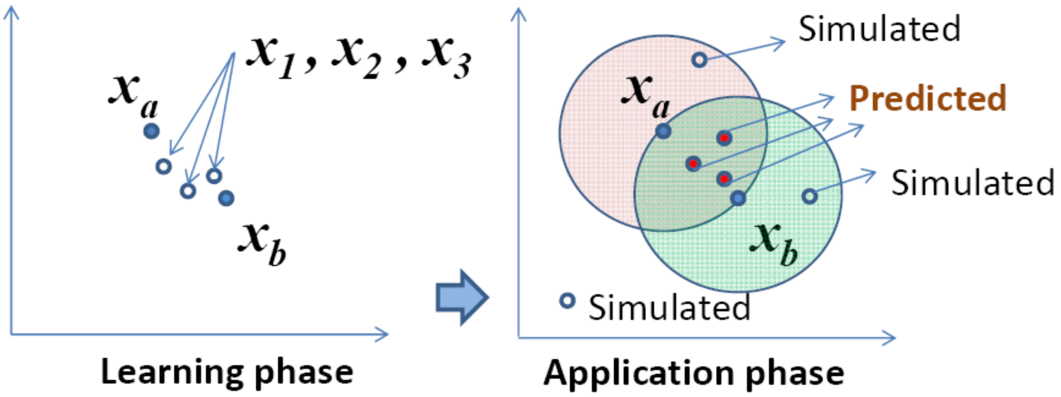


Figure 4.22: Illustration of learning phase and model application

Figure 4.22 summarizes the learning and the model application. In learning a local predictor is based on two samples x_a,x_b. Three additional samples x_1, x_2, x_3 are inside the potential predictable region defined by x_a,x_b. If the three samples can be predicted by a model such as Equation 4.2 or Equation 4.3, a predictable region is obtained. In the model application phase, if a future sample falls inside the predictable region, it is predicted. If it is outside, the sample is simulated. The simulated samples can then be used to learn local predictors in the next iteration. Figure 4.19 above also summarizes this iterative process of learning and model application.

4.6 The Overall Framework

This section will outline how the theories developed in the previous sections can be applied to an example circuit. The circuit in Figure 1.1 is partitioned into three PEs using the channel connected component graph, CCG, Figure 1.2. The first step is to determine the input set from **X** to cover the output behaviors in **Y**. Throughout the remainder of this chapter the similarity function is the Gaussian distance measure between waveforms and the clustering algorithm is hierarchical clustering. Table 4.1 shows the similarity between the waveforms displayed in Figure 4.23 where 100% similarity means that the two waveforms are identical. For this section we assume that the waveforms are encoded in a vector form where each element in the vector is a voltage value representing the measurement at each time step. All waveforms are of the same size.

Table 4.1: Similarity matrix between events 1-4 in Figure 4.23

	Event 1	Event 2	Event 3	Event 4
Event 1	100%	63.6%	59.6%	**97.2%**
Event 2		100%	**98.3%**	69.9%
Event 3			100%	65.3%
Event 4				100%

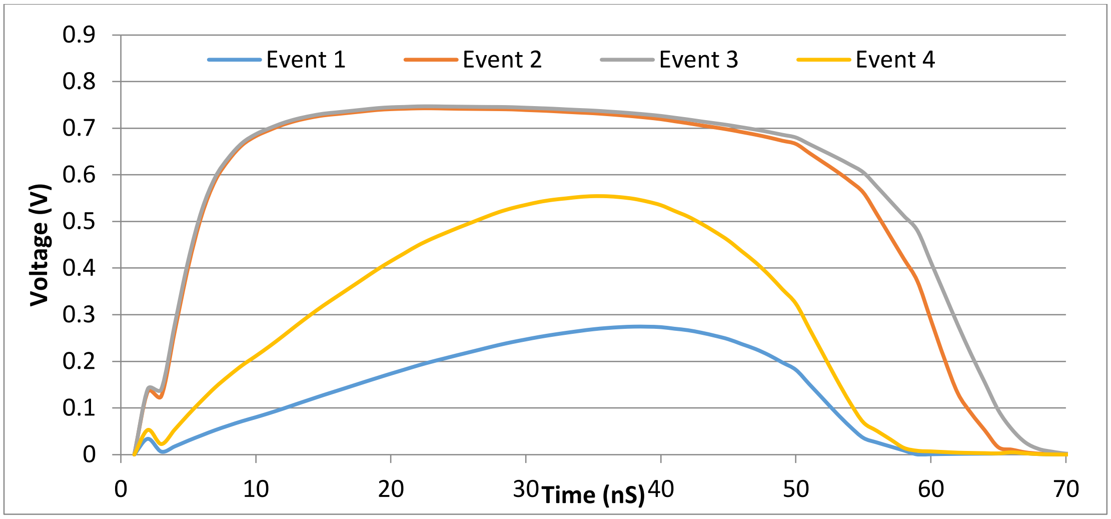


Figure 4.23: Four analog events

Hierarchical clustering is a tree clustering algorithm whose purpose is to join together objects into successively larger clusters based on a similarity measure. The similarity measure is often a distance measurement between pairs of objects. The algorithm begins with each object in a class by itself. For each level in the tree, the similarity measure threshold is relaxed or decreased and clusters are merged together. This continues until all of the clusters are merged together forming a tree structure. The benefit of hierarchical clustering is that there is no need to set specific number of clusters prior to the algorithm running. This is extremely important if the trends and information contained in the data set is relatively unknown. Unfortunately this algorithm has a long run time O(n^2). The algorithm is also not very robust towards outliers. Outlier is an object that is distant from any other cluster or object. The algorithm treats outliers as a cluster which can affect the way clusters are merged.

Table 4.2 displays the number of clusters per partition per similarity level for the output events of nine partitions from various circuits in Chapter 2. The number of events represents the number of simulated output events being applied to the clustering algorithm. The majority of the events are similar and can be grouped into significantly fewer clusters at 99%. The number of clusters steadily declines as the precision decreases as the allowable distance from the center is increased. The analog repeating partitions, 2 and 3, have a significant drop in events even at 99%. For Partition 2 when the precision is increased to 99.9% the number of clusters becomes 32 and at 100% the number of clusters is 2247. Similarly for Partition 3 at 99.9% precision the number of clusters is 23 and for 100% the number of clusters is 215. Each of the input events to the partitions is unique which implies that there is very little noise generated from the partition. The number of clusters at 100% implies that no two events are exactly the same, but the dramatic decrease in clusters implies that they are very similar.

Table 4.2: Number of Clusters per Partition and Precision

	# Events	99%	95%	90%	80%	70%
Partition 1	100	20	11	7	7	6
Partition 2	2247	9	5	4	3	3
Partition 3	215	8	4	2	2	2
Partition 4	284	153	122	107	90	70
Partition 5	2016	365	261	190	136	97
Partition 6	184	21	21	20	15	15
Partition 7	2648	578	455	455	374	353
Partition 8	127653	1120	872	740	573	417
Partition 9	11644	130	118	85	75	72

4.6.1 Avoiding the Simulation of Unimportant Inputs

For the circuit in Figure 1.1 the set of input waveforms is 6000 where each waveform is 70pS. For a small circuit, 6000 simulations may not seem like very many and may take little time. When the circuits get much larger and the simulations are on the order of minute to hours, reducing the input set even by a fraction is crucial. Following the iterative approach from Section 4.4, hierarchical clustering is used to determine an initial set of input events to simulate through the whole circuit. The cluster is done with 90% similarity which results in 50 clusters.

A representative sample, nucleus of the cluster, from each cluster is chosen and simulated. The 50 output events are clustered with the same similarity and 17 clusters are identified. The nucleus of each output cluster is mapped back to the respective input waveform where each is determined to be *important* waveform while the remaining 33 are deemed *unimportant*.

For the remaining iterations until there are no more unique output clusters, the input space clustering is not based on just the similarity of the time steps. A new set of features is created for each of the 6000 waveforms. Each feature is the similarity measure between the input waveform with the *important* input waveforms. The clustering is performed and 26 clusters are identified. Of the 26 clusters 17 of those contain the important waveforms and do not need to be re-simulated, resulting in only 9 new simulations. The clustering at the output remains a comparison between the waveforms and 3 new clusters identified. This process repeats until there are no new output clusters, i.e. convergence. This circuit converges after two iterations where there are 21 output clusters. To generate a golden set of data for comparison, all 6000 waveforms were simulated. The number of output clusters is 21 which matches the number of clusters determined from just 68 simulations.

Table 4.3: Overview to clustering in each iteration

Iteration	Input Clusters	Output Clusters
Initial	50	17
1	26	20
2	35	21
Golden Results	6000	21 clusters

4.6.2 Determining Primitive Elements

The next phase in simulation reduction is the abstraction of the circuit; determining and behaviorally modeling PEs. Simulation data from the input space reduction phase is decomposed into input/output data for each partition. To determine the complexity of the PE the input and output data is clustered. If the output clusters number is more than the input clusters then the PE is complex. Table 4.4 shows the number of clusters for each PE in the circuit. At various similarity measures, the complexity of the circuit shows that Partitions 1 and 3 are information reducers while Partition 2 is an information injector. We use 90% similarity measure in this work therefore Partition 1 and 3 are candidates for behavioral modeling

Table 4.4: Information Measure of PEs

Precision	Partition 1		Partition 2		Partition 3	
	Input	Output	Input	Output	Input	Output
99%	72	66	66	122	122	40
98%	45	38	38	86	86	37
97%	33	29	29	72	72	36
96%	24	22	22	63	63	35
95%	21	19	19	54	54	34
94%	20	18	18	52	52	31
93%	17	15	15	45	45	31
92%	14	14	14	40	40	31
91%	13	13	13	34	34	27
90%	12	12	12	42	42	25

The complexity of the partitions are measured by the distance between three events, Figure 4.17. If the ordering distance between each event is maintained between all triples then the partitions is considered non-complex. Table 4.5 shows the complexity measure for each of the three partitions. The OUT column indicated how many of the samples have not had the distance maintained between all three events at the input and the output, i.e. input $d_1<d_2<d_3$

and output $d_1<d_3<d_2$. 50,000 random combinations of events are analyzed for each partitions. Even though Partition 2 is an information injector and removed from the behavioral modeling candidates, we show the complexity for reference. Of the mismatched events in Partition 1 and 3, the difference for all the events is <<1%. For example given at the input {d1, d2, d3}={.00011387,.00011382,.9999} and at the out {d1,d1,d3}={.0375,.0376,.999} the change in order for the output is so insignificant. This is not the case for Partition 2. The column INS represents the insignificant events which have differences <<1%.

Table 4.5: Complexity Measure of PEs

50k Runs	Partition 1		Partition 2		Partition 3	
	OUT	INS	OUT	INS	OUT	INS
1	11.42%	11.32%	26.31%	11.76%	10.03%	10.02%
2	11.70%	11.58%	26.57%	11.98%	9.65%	9.65%
3	11.95%	11.86%	26.41%	12.10%	9.86%	9.86%
4	12.02%	11.90%	26.43%	11.77%	10.08%	10.07%
5	11.86%	11.75%	26.89%	12.08%	10.11%	10.11%
6	12.02%	11.93%	26.58%	12.06%	10.09%	10.08%
7	11.64%	11.55%	26.37%	11.91%	10.28%	10.27%
8	12.15%	12.04%	26.40%	11.99%	9.99%	9.90%
9	11.80%	11.68%	26.22%	11.89%	9.99%	9.98%
10	12.18%	12.05%	26.64%	11.81%	10.14%	10.13%
Average	**11.87%**	**11.77%**	**26.48%**	**11.94%**	**10.02%**	**10.01%**

Chapter 3 discussed an SVM transient based prediction method for modeling the behavior of analog circuits. As method earlier this method is not general and has various limitations. For non-complex partitions which have ordered input/output waveforms any behavioral modeling, even SVM, will be effective. In this work we will use a lookup table with interpolation to predict the output waveform.

The initial table is built using the simulation data from the complexity analysis of the PE. Monte Carlo simulations are performed on the partition and simulated. For each input in the table 100 samples of variation data are collected. The events are analyzed for information measurements based on the discussion above. This reduces the table entries from 2121 to 27. This tells us that the variation data had little impact on the partitions behavior.

Every new input waveform is applied to the lookup table behavioral model. If the output can be predicted by interpolation of the table data then no simulation is necessary. If the output cannot be predicted then the waveform is simulated with and without Monte Carlo variation sampling and added to the lookup table as a new entry. Figure 4.24 displays the number of waveforms added to the lookup table as more input waveforms and variation samples are applied. For all 6000 input waveforms only 77 are simulated which is 1.28% of the simulation time. These models are significantly faster the circuit simulation, on the order of two to three magnitudes depending on the PE size.

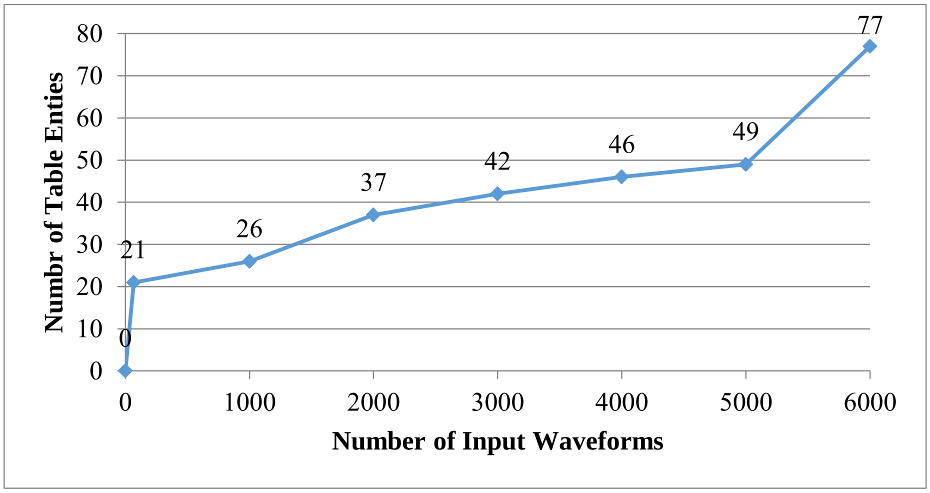


Figure 4.24: Number of input waveforms verse the number of lookup table entries

4.7 Experimental Results

The methods outlined in Section 4.6 are further extended to a number of individual partitions for PE analysis and large complex analog circuits, UWB-PLL. Nine partitions are used in the remaining chapter as a representative sub-set of the partitions. The partitions are as follows:

1) Voltage Regulator Partition 3
2) Voltage Regulator Partition 1
3) Differential amplifier from ∑Δ-SDM
4) Charge Pump bias UWB-PLL
5) Charge Pump differential down partition
6) PFD Down from UWB-PLL
7) PFD Up from PLL for Clock Recover
8) Divider Partition 1 from UWB-PLL
9) Divider Partition 3 PLL from Clock Recovery

To determine which PEs are complex the input and output clustering is applied to Partition 1-9. The analysis results for the input (IN) and output (OUT) clusters are displayed in Table 4.6a and Table 4.6b. All events of the input and output are simulated over the same amount of time, 100nS. Partitions 1-5 are all non-complex PE and can be modeled using a lookup table. Partitions 6-9 on the other hand will need to be simulated. These partitions which are digital are highly sensitivity to changes at the input, in particular delay based events. Partitions 4 and 5 have a digital input and produce an analog output. The analog output responds slowly over time to changes in the digital input which is why so many are similar while there are so many different combinations of digital inputs. Partitions 1-3 are partitions of circuits whose functionality is to transfer or amplify input waveforms which explains why the clustering of the input and output are very similar. Figure 4.25 displays the events of Partitions 1. The 100 input events over 100nS to the partitions are displayed in Figure 4.25(a) followed by the central event for each clusters for (b) 100%, (c) 99%, and (d) 90%.

Table 4.6a: Input Cluster Analysis

	Partition 1 IN	Partition 1 OUT	Partition 2 IN	Partition 2 OUT	Partition 3 IN	Partition 3 OUT	Partition 4 IN	Partition 4 OUT
100%	100	100	100	100	100	100	200	200
99%	20	15	11	7	21	15	10	2
98%	16	15	8	5	15	11	7	2
97%	12	11	7	4	12	9	7	2
96%	11	11	7	4	11	8	6	2
95%	11	9	6	4	10	8	6	2
94%	10	8	6	4	9	8	6	2
93%	9	7	6	4	8	7	6	2
92%	9	7	5	3	8	7	5	2
91%	7	7	5	3	8	7	5	2
90%	7	7	5	3	8	6	5	2
89%	7	7	5	3	8	6	5	2
88%	7	7	5	3	7	6	5	2
87%	7	6	5	2	7	6	5	2
86%	7	5	4	2	6	6	5	2
85%	7	5	4	2	6	6	4	2

Table 4.6b: Input Cluster Analysis

	Partition 5 IN	Partition 5 OUT	Partitions 6-9 IN	Partition 6 OUT	Partition 7 OUT	Partition 8 OUT	Partition 9 OUT
100%	300	300	300	300	300	300	300
99%	269	2	125	269	131	198	202
98%	266	2	93	266	131	198	201
97%	261	2	71	261	131	198	201
96%	251	2	59	251	120	198	201
95%	244	2	52	244	103	197	201
94%	236	2	52	236	96	197	201
93%	231	2	43	231	92	197	201
92%	223	2	41	223	86	197	201
91%	215	2	39	215	82	197	201
90%	207	2	38	207	80	197	201
89%	197	2	33	197	77	196	201
88%	192	2	32	192	74	196	201
87%	187	2	31	187	69	196	201
86%	179	2	30	179	68	195	201
85%	169	2	29	169	68	195	201

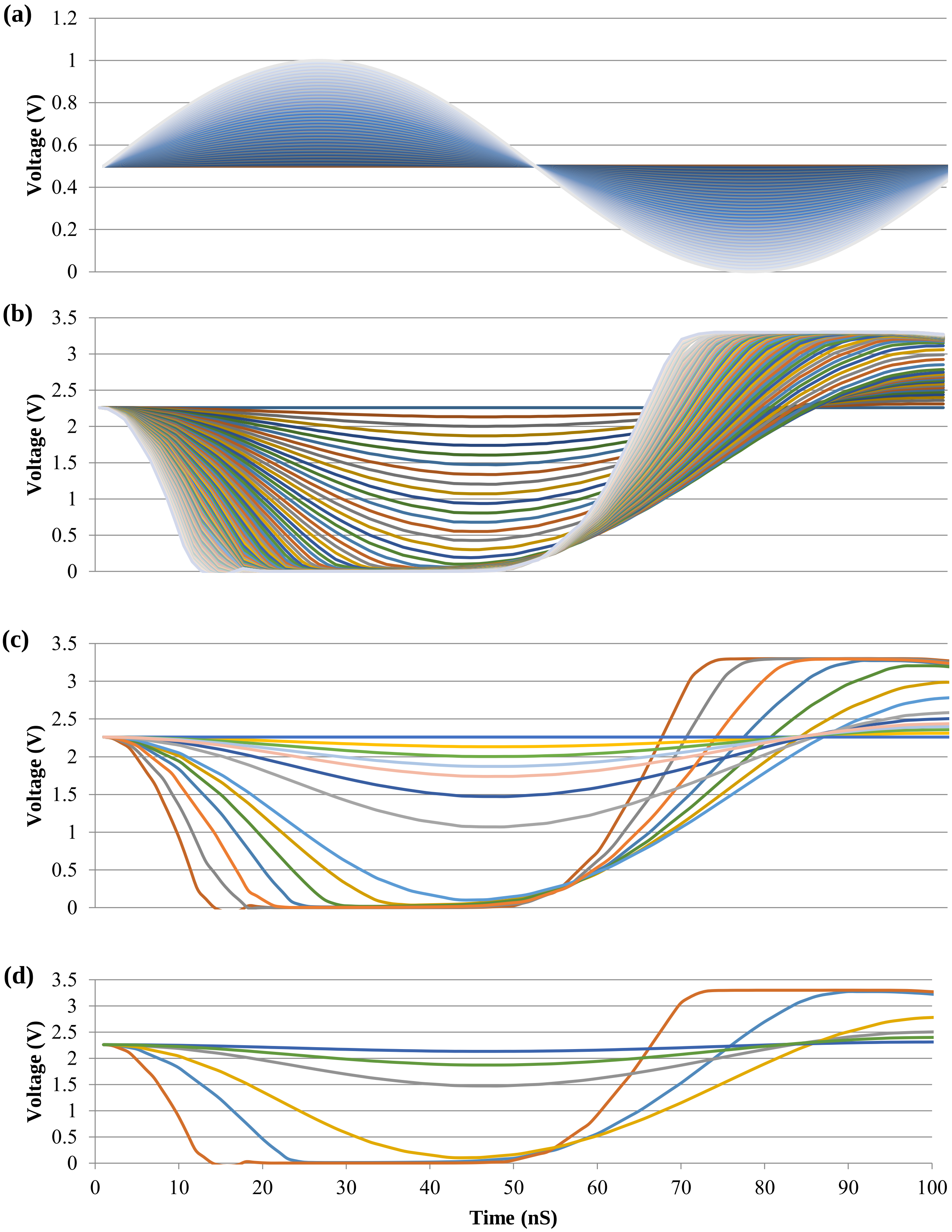


Figure 4.25: Clustered events from Partition 1 output (a) input (b) 100%, (c) 99%, (d) 90%

Table 4.7: Complex Primitive Elements with Reduced Simulation Time

Circuit	Complex PE	Time to Convergence	Golden Simulation Time
UWB-PLL	22	647.3h	1481.2h
PLL CR	18	377.5h	1232.1h
Regulator	2	20m	119.4m
TIA	1	17.9m	16.5m

The results in Table 4.7 present the applied methodology on 4 circuits from Chapter 2, the two PLLs are extremely large and complex. The time to convergence is the amount of time it takes to (1) converge on the input space, (2) create the behavioral models, and (3) apply Monte Carlo analysis. The golden simulation time is the amount of time to run 1,000 Monte Carlo simulations on the full circuit with 10 different frequencies (PLLs) and 10 different input combinations. Except the TIA, where the simulation is just as fast as the model predictions, the other circuits have 55-75% reduction in simulation time.

4.8 Conclusion

This Chapter developed a methodology for applying statistical learning to the verification of analog circuits. The first methodology developed utilizes unsupervised learning, circuit partitioning, and event propagation to determine the minimal representative set of input events which describe the output space. A significant amount of simulation time is saved by only simulating the important inputs. The second methodology developed locates primitive elements with low complexity which can be modeled behaviorally instead of simulated. Unsupervised learning is used on the input and output of each cluster to determine if the primitive element increases, transfers, or decreases the information content. Distance calculations are used to determine the complexity of the events through the primitive element.

Low complexity primitive elements maintain the distance ordering of the events. Behavioral models are created using supervised learning techniques for primitive elements with transferring or decreasing information content. We have shown the effectiveness of the method on four analog circuits where the simulation time is decreased by 55-75%.

4.9 Chapter 4 References

[1] C. Visweswariah, et. al. First-order incremental block-based statistical timing analysis. in *IEEE Trans. CAD* v25, 10, 2006, pp. 2170-2180.

[2] Xin Li, et al. Statistical Performance Modeling and Optimization. *Now Publishers*, 2007.

[3] A. Monti, F. Ponci, Member, T. Lovett. A polynomial chaos theory approach to uncertainty in electrical engineering. in International Conference on Intelligent Systems Application to Power Systems, 2005, pp.

[4] F. Augustin, A. Gilg, M. Paffrath, P. Rentrop and U. Wever. Polynomial chaos for the approximation of uncertainties: Chances and limits. in *European Journal of Applied Mathematics*,v 19, 02, 2008, pp. 149-190.

[5] A. Singhee, R. A. Rutenbar. Statistical Blockade: A Novel Method for Very Fast Monte Carlo Simulation of Rare Circuit Events, and its Application. in *DATE 2007*.

[6] X. Li, W. Zhang and F. Wang. Large-scale statistical performance modeling of analog and mixed-signal circuits. *IEEE Custom Integrated Circuits Conference (CICC)*, 2012.

[7] X. Li, F. Wang, S. Sun and C. Gu, Bayesian model fusion: a statistical framework for efficient pre-silicon validation and postsilicon tuning of complex analog and mixed-signal circuits. *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2013, pp. 795-802.

[8] M. C.T. Chao, L.-C. Wang, and K.-T. Cheng. Pattern selection for testing of deep sub-micron timing defects. in *DATE*, 2004, pp. 1060-1065.

[9] O. Guzey, *et. al*., Functional Test Selection Based on Unsupervised Support Vector Analysis. in *Design Automation Conference*, 2008.

[10] B. Sch¨olkopf, *et. al*. Estimating the Support of a High- Dimensional Distribution. in Journal Neural Computation, v 13, 7, 2001, pp. 1443-1471.

[11] W. Chen, *et. al*. Novel Test Detection to Improve Simulation Efficiency — A Commercial Experiment. ACM/IEEE ICCAD, 2012.

[12] P.-H. Chang, *et. al*., A Kernel-Based Approach for Functional Test Program Generation. In *International Test Conference*, 2010.

[13] L.-C. Wang. Data Mining in Functional Test Content Optimization. in *ASP-DAC*, 2014.

[14] Li-C. Wang, Magdy Abadir. Data Mining In EDA - Basic Principles, Promises, and Constraints in Design Automation Conference, 2014.

[15] O. Guzey, *et. al*. Extracting a Simplified View of Design Functionality Based on Vector Simulation. *Lecture Note in Computer Science, LNCS*, Vol 4383, 2007, pp. 34-49.

[16] M. J. Kearns and U. V. Vazirani. An Introduction to Computational Learning Theory, *MIT Press*, 1994.

[17] H. Li, *et. al*. Analog behavioral modeling flow using statistical learning method. in Proc. *ISQED*, 2010.

[18] S. Alt, L.-C. Wang, M. Marek-Sadowska. Circuit Partitioning for Behavioral Full Chip Simulation Modeling of Analog and Mixed Signal Circuits. *International Conference on System Modeling and Optimization*, 2014.

[19] O. Chapelle, B. Schlkopf and A. Zien. Semi- Supervised Learning. *The MIT Press*, 2010.

[20] B. Schlkopf, and A. J. Smola. Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. *The MIT Press*, 2001.

[21] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis. Cambridge University Press 2004.

[22] R. L. Allen and D. W. Mills. Signal Analysis*, IEEE Press Wiley-Interscience*, 2004.

[23] http://scikit-learn.org/stable/user_guide.html\#user-guide

http://scikit-learn.org/stable/modules/classes.html

[24] V. Vapnik, The nature of Statistical Learning Theory. 2nd ed., *Springer,*1999.

[25] Sayan Mukherjee and Vladimir Vapnik. Support Vector Method for Multivariate Density Estimation. A.I. Memo No. 1653, C.B.C.L. Paper No. 170, MIT 1999.

[26] V. Kamath, et. al., Functional Test Content Optimization for Peak-Power Validation — An Experimental Study. In *International Test Conference*, 2012.

[27] W. Chen, *et al*., Simulation knowledge extraction and reuse in constrained random processor verification. In *ACM/IEEE Design Automation Conference*, 2013.

[28] K.-K. Hsieh, *et al*., On Application of Data Mining in Functional Debug. In *ICCAD*, 2014.

[29] F. Rosenblatt. Principles of Neurodynamics. Washington, DC. *Spartan Books*, 1962.

[30] O. Bousquet, *et. al*. Introduction to Statistical Learning Theory. *Springer* LNCS, V 3176, 2004, pp. 169-207.

[31] F. E. Grubbs. Procedures for Detecting Outlying Observations in Samples in *Technometrics*, v11, no 1, 1969, pp. 1-21.

[32] T. Hastie, *et al*. The Elements of Statistical Learning - Date Mining, Inference, and Prediction. *Springer Series in Statistics*, 2001

[33] Y. Katz and et al. Learning microarchitectural behaviors to improve stimuli generation quality. In *ACM/IEEE Design Automation Conference*, pages 848 –853, 2011.

[34] N. Callegari, *et. al*. Classification rule learning using subgroup discovery of cross-domain attributes responsible for design-silicon mismatch. *DAC*, 2010, pp. 374-379.

[35] J. Chen, *et. al*. Mining AC Delay Measurements for Understanding Speed-limiting Paths. In *IEEE ITC*, 2010.

[36] G. Batista. A Study of the Behavior of Several Methods for Balancing Machine Learning Training Data. *Sigkdd Explor*., 6(1), pp. 20-29, 2004.

[37] N. Sumikawa, *et. al*. Screening Customer Returns With Multivariate Test Analysis. *In IEEE ITC*, 2012.

[38] J. Tikkanen *et. al*., Yield Optimization Using Advanced Statistical Correlation Methods. In *IEEE ITC*, 2014.

[39] O. Guzey, *et. al*., Enhancing signal controllability in functional testbenches through automatic constraint extraction. In *International Test Conference*, 2007.

[40] C. Zhang and S. Zhang. Association Rule Mining, Models and Algorithms. Lecture Notes in CS Vol. 2307, *Springer* 2002.

[41] C.H.-P. Wen, *et. al*., An Incremental Learning Framework for Estimating Signal Controllability in Unit-Level Verification. In *ICCAD*, 2007.

[42] K. J. Cios, *et. al*., Data Mining - A Knowledge Discovery Approach, *Springer*, 2007.

Chapter 5

Efficient Method for Critical Node Identification Time Varying Large Analog and Mixed Signal Circuit with Process and Environment Variation

Every transition to a new technology node increases the complexity of the process, resulting in larger numbers of process faults, greater parameter variations and more complicated descriptions of tolerances. This is especially true with nanometer technologies and low-power high-frequency circuits. Random and systematic process variations have a large influence on the quality, yield, and reliability of the analog, mixed signal, and RF circuits. This is a challenge for today's designs because varying parameters are usually uncorrelated and increasingly hard to capture. Undetected critical behaviors can result in many design re-spins and lower yield, increasing the time to market and production costs. It is important to detect critical transistors beforehand and determine their tolerance ranges so that critical behaviors can be analyzed before manufacturing the circuit.

Circuits that are high-speed and low-power are highly susceptible to various reliability and yield concerns in CMOS technologies, and are also negatively impacted by various environmental effects. For these reasons analyzing circuits under environment variation, reliability effects, and manufacturing variation ensure the circuit maintains proper functionality over time out in the field.

An example of concerns in reliability and yield is illustrated in high precision analog circuits, such as converters and comparators. These require stable threshold voltages that can be disrupted by such effects as hot carrier injection (an effect within transistors where an electron or hole gains enough kinetic energy to break into the gate dielectric and become trapped), negative bias temperature instability (when positive oxide charge is generated in transistors under negative bias in increased temperatures) or process variation. Additionally, environmental effects, such as VDD fluctuation caused by high switching digital circuits, result in fluctuation of speed and functionality of transistor. Therefore it is important to analyze the circuit under environment variation, reliability effects, and manufacturing variation to ensure the circuit maintains proper functionality over time out in the field.

This chapter provides a directed, simulation-based, sensitivity analysis of process variation of very large analog circuits for the purpose of critical transistors identification. The partitioning and input space methods discussed in Chapter 3 locates sensitive and non-sensitive partitions and provides an event comparison methodology based on waveform extraction and sensitivity analysis. Using these methods we will show that it is possible to locate high sensitivity transistors which cause critical behaviors due to a combination of process variation and waveform excitation. To our knowledge this has never been reported for circuits of this size.

After locating the high sensitivity transistors, the method is extended in order to identify reliability and yield concerns. Using these methods we will show how we can quickly and automatically find transistors that pose reliability and yield concerns due to process variation and power fluctuation for very large analog circuits. Due to the overtime nature of reliability concerns the analysis is performed in the transient domain. We will show that if both V_{th} and V_{DD} change simultaneously, faulty behavior can be observed even though each source of variation has been verified for proper behavior independently. We will analyze how the variations and fluctuations affect the circuit behavior and provide trade-off between process variations and supply voltage resilience for the most sensitive transistors. Though we only focus on V_{th} and power fluctuation, the method can be extended to any number of parameters.

The target uses of this tool is pre-layout and post-design syntheses, where the critical circuit elements and behaviors need to be identified taking into account process variation. There are three main applications for critical transistor analysis:

1. Re-designing the circuit by resizing or adding dummy elements to reduce the critical transistors effect on the behavior.
2. Reducing the parameter space of Monte Carlo simulations to just critical transistor parameters of the design. This is useful in Monte Carlo-based yield optimization methods or verification.
3. Incorporating into an optimization loop for design or layout.

This chapter is organized as follows. Section 5.1 provides a motivational example for power variation and circuit sensitivity. Section 5.2 provides background on sensitivity analysis techniques. Section 5.3 provides background on reliability and yield optimization. Section 5.4 discusses critical elements and sensitivity. Section 5.5 outlines the automatic method for identifying critical transistors. Section 5.6 provides experimental results. Section 5.7 concludes.

5.1 Motivational Example

As the power supply voltage, V_{DD}, gets closer to the threshold voltage, V_{th}, the transistor's threshold voltage becomes more sensitive to variation. For low power high frequency components, power and process variations have a drastic effect on the performance and reliability of the device. In this motivational example we will first show the effects of varied power levels on the circuits from Chapter 2. We will then show the effect of process variation from -6σ to 6σ V_{th} on the lock time performance of three different transistors. Finally we will show the effect of combined power variation and process variation on the lock time.

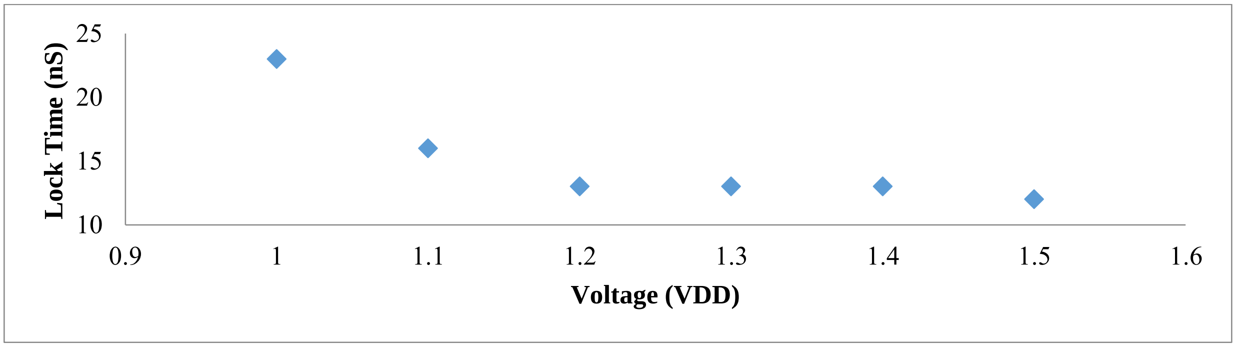


Figure 5.1: VDD effects on the lock time performance of UWB-PLL

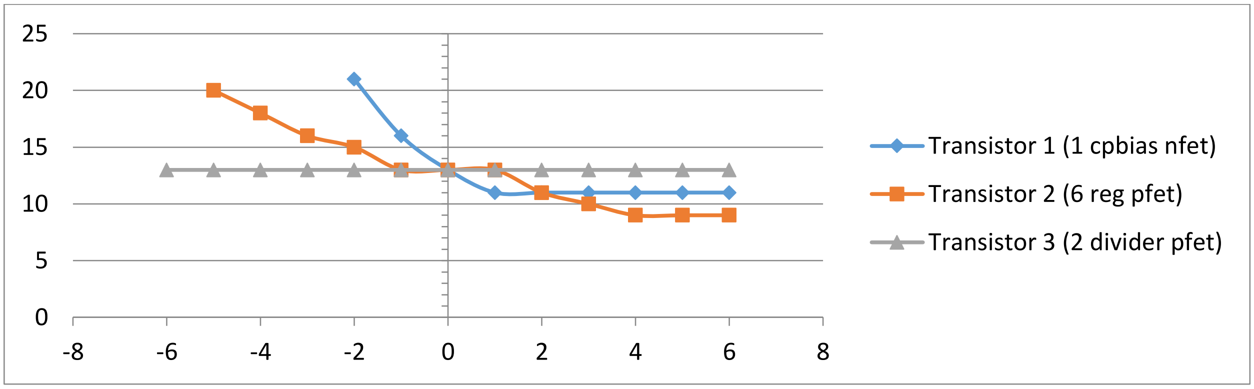


Figure 5.2: Vth variation from -6σ to +6σ for three transistors

Figure 5.1 shows the effect of V_{DD} on the lock time of the UWB-PLL. V_{DD} is varied from 0.9V to 1.5V where the nominal operating V_{DD} is 1.2V. At 0.9V the circuit can never achieve lock and thus is not included in the graph. The supply power is held at a constant V_{DD} for the entire transient simulation. As V_{DD} decreases, the circuit becomes unstable and eventually is unable to lock. Figure 5.2 shows the effect of V_{th} variation of three individually varied transistors and their respective lock times.

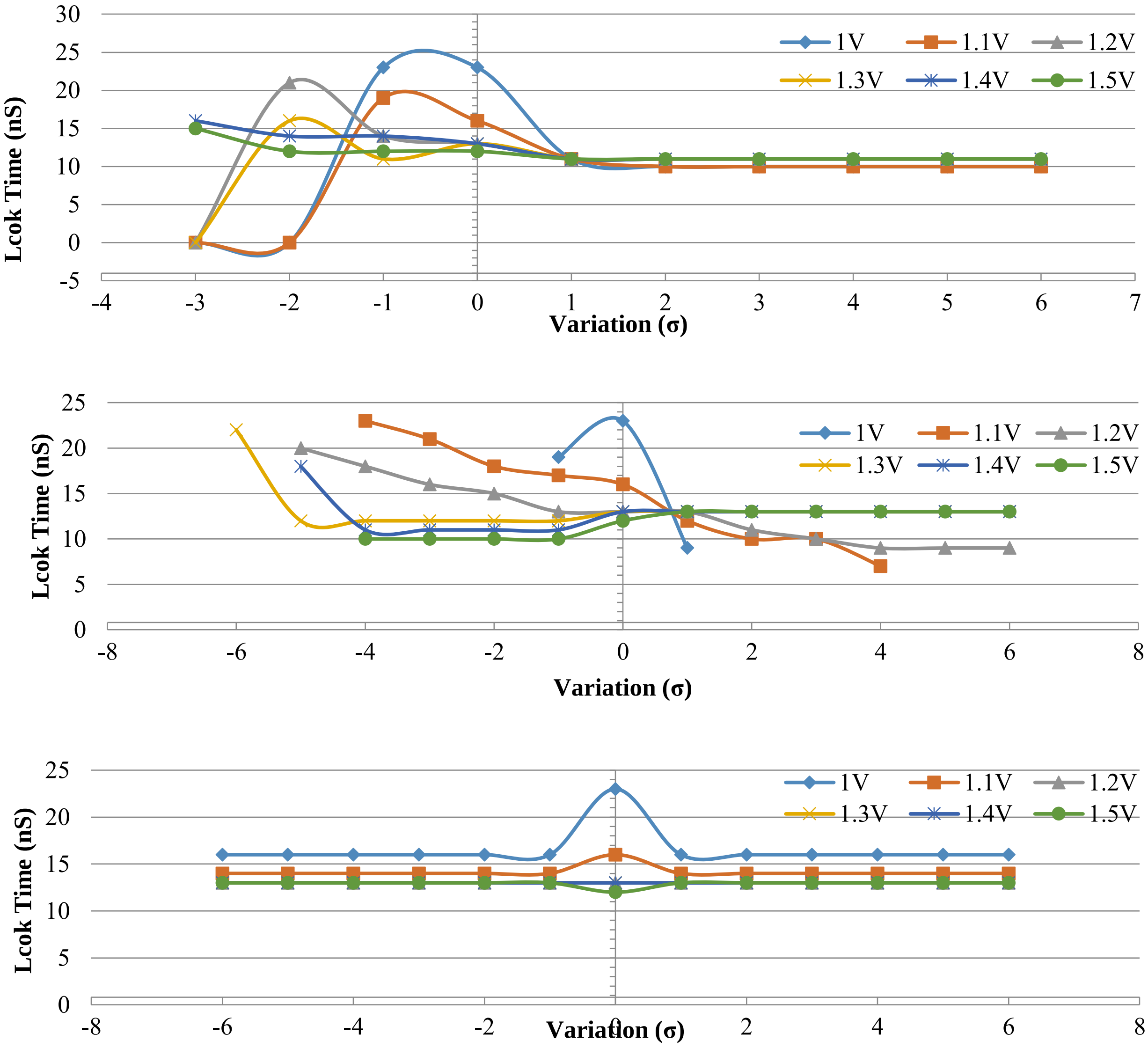


Figure 5.3: Simultaneous VDD and Vth variation simulations for (a) Transistor 1, (b) Transistor 2, and (c) Transistor 3

Each simulation contains only a single transistor varying at 13 different σ points. Values of σ that are not graphed for a transistor indicate that the PLL never achieves lock. Certain transistors affect the circuit behavior more drastically then others even if they are varied by the same amount and have similar functionality. The final Figure 5.3 presents the effect of both V_{th} variation and V_{DD} changes and their combined effect on the lock time. As observed in the graphs, coupled V_{th} and V_{DD} variation have drastic effects on the performance that may be otherwise unnoticed by performing the experiments independently. For example when combining V_{DD} =1.0V and +3σ V_{th} for Transistor 1 the circuit no longer locks as compared to nominal V_{DD} where the circuit locks almost at the ideal time. The effects of power supply and process variations are a source for serious reliability concerns and must be addressed before the device is manufactured

5.2 Techniques for Sensitivity Analysis

Over time circuit behavior is subject to change through ageing and environmental effects and sensitivity analysis is useful in locating the critical parameters and elements. Sensitivity analysis computes the effect of parameter variations on the circuit performance. The analysis provides quantitative insight into performance deviations due to process variation and their effects on design specifications.

5.2.1 Adjoint Techniques

The adjoint method [1] is a way to perform sensitivity analysis of a circuit in the transient domain. The method can be formulated as a convolution of circuit equations with a carefully constructed function. Proper initial values have to be determined by a backward integration of time in the related differential algebraic equations. Aside from transient analysis, adjoint methods are useful in determining optimal reduced order models [2][3] in the case of a large number of parameters.

Many works in this field are for linear circuits [1][4-6]. The method has been extended to incorporate nonlinear differential equations [7-9]. The works in [10-17] use adjoint linear analysis for AC, periodic AC, and AC steady state analysis for time-invariant or periodically time-varying systems.

Though the adjoint method has the ability to perform sensitivity analysis in the transient domain, it is not robust enough to incorporate a very large number of parameters. When dealing with sensitivity of the circuit with process variation, the resulting differential equations become unrealistic to be solved efficiently.

5.2.2 Symbolic Techniques

The main purpose of symbolic simulators is to replace tasks that require repeated computations such as Monte Carlo simulations and design space exploration [18][19]. These techniques are applied in early design stages where fast analyses for quick re-designs are required. This stage often relies on designer’s knowledge, and is mainly done by hand. With the advancements of computers and their processing power on one hand and increased circuit complexity on the other hand, simulation-based techniques such as Monte Carlo-based methods are becoming more popular. Nonetheless, symbolic simulators are very fast and provide an early analysis of circuit sensitivities.

With the introduction of Binary Decision Diagram-based computation methods, symbolic simulators had been extended to handle opamp circuits containing over 20 transistors. If hierarchical methods are employed [20] more than twice as many transistors can be analyzed. Symbolic sensitivity analysis techniques are applicable to circuits on a scale of tens of transistors. Large analog blocks, with more than 20 transistors, need to be approached with either approximation [21] or hierarchical methods [22].

Hierarchical methods [20][22-28] derive symbolic formulas for a sequence of expressions or nested symbolic expressions from the decomposition of circuit transfer functions. In topological hierarchical analysis [24] the circuit topology is represented as a directed graph whose edges and weights of the edges are circuit parameters. Decomposition is applied to the graph and analysis is performed to find disjoint paths and loops. In network formulation [22] decomposition is applied directly to the transfer functions and variables are eliminated one at a time for each sub circuit.

Approximation methods [21][29-35] discard insignificant terms based on the relative numerical magnitudes of the symbolic parameters and the frequency defined at some nominal design point or range. These methods suffer with respect to accuracy, but make up for it with the reduced expression, which increases speed. Unfortunately the simplified expressions often lose certain information for sensitivities with respect to parasitic or process variation [26].

Determinant decision diagrams (DDDs) [36][37] exploit the sparsity and sharing of sub-expressions within the circuit matrix and create graph representations of the symbolic determinants, under the assumption that each element in the matrix is unique. In the worst case DDDs grow exponentially with the size of the circuit, but in practice show orders-of-magnitude reduction in the number of elements. Element-Coefficient Diagrams [38][39] are an extension of determinant decision diagrams to include multiple roots.

The Method of Moments (MoM) was first introduced in [40][41] and was used to solve the integral equations of the sensitivities of electrostatic problems for planar structures. The works in [42][43] introduced adjoint techniques with MoM for full-wave sensitivity analysis. Symbolic analysis MoM-based sensitivity scales exponentially with the circuit size.

Sensitivity analysis has also been used in genetic circuits [44]. The work reported in [45] uses Random Sampling – High Dimensional Model Representation (RS-HDRM) algorithm which can provide reliable pre-experimental estimates on sensitivities of the circuit properties with respect to broad scale variations in the model parameters without knowing their precise values. The global sensitivity analysis technique can decompose the high-dimensional, nonlinear contributions of reaction rate constants to the network properties (represented by their total sensitivity) into a hierarchy of low-dimensional terms. Genetic circuit components are built from well-studied natural networks, therefore the ordinary differential equations, initial conditions, and tolerance ranges are all usually known beforehand.

All of these symbolic techniques require that an efficiently solvable function can be derived. The main difficulty with symbolic analysis is that the number of product terms in an expression may increase exponentially with the size of the circuit. For a circuit with 15 nodes and 25 devices (transistors, resistors, etc.) the determinant of the circuit matrix contains more than 10^{11} product terms [46]. The above techniques have attempted to reduce the terms as much as possible, but even with the advancements in the field most state-of-the-art techniques can handle "large" analog circuits ranging from 20-40 transistors or are tailored specifically to a circuit type and not general enough to be used on other circuits. Many state-of-the-art works using symbolic analysis are limited to linear analysis of the circuit [47-50]. In contrast the sizes of circuits in this thesis range from tens to thousands of transistors. The existing methods are impractical for circuits of such size. To the best of our knowledge there are no symbolic sensitivity analysis techniques able to handle circuits as large as a PLL.

Another shortcoming of these methods is their inability to do analysis in the transient domain. Transient analysis provides the picture of the system as a whole and how its varied behavior over time may affect neighboring digital components, i.e. lock time of a PLL.

5.2.3 Genetic Algorithms

Genetic algorithms are very fast and very powerful global optimization techniques. Their goal is to find an optimum solution based on a set of objectives and parameters. For sensitivity analysis a genetic algorithm [51-54] is combined with some kind of numerical circuit simulator such as SPICE. The data from the simulator is analyzed and sensitivity is computed by the algorithm, which then provides a new set of optimized parameters for simulations. The loop and simulations continue until an optimized solution for the objectives is found.

5.3 Techniques for Yield and Reliability

5.3.1 Reliability

Traditionally, reliability testing of a device is performed through stress testing at the device level. Designers are forced to use large design margins since the effect of device failures at the circuit level are not considered, limiting performance [55]. Given a specific process technology, certain analog components can admit 10% parameter drift, while others as little as .1%. The Vth shift caused by process variation and reliability concerns has dramatic effects on the performance of analog circuits.

There are four classes of reliability concerns; spatial stochastic unreliability effects, temporal deterministic unreliability effects, temporal stochastic unreliability effects, and dynamic unreliability effects [56]. Spatial stochastic reliability effects affect the yield of a circuit right after manufacturing. These effects include parametric process variation and time dependent wear out effects. Temporal deterministic reliability effects such as negative bias temperature instability and hot carrier degradation cause a shift in transistor parameters over time. These effects can be prevented by reducing the power supply [57], but the supply voltages can no longer be scaled at the same rate as in previous transistor generations due to the non-scalability of the sub-threshold voltage [58-60] Therefore hot carrier injection is a major concern whose effects increase the number of interface and oxide traps resulting in shifts in threshold voltage, V_{th}, carrier mobility, β, and output conductance, g_o. Bias temperature instability is a shift in Vth after a bias voltage has been applied to the gate at elevated temperatures [61]. Temporal stochastic unreliability effects also cause a shift in transistor parameters, but can also result in circuit failure from time dependent transistor mismatch. Dynamic reliability effects are caused primarily by the environment in which the circuit is in [62].

5.3.2 Yield Optimization

The goal of yield optimization of to find the design point of a circuit such that the maximum yield is achieved taking into account the manufacturing and environmental variations [63]. In contrast, the goal of design optimization is to determine design parameters such that the circuit performance is as close to the nominal as possible, taking into account process variation. In both cases the circuit needs to be analyzed for its sensitivities with respect to various parameters. This section will focus only on the analysis methods used in yield analysis and not on the optimization engines.

The typical flow for circuit yield optimization analyses the effect of parameter values on a circuit and the optimization engine uses the analysis to produce a new set of parameter values. This cycle continues until an optimized set of parameters is found. There are four types of analysis methods that are used in yield optimization: corner methods [64][65], performance-based worst case methods [66][67][68][77], response surface methods [69][70], and Monte Carlo methods [63][71-75]. A corner based method uses the fast/slow parameter sets of a device model as the worst-case parameters for all devices in a given circuit. The number of simulations is extremely low and they give a ballpark of the worst-case performances, but they are pessimistic and lead to potential overdesign. They do not consider every possible performance parameter that reduces the yield. The performance specific worst-case methods extract the worst-case parameters for each specific performance of the circuit in its nominal state, though the search for this state is difficult. Response surface methods create macro models based on regression to estimate the yield based on design variables and process parameters. These models benefit from the low computational cost making exploration very fast, but they must trade-off between accuracy and complexity of the model and require a large number of samples for higher accuracy. Monte Carlo is the most commonly used and reliable technique due to its generality and high accuracy, but cannot be used within an iterative optimization loop due to the number of simulations required.

Corner case and worst-case performance methods are generally pessimistic and do not test the full range of the circuit which reduces the yield. The method in [67] linearizes the performance at the worst-case point even though the search for this point uses nonlinear optimization, which introduces errors. [68] Builds a response surface model relating the performance to intra-die parameters that are correlated to the design parameters. The method in [77] utilizes the Box-Behnken Design which is an independent quadratic as it does not contain an embedded factorial or fractional factorial [78]. Each factor has three levels that form a box around the design space and samples are taken on the midpoint on each edge of the box.

Response Surface techniques model the circuit performances as a function of the parameters around the nominal design point. This model is then used to estimate the yield and provides insight into the design, which potentially may produce a more efficient solution. [69] Uses genetic algorithms to mine data from simulations to find the best-fit model automatically without an a-priori template. Performance based macro-modeling technique [70] employs a quasi-random sampling scheme using Halton Sequence Generation [81] that uniformly samples the design space. With any response surface modeling or macro-modeling technique, the accuracy of the models is largely dependent on the sampling of the space and dimensionality of the problem.

The Monte Carlo algorithm takes random combinations of values chosen from within a specified tolerance range of each parameter. For large number of parameters and dimensions, the number of simulations could be very high if the application requires some kind of space exploration. Therefore, various sampling methods have been introduced in order to reduce the number of simulations while still maintaining robustness of Monte Carlo.

Latin hypercube sampling [63][72][73][76] is the most common method to reduce the number of evaluations required while still enduring reasonable accuracy in computing the performance distribution function with multiple variables. It is used in conjunction with Monte Carlo to reduce the number of simulations and achieve a reasonably accurate random distribution. The key to sampling is stratification of input probability distributions, which divides the cumulative curve into equal intervals on the cumulative probability scale. A sample is randomly taken from each interval and that sample represents the interval. All of the samples are combined to recreate the probability distribution.

Quasi-Monte Carlo [74][75] requires a careful mapping of important statistical variables to the individual dimensions of the sampling process for effective use in higher dimensions. The work in [79] recommends either providing designer expertise or rank correlation coefficients for the mapping. If neither is available then the Karhunen Loéve Expansion (KLE) can be employed [80]. The KLE of a random field model of intra-die statistical variation can take an extremely large model with many random variations and reduce them to just a few uncorrelated random variables.

All of the above sampling techniques suffer from large dimensional spaces. Though there are techniques to reduce the dimensionality, they either require a knowledge-based guide or do not reduce the dimensionality enough for the scale of the circuits considered in this thesis. It was reported in [63] that while the Quasi-Monte Carlo and Latin Hypercube samplings are significant speed ups over traditional Monte Carlo, their computational load is too high for large analog circuits.

5.4 Critical Elements and Sensitivity

Sensitivity is defined as the amount the circuit behavior changes with respect to particular circuit elements. In this work the circuit behavior is the output waveform over time with respect to changes in V_{th}. Equation 1 defines sensitivity as

$$S_x^y = \lim_{\Delta x \to 0} \left(\frac{\frac{\Delta y}{y}}{\frac{\Delta x}{x}} \right) = \frac{x}{y} \frac{dy}{dx} \tag{5.1}$$

Where S is the sensitivity, x is the changing circuit component, and y is the circuit behavior we wish to evaluate as x changes. The equation simply evaluates the dependent variable Δy/y changes with respect to the change in the independent variable Δx/x. The limit as Δx goes to 0 evaluates the expression for small changes. As described in Chapter 4 each waveform is decomposed into events where each event is a vector of length n. Each event is based on the type of waveform detected; analog, digital, analog oscillating. Each vector is compared using the sum of squares.

A critical transistor is a highly sensitive circuit component. High sensitivity is determined by ranking all of the sensitive components by the level of sensitivity. The levels are determined by percentages of the varied component. For example, V_{th} of a transistor component is varied at ten different levels of sigma, where the lowest level corresponds to small changes and the highest level to large changes. Components that fall into the lowest ranks affect the circuit behavior with just small changes in sigma, making them the most sensitive components. In this work, we consider 20% of the ranking levels to be highly sensitive as default - the lowest two ranks will contain all the highly sensitive transistors. This value can be changed depending on how conservative the user is. The higher the percentage of the ranking, the more highly sensitive transistors there will be. The number of ranks depends on the distribution and uniform sampling of the component being varied. If the sampling steps are small there will be more ranks and vice versa.

5.5 Automatic Critical Transistor Identification

While the goal in Chapter 4 was to determine the minimum set of inputs of X (input events) and C (transistor variation) to completely verify a large analog feedback circuit, here we want to use the output clusters to determine the subset of C which has the lowest tolerance to variation and has the largest impact on the output behavior. Figure 5.4 represents the behavioral output space where X is a held constant and C is varying. The star symbol represents the nominal output behaviors and the clusters represent similar behaviors. The cause of each of the behaviors within the clusters that do not contain the nominal output is analyzed to determine the location of the varied transistor and its tolerance ranges.

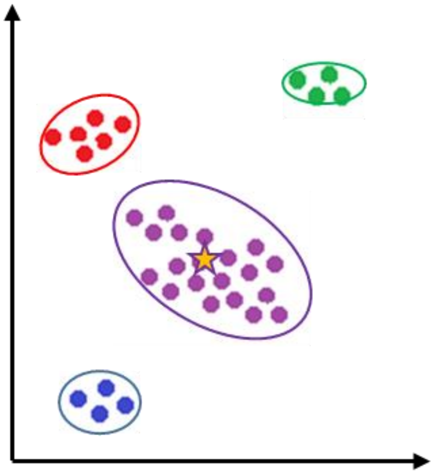


Figure 5.4: Clustering of the behavioral output space. Nominal behavior is the star symbol and each cluster represents a group of similar output behaviors.

The circuit is first partitioned into primitive elements (PEs) using CCG. Stage one is to remove non-critical nodes at the PE level. An input event is applied to the PE and each transistor is varied and simulated independently by the maximum, i.e. σ=6. If the similarity between the non-varied output event and the varied output event is >99% then the transistor is considered non-critical. A large portion of the transistors are pruned from the critical set with only performing simulation on a PE. These non-critical transistors can be removed since the variation did not affect the output of the PE it will not affect the output of the whole circuit.

Performing the pruning of transistors at the PE level in Stage 1 drastically reduces the space C. Stage 2 is to locate transistors which affect the final output behavior and not just the PE output behavior. This will provide us with two things (1) how the transistor variation behavior affects the entire circuit, and (2) if the transistor variation is masked or corrected by the normal operation of the circuit. The transistors that fall into category (2) are pruned early in the parameter sweeps. Instead of full circuit simulation, the simulation reduction method in Chapter 3 is used. All PEs classified as non-critical are replaced with behavioral models for faster simulation while critical PEs are simulated. Instead of performing Monte Carlo in C, sweeps are performed on a single transistor at a time. The transistors not pruned from Stage 1 are swept from the next largest variation, i.e. σ=5, until the circuit output behavior is clustered in the same cluster as the nominal output response of the circuit. When transistor variation is being swept on non-critical PEs, the behavioral models are replaced with circuit simulation for more accurate responses; otherwise they are replaced with behavior models.

It is true that transistors can interact with one another to form new behaviors and together become critical. It would require an extensive amount of simulations and time in order to locate these combinations, especially in a very large circuit. The goal of this chapter is to quickly locate the most critical transistors within the design in order to help facilitate and guide layout. Therefore we are only looking for the independently critical transistors. Each simulation will have independent analysis for each transistor, they can be run in parallel reducing the amount of time to find the transistors.

The following example is performed on the circuit in Figure 1.1. The transistor set contains all of the transistors in the design and partition analysis at 6σ is performed on each transistor. At 6σ all of the transistors are very sensitive. To find the tolerance range for each sensitive transistor we perform a greedy search which splits the max σ range in half at the specified uniform intervals. If the max range is 6σ, then the transistor is simulated next at 3σ. Partitioning with behavioral modeling and simulation is performed at 5σ thru .1σ and the results are displayed in Table 1 where S stands for sensitive, NS represents non-sensitive circuit responses and NT represents variation levels not tested. For Transistor 7 the variation simulations are ±6σ partition simulation and {±3σ, ±1σ, ±.1σ, +.5σ} circuit simulations. The total number of transistors for each rank is shown in the bar graph Figure 5.5. The highly sensitive critical transistors which are within Rank 1 and 2 (±.1σ and ±.5σ) are displayed in Figure 5.6(a).

Table 5.1: Sensitivity for each transistor with specified variation. S=Sensitive, NS=Non-Sensitive, NT=Not Tested

Transistor	-5 σ	-4σ	-3σ	-2σ	-1σ	-.5σ	-.1σ	+.1σ	+.5σ	+1σ	+2σ	+3σ	+4σ	+5σ
1	S	NT	S	NT	S	S	S	S	S	S	NT	S	NT	S
2	S	NT	S	NT	S	S	S	NS	NS	S	NT	S	NT	S
3	NS	NT	NS	NT	NS	NT	NS	NS	NT	NS	NT	NS	NT	NS
4	S	S	NS	NT	NS	NT	NS	NS	NS	S	NT	S	NT	S
5	S	NT	S	NT	S	NS	NS	NS	NS	S	NT	S	NT	S
6	S	NT	S	NT	S	NS	NS	NS	NS	S	NT	S	NT	S
7	S	NT	S	NT	S	NT	S	NS	NS	S	NT	S	NT	S
8	S	NT	S	NT	S	NT	S	S	S	S	NT	S	NT	S
9	S	NT	S	NT	S	NS	NS	NS	NS	S	NT	S	NT	S
10	S	NT	S	NT	S	NS	NS	NS	NS	S	NT	S	NT	S
11	S	NT	S	NT	S	NS	NS	NS	NS	NS	S	S	NT	S
12	NS	NT	NS	NT	NS	NT	NS	NS	NT	NS	NT	NS	NT	NS

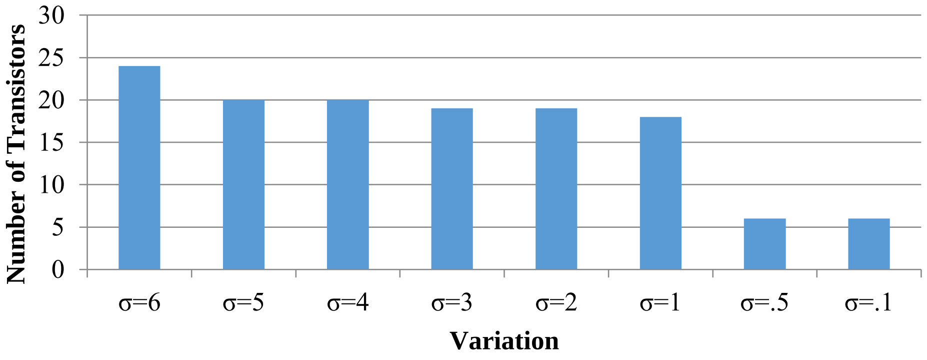


Figure 5.5: Bar graph of sensitive variation distribution of transistors

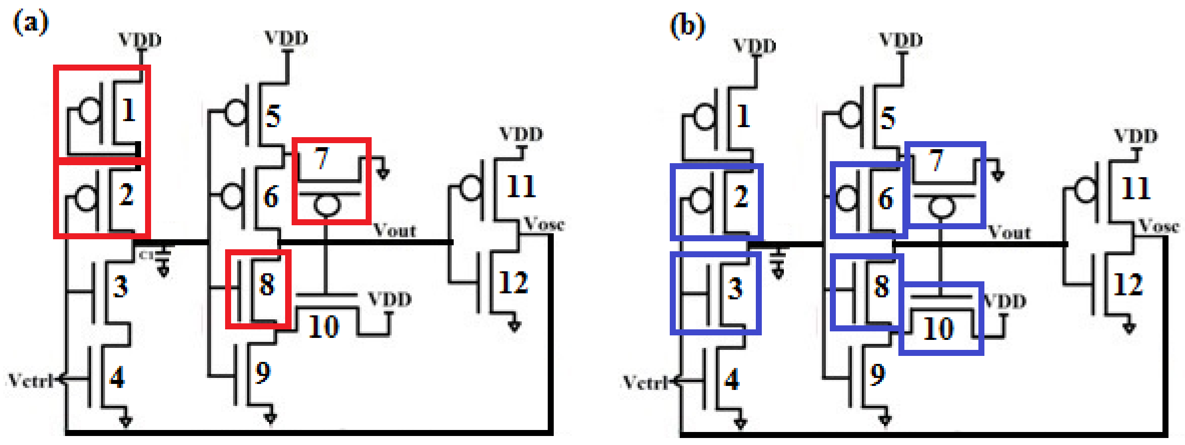


Figure 5.6: Critical transistor comparison between (a) Rank 1 with (b) expert design analysis

Figure 5.6 displays the critical transistor locations found through the ranking systems. Rank 1 and 2 critical transistors are encapsulated in a red box in (a) while the designer nodes locations (b) are within a blue box. The designer located the critical nodes which reside on nets V_{in}, V_{out} and V_{ctrl}, therefore the critical transistors are those which directly feed those nodes. V_{ctrl} is held constant through these simulations so we have removed V_{ctrl} from the critical node list. Transistors 6 and 8 are within Rank 3 at σ=1 which is close to being critical. Transistor 3 on the other hand is not critical and hardly sensitive during full circuit simulation while Transistor 1 which was not located by the designer is highly sensitive. We were able to prune one transistor from the critical list while adding another.

5.6 Experiments

The experiments were run on the circuits with variations described in Chapter 2. Sensitivities that are less the 99% are added to each rank. In this work extremely small perturbation in the waveforms are not considered sensitive enough for analysis. For a transistor to be critical it needs to be in ranks 1 or 2 and be at least 95% sensitive for a given power level.

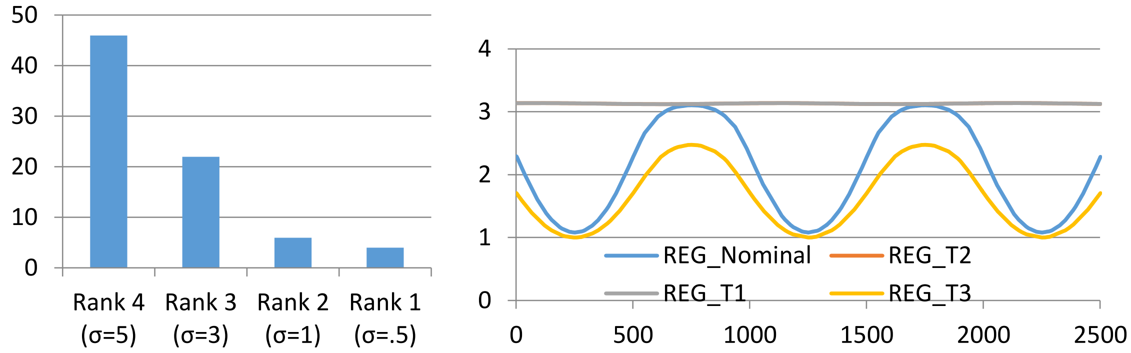
95% is the sensitivity measure for the output behavior clustering which can be changed based on the user need. The power is varied in a range from ±20% of the nominal VDD. The range is uniformly sampled and simulated in combination with V_{th} variation. Each circuit will be analyzed by the number of sensitive transistors per rank and the highest ranked transistors are compared against an expert analog designers hand done critical node predictions.

5.6.1 Cascode Regulator

The distribution of sensitive transistors for the regulator is shown in Figure 5.7(a). The number of sensitive transistors is slightly deceiving because most the sensitivities fall within 97-99% range. Of the highest sensitivity transistors, 2 of the 3 are highly critical, resulting in catastrophic failures for low levels of variation, Figure 5.7(b). The third transistor shows decreased amplitude with respect to the nominal waveform, but still results in an oscillation. Each transistor location is marked in Figure 5.8(a), where the red boxes are the two highly sensitive transistors in Rank 1 and the green box is the Rank 2 transistor. As compared with the designers critical node analysis in Figure X(b) where nodes 8 and 9 are the most critical, then 1-5, and finally 6 and 7. In comparison we detect that the most critical nodes in the design with be nodes 8 and 9 which are labeled red and 5 which is labeled green. The remaining nodes are in Ranks 3 and 5. The most critical transistors are within the biasing portion of the circuit.

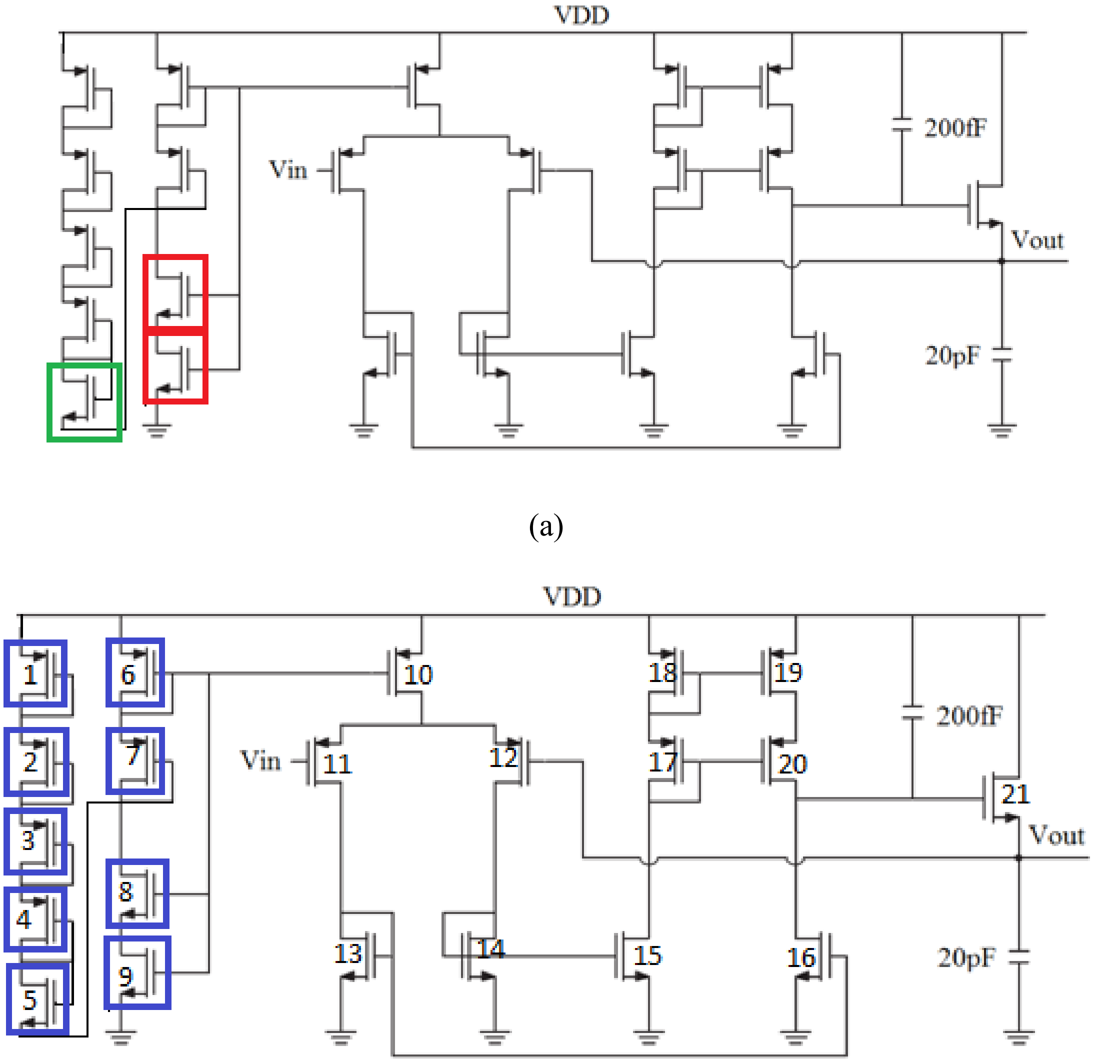
The distribution in Figure 5.9(a) depicts the total number of sensitive transistors for each σ variation level at different power levels. While there may appear to be a large number of sensitive transistors in σ=5, this is slightly misleading since the majority of those sensitivities fall into the high end of the 95-99% category. Aside from two critical transistors and one mildly sensitive transistor, all three sensitive in both the ±σ direction, any sensitive transistor within the 95-99% range typically is much closer to the high end of the distribution, Figure 5.9(b).

Aside from the two critical transistors this is a highly stable design. Figure 5.10 shows the distribution of the number of sensitive transistors to the respective sensitivity for VDD and σ variation. Figure 5.10(c) shows that at high levels of variation for σ and VDD the sensitivity rarely falls below 90% threshold.



(a) (b)

Figure 5.7: (a) Bar graph for the number of sensitive transistors per rank, (b) waveforms generated from three transistors: T1 and T2 from Rank 1 and T3 from Rank 2



(a)

(b)

Figure 5.8: (a) Circuit diagram with sensitive transistors enclosed; red=rank 1, green=rank 2

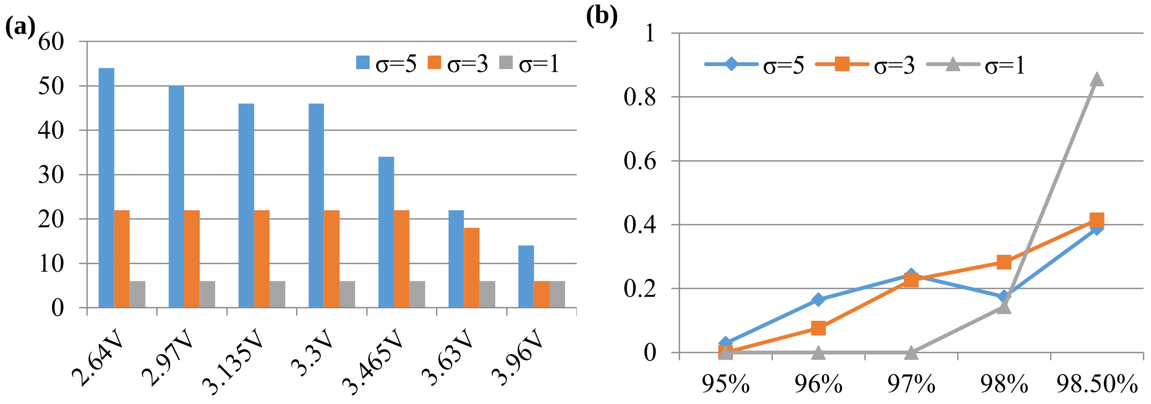


Figure 5.9: (a) Bar graph depicting the total number of sensitive transistors for each σ and VDD for the Cascode Regulator (b) The distribution of sensitive transistors within the 95-99% bracket vs. the ratio between the number of transistors within each sensitivity to all the sensitive transistors

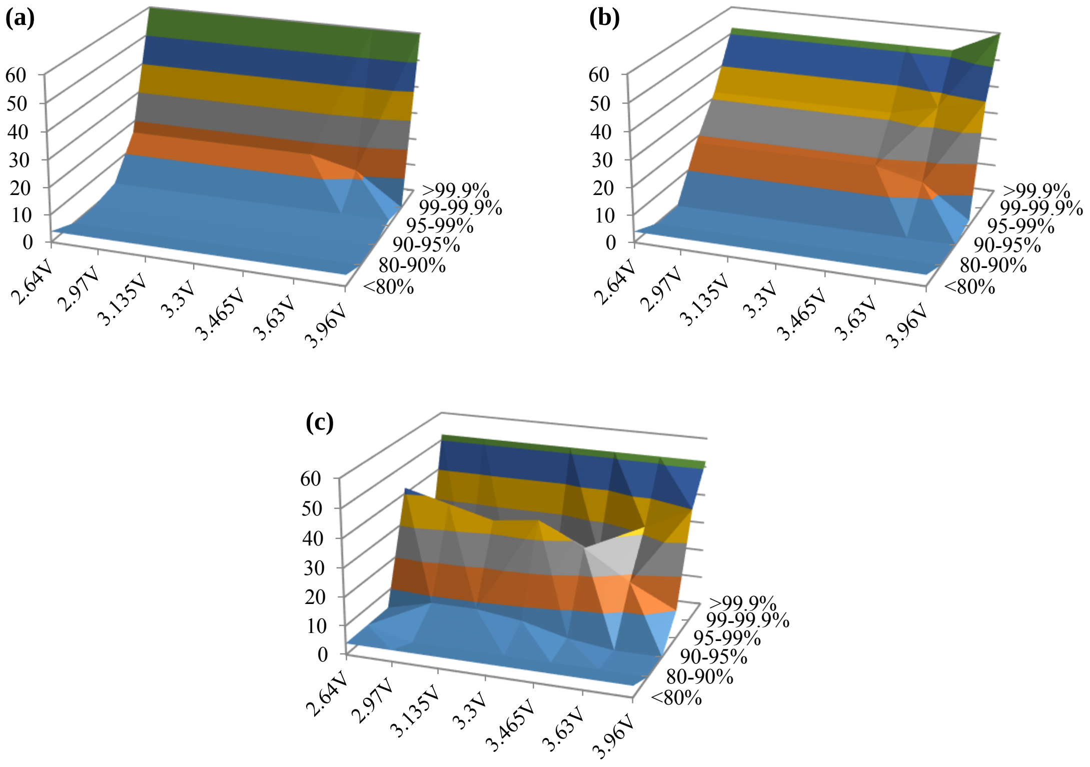
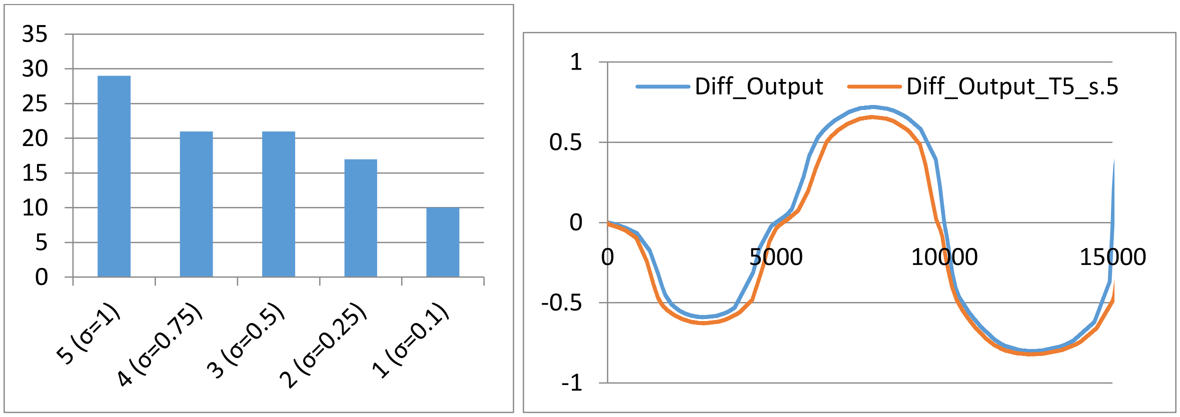


Figure 5.10: Regulator relationship between Voltage (x-axis), Number of Transistors (Y-axis), and Sensitivity Value (Z-axis), for (a) $\sigma=1$ (b) $\sigma=3$ (c) $\sigma=5$

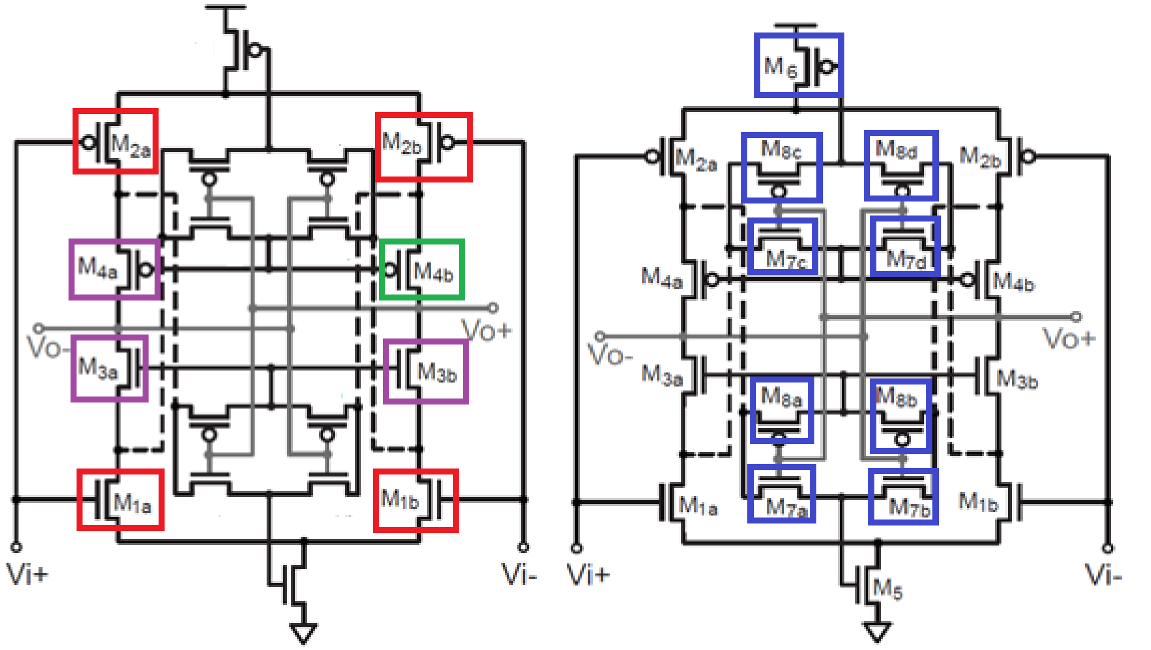
5.6.2 Differential Amplifier

The differential amplifier design is a fairly sensitive design with the rankings spanning from .1σ to 1σ, Figure 5.11(a). As differential pairs require identical matching, injecting process variation into only a single element of those pairs causes a deviation in the output, Figure 5.11(b). Though most of the transistors are sensitive the most sensitive critical transistors are those within Ranks1-3, Figure 5.12(a). As compared with the designers analysis, Figure 5.12(b) we can see that the predictions are completely opposite. The bias points M8 and M7 are contained in rank 5. The transistors identified in Figure 5.12(a) are identified as critical due to the mismatch occurring between the inputs. The circuit connecting to the amplifier will dictate how much noise can be tolerated and which ranks need to be re-sized, re-designed, or require specific layout techniques.



(a) (b)

Figure 5.11: (a) Bar graph for the number of sensitive transistors per rank (b) Difference between the output for nominal simulations and Rank 3 variation for Transistor 5



(a) (b)

Figure 5.12: (a) Circuit diagram with sensitive transistors enclosed; red=rank 1, green=rank 2, purple=rank 3 (b) Circled critical nodes from designer

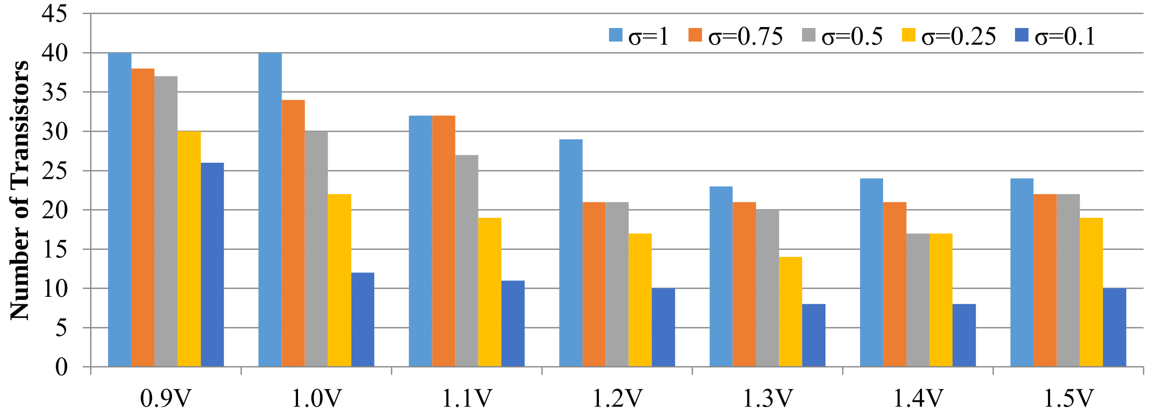


Figure 5.13: Total number of sensitive transistors for each σ and VDD for Differential Amplifier

This highly sensitive circuit exhibits increasingly more sensitive behavior on the tail ends of VDD. In Figure 5.13 the bar graph depicts extremely high sensitivity for all levels of variation at the lower end of VDD. As the voltage increases above the nominal, more transistors become sensitive. This is because of the differential self-biasing of the circuit. Vth effects the mismatch while VDD effects the current amplifying the effects of the mismatch. Figure 5.14(a-e) show the effects of VDD variation on the sensitive transistors with respect to the sensitivity level.

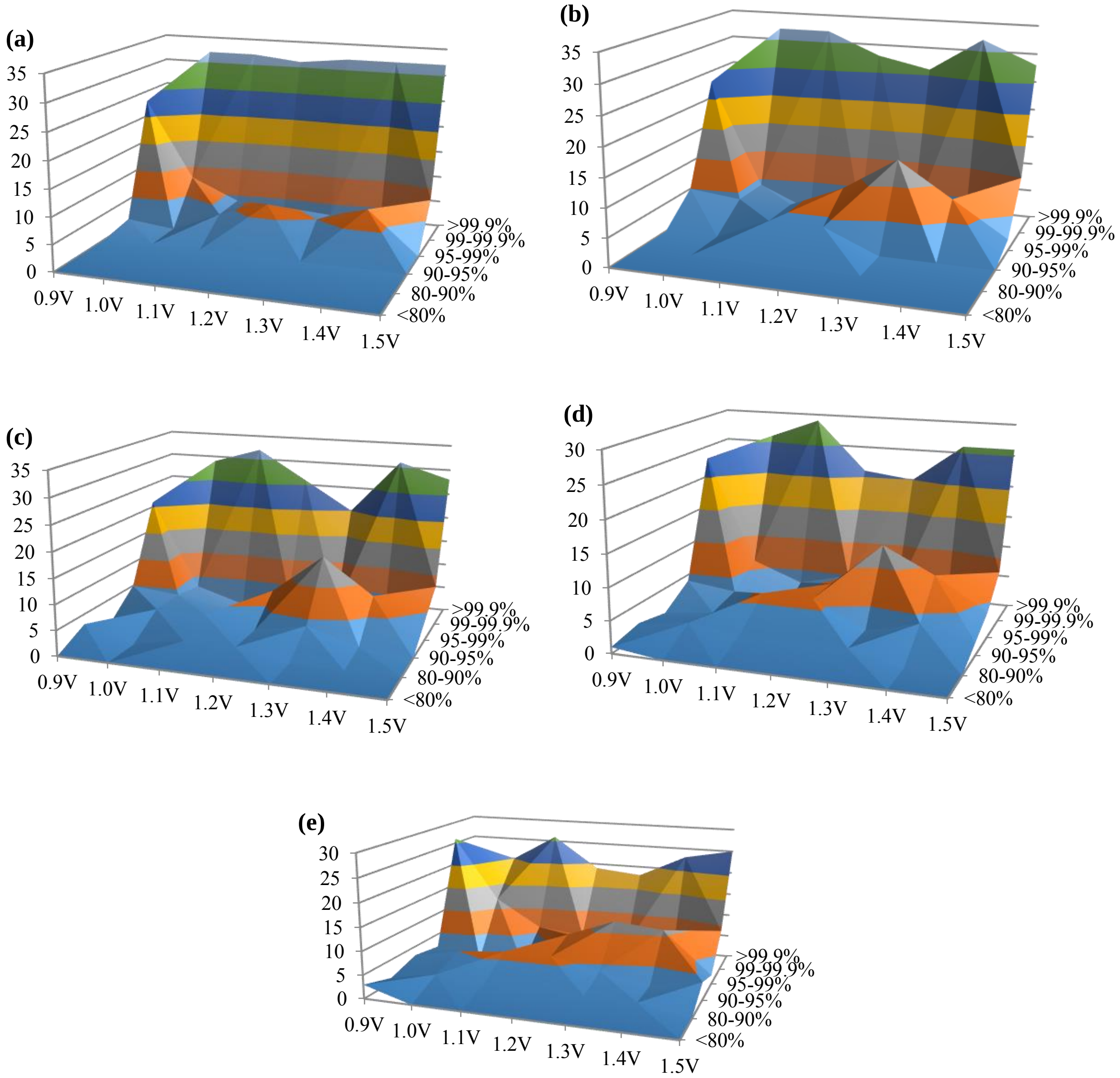


Figure 5.14: Differential amplifier relationship between Voltage (x-axis), Number of Transistors (Y-axis), and Sensitivity Value (Z-axis), for (a) σ=.1 (b) σ=.25 (c) σ=.5 (d) σ=.75 (e) σ=1

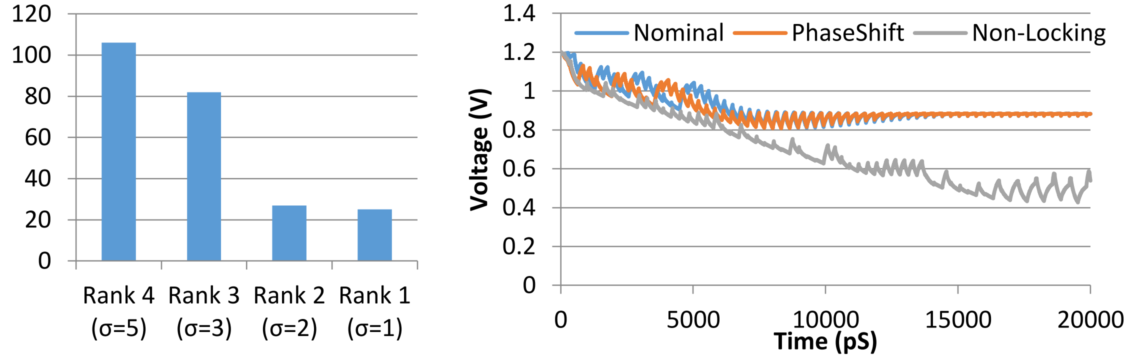
5.6.3 UWB-PLL

Due to the size and complexity of this circuit, for display purposes, only the transistors which cause catastrophic failures or unreliable waveforms will be discussed. The sensitivity level for the discussed transistors will be when the sensitivity<95%. In this section we will consider only failure to lock as the failure conditions. Due to the self correcting nature of this circuit, small variations at the partition level are corrected for within the next few cycles. Figure 5.15(b) shows the V_{cp} behavior for nominal, a shifted but locking, and a non-locking behavior. While the sensitivity of the phase shifted behavior is <95%, it still locks at the same time as the nominal behavior.

Figure 5.15 shows the number of transistors which cause failing or unreliable behaviors in this circuit. Out of the 1000+ transistors in the design only 643 were sensitive during partition level 6σ simulations. This reduced the number of simulations by more than half. In a circuit of this size where the time to simulate the entire circuit is long, this saves approximately 100 hours of simulation time. Of those 643 transistors only 111 were sensitive at the full circuit level at 5σ and 84 at 3σ. Most of the variation injected into the logic partitions of the circuit was not sensitive past 3σ leaving only the analog partitions with Rank 1 and 2 variations.

The analog partition of the circuit we show in Figure 5.16 is the charge pump. The circuits in Figure 5.16(a) are the critical transistors found through the tool where Rank 1 is red and Rank 2 is green. The corresponding designer nodes and transistors Figure 5.16(b) are marked in blue where the blue circles are the predicted critical nodes and boxes are critical transistors. For the charge pump we calculate that there will be extra critical transistors in the bias portion of the circuit while fewer transistors in matched pairs will be critical. For the other sensitive analog partitions the results are similar. Some of the differential pairs which would originally be considered sensitive are not critically sensitive while more transistors in the biasing regions are sensitive.

Due to the nature of the circuit behavior under the influence of variation, which exhibits primarily large deviations from the nominal, the analysis in this section will be on the sensitivity with respect to the lock time instead of the entire transient waveform. The lock time for the nominal circuit is around 12nS. The sensitivity categories will be broken down into locking {<9nS, 10-14nS, 15-20nS, >20nS} and never locking {high, low, oscillating}. The three types of behaviors are there to distinguish between the types of non-locking behavior; Figure 5.17. High indicates the frequency is maxed due to the inability to decrease frequency while low indicates the opposite. Oscillating indicates that the circuit never stabilizes on a frequency. The distribution for the total number of sensitive transistors is shown in Figure 5.18. As VDD increases the circuit becomes less sensitive especially at higher ranges of σ. The graphs of Figure 5.19 show the distribution of the sensitivities with respect to VDD and the sensitivity categories.



(a) (b)

Figure 5.15 (a) Circuit Bar graph of the failing and unreliable waveforms (b) Graph of Vcp for nominal operating conditaions, phase shift due to variation, non-locking behavior due to process variation

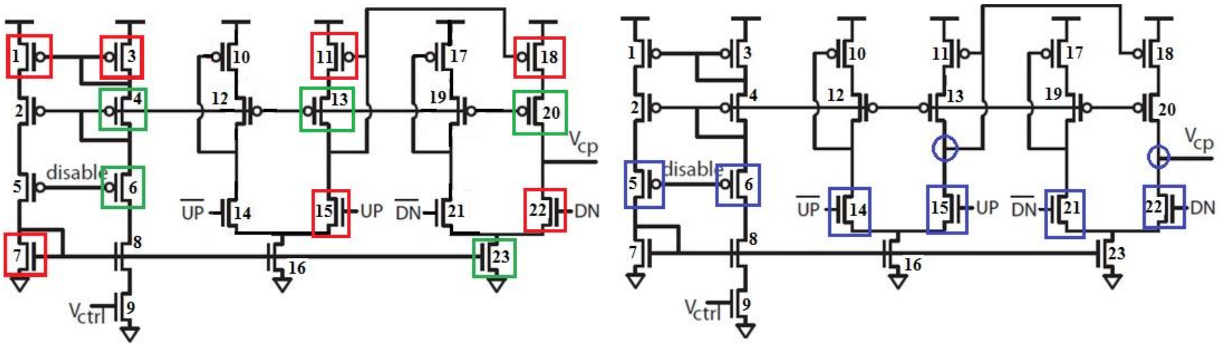


Figure 5.16: Circuit diagram with sensitive transistors enclosed; red=Rank 1, green=Rank 2, blue=designer

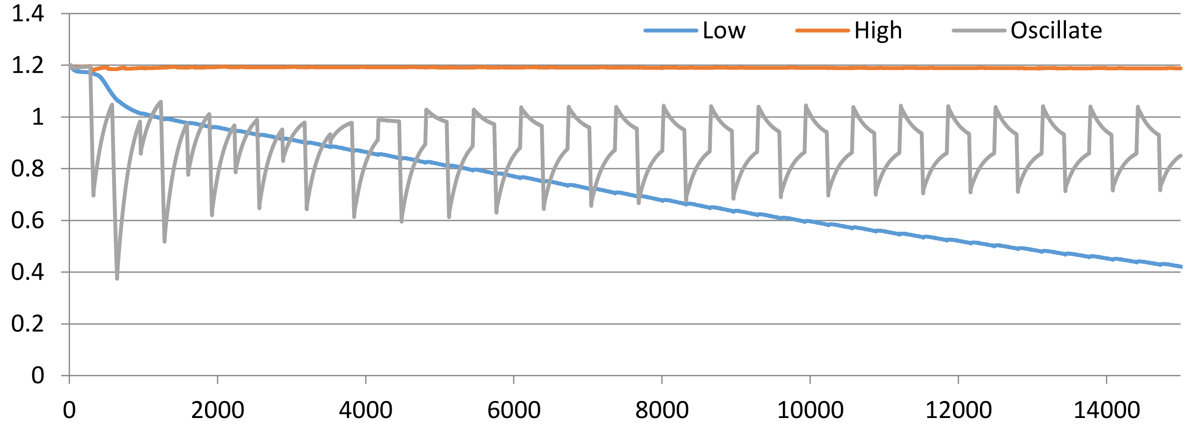


Figure 5.17: Non-Locking behavior catagorites of the UWB-PLL

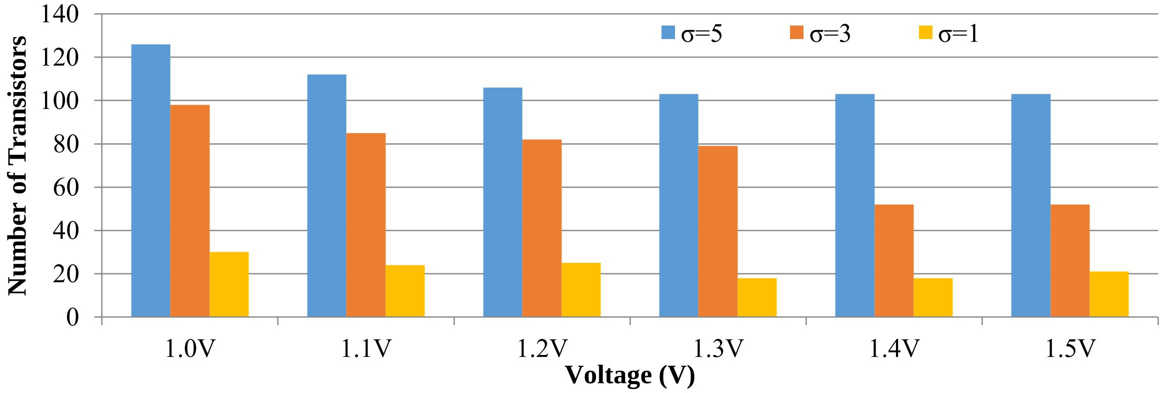


Figure 5.18: Total number of sensitive transistors for each σ and VDD for UWB-PLL

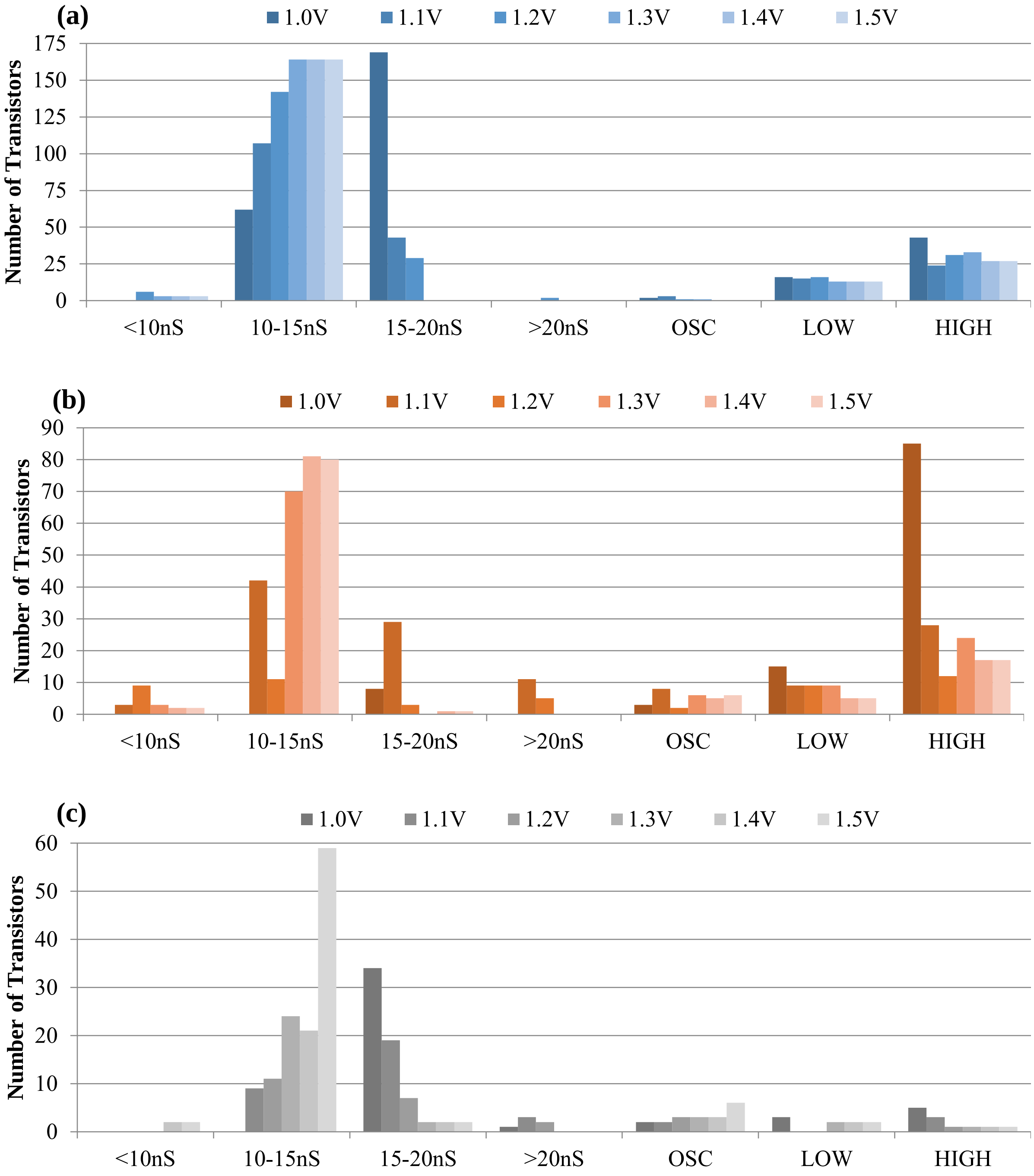


Figure 5.19: UWB-PLL relationship between number of transistors and sensitivity category for each voltage and (a) σ=5 (b) σ=3 (c) σ=1

5.6.4 TIA

The TIA circuit is not very sensitive and does not contain any critical transistors. Due to the way we compare waveforms and events, phase shifted events do not constitute a unique event or behavior. Figure 5.20 shows the (a) nominal waveform (b) 5σ transistor shifted waveforms. The behavior always oscillated at the expected frequency where the worst variation only causes a shift in the phase. This is because of the inverter chain which adjusts for variations by converting the analog signal to a discrete digital signal and always oscillates.

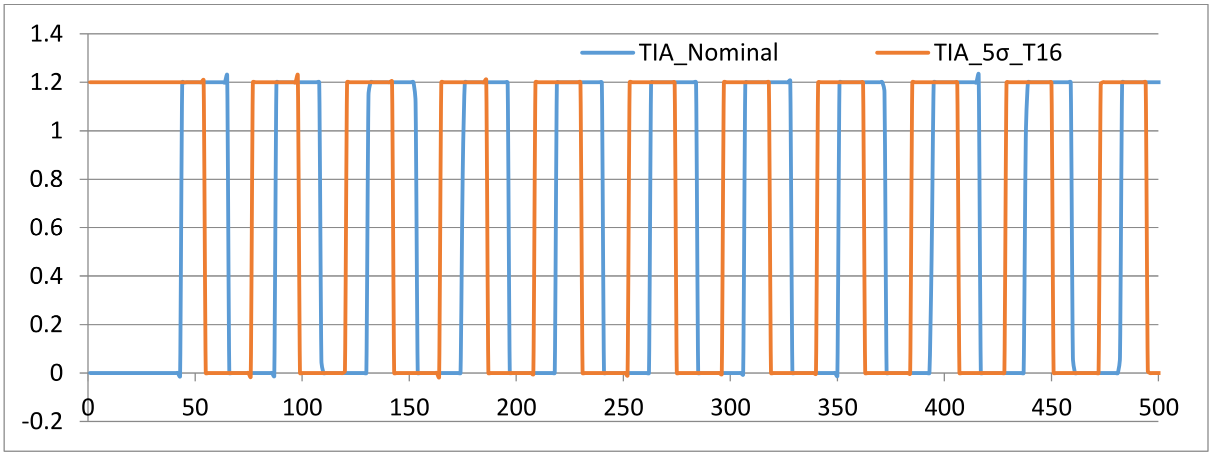


Figure 5.20: Phase shifted behavior introduced by 5σ variation on transistor 16

The circuit never fails in terms of generating a frequency, similarly to the critical transistor evaluation where the process variation causes a shift in the phase. When VDD varies the frequency is changed based on the variation level. The effect of VDD variation is shown in Figure 5.21. A change in the supply voltage directly impacts the frequency generated due to the impact on the supply current. In this circuit the combination of variation types do not produce new unexpected behaviors. For each VDD level there are variations that cause a phase shift, but the frequency generated is always the same at a given level.

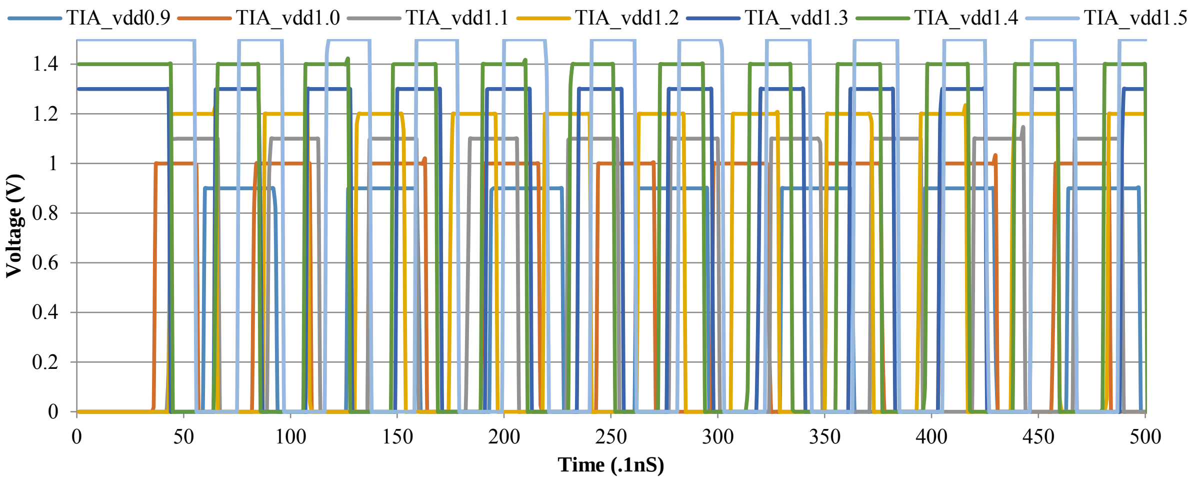


Figure 5.21: Frequency and phase sensitivity of TIA with respect to power supply variation

5.7 Conclusion

In this chapter we have shown we can automatically detect critical transistors of a design in a quick and efficient way. This chapter addresses the need for an automated tool to assist a designer in understanding and identifying critical transistors within a design. The result of the tool is compared to an expert's critical node analysis. We have shown that we can produce a set of critical nodes based on sensitivity levels to the entire circuit. Each transistor identified by the expert designer was identified by the tool. The most sensitive of the transistors were identified and the results generated by the tool which supplies the designer with feedback based on the circuits' sensitivity or overdesign.

We have been able to show that we can efficiently and automatically produce tradeoff data for environmental and process variation. The method provides a designer or layout engineer with the power and variation sensitivities for specific transistors so that they can adjust the design or layout accordingly. This method can be incorporated into a feedback optimization or simulation sizing algorithms to automatically adjust the design based on sensitivity analysis. We have shown the validity of this technique on multiple large and complex analog and mixed signal circuits.

5.8 Chapter 5 References

[1]. L.T. Pillage, R.A. Rohrer, C. Visweswariah, "Electronic circuit and system simulation methods," *McGraw-Hill, Inc, New York*, USA, ISBN 070501696, 1994.

[2] O. Balima, Y. Favennec, M. Girault, D. Petit, "Comparison between the modal identification method and the POD-Galerkin method for model reduction in nonlinear diffusion systems," *International Journal for Numerical Methods in Engineering*., Vol. 67, pp. 895-915, 2006.

[3] Y. Favennec, M. Girault, D. Petit, "The adjoint method coupled with the modal identification method for nonlinear model reduction," *Inverse Problems in Science and Engineering*., Vol. 14, No. 3, 153-170, 2006.

[4] A.R. Conn, P.K. Coulman, R.A. Haring, G.L. Morrill, C. Visweswariah, C.W.Wu, "JiffyTune: circuit optimization using time-domain sensitivities," *IEEE Trans. on CAD of ICs and Systems*, Vol. 17-12, pp. 1292-1309, 1998.

[5] Y. Fei Yuan, A. Opal, "Sensitivity analysis of periodically switched linear circuits using an adjoint network technique," *Proceedings of the 1999 IEEE International Symposium on Circuits and Systems, 1999. ISCAS '99.*, vol.5, no., pp.331-334 vol.5, 1999

[6] J.H. Laning, R.H. Battin, "An application of analog computers to the statistical analysis of time-variable networks," *IRE Transactions on Circuit Theory,* vol.2, no.1, pp.44,49, March 1955

[7] H. Xu., "Transient Sensitivity Analysis in Circuit Simulation," MSc-Thesis, Department of Mathematics and Computing Science, Eindhoven University of Technology, 2004.

[8] Z. Ilievski, H.Xu, A. Verhoeven, E.J.W. Maten, W.H. Schilders, R.M. Mattheij, "Adjoint Transient Sensitivity Analysis in Circuit Simulation," *Scientific Computing in Electrical Engineering Mathematics in Industry*, vol. 11, pp. 183-189, 2007

[9]Y. Cao, S. Li, L. Petzold, R. Serban, "Adjoint sensitivity for differential-algebraic equations: the adjoint DAE system and its numerical solution," *SIAM Journal of Scientific Computing*, Vol. 24-3, pp. 1076-1089, 2002.

[10] L. T. Pillage, A. R. Rohrer, C Visweswariah, Electronic Circuit and System Simulation Methods, *McGraw-Hill, New York*, 1995.

[11] S. W. Director and R. A. Rohrer, "The Generalized Adjoint Network and Network Sensitivities," *IEEE Trans. Ckt. Theory*, vol 16, pp. 318-323, Aug. 1969.

[12] J. W. Bandler, Qi-J Zhang, R. M. Biernacki, "A Unified Theory for Frequency Domain Simulation and Sensitivity Analysis of Linear and Nonlinear Circuits," *IEEE Trans. Microwave Theory and Tech*., vol 36. pp. 1661-1669, Dec. 1988

[13] K. S. Kundert and A. Sangiovanni-Vincentelli, "Simulation of Nonlinear Circuits in the Frequency Domain," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. CAD-5, pp. 521-535, OCt. 1986.

[14] J. Kim, J. Ren, M.A. Horowitz, "Stochastic steady-state and AC analyses of mixed-signal systems," *Design Automation Conference, 2009. DAC '09. 46th ACM/IEEE* , vol., no., pp.376,381, 26-31 July 2009

[15] Y. Fei, A. Opal, "Distortion analysis of periodically switched nonlinear circuits using time-varying Volterra series," *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications,* vol.48, no.6, pp.726,738, Jun 2001

[16] V. C. Prasad, S.N.R. Pinjala, "A fast algorithm for the generation of fault dictionary of linear analog circuits using adjoint network approach," *International Symposium on Circuits and Systems, 1990, IEEE*, vol., no., pp.37,40 vol.1, 1-3 May 1990

[17] M.E. Valtonen, "Equivalence in sensitivity calculation between direct differentiation and the method based on Tellegen's theorem," *Proceedings of the IEEE* , vol.65, no.11, pp.1602,1603, Nov. 1977

[18] F. Ferñandez, A. Rodriguez-Vazquez, J. Huertas, and G. Gielen, *Symbolic Analysis Techniques – Applications to Analog Design Automation*. NewYork: IEEE Press, 1998.

[19] P. Wambacq, G. Gielen, and W. Sansen, “Symbolic network analysis methods for practical analog integrated circuits: a survey,” *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, vol. 45, no. 10, pp. 1331–1341, 1998.

[20] H. Xu, G. Shi, and X. Li, “Hierarchical exact symbolic analysis of large analog integrated circuits by symbolic stamps,” in *Proc. Asia South-Pacific Design Automation Conference (ASPDAC)*, Yokohama, Japan,2011.

[21] Q. Yu and C. Sechen, “A unified approach to the approximate symbolic analysis of large analog integrated circuits,” *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications*, vol. 43, no. 8, pp. 656–669, 1996.

[22] M. M. Hassoun and P. M. Lin, “A hierarchical network approach to symbolic analysis of large-scale networks,” *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications*, vol. 42, no. 2, pp. 201–211, 1995.

[23] O. Guerra, E. Roca, F. V. Fern´andez, and A. Rodr´ıguez-V´azquez, “Approximate symbolic analysis of hierarchically decomposed analog circuits,” *Analog Integrated Circuits and Signal Processing*, vol. 31, pp. 131–145, 2002.

[24] J. A. Starzyk and A. Konczykowska, “Flow graph analysis of large electronic networks,” *IEEE Transactions on Circuits and Systems*, vol. CAS- 33, no. 3, pp. 302–315, 1986.

[25] M. M. Hassoun and K. McCarville, “Symbolic analysis of large-scale networks using a hierarchical signal flowgraph approach,” *J. Analog VLSI Signal Processing*, vol. 3, pp. 31–42, Jan. 1993.

[26] X. D. Tan and C.-J. R. Shi, “Hierarchical symbolic analysis of analog integrated circuits via determinant decision diagrams,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 4, pp. 401–412, April 2000.

[27] A. Doboli and R. Vemuri, “A regularity-based hierarchical symbolic analysis methods for large-scale analog networks,” *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, vol. CAS-48, no. 11, pp. 1054–1068, 2001.

[28] S. X. D. Tan, W. Guo, and Z. Qi, “Hierarchical approach to exact symbolic analysis of large analog circuits,” in *Proceedings Design Automation Conference*, 2004, pp. 860–863.

[29] S.-M. Chang, J.-F. MacKey, and G. M. Wierzba, “Matrix reduction and numerical approximation during computation techniques for symbolic analog circuit analysis,” in *Proceedings IEEE International Symposyum Circuits and Systems*, 1992, pp. 1153–1156.

[30] J.-J. Hsu and C. Sechen, “DC small signal symbolic analysis of large analog integrated circuits,” *IEEE Transactions Circuits Systems*, vol. 41, pp. 817–828, Dec. 1994.

[31] S. J. Seda, M. G. R. Degrauwe, and W. Fichtner, “Lazy-expansion symbolic expression approximation in SYNAP,” in *Proceedings IEEE International Conference Computer-Aided Design (ICCAD)*, 1992, pp. 310–317.

[32] P. Wambacq, G. Gielen, and W. Sansen, “A new reliable approximation method for expanded symbolic network functions,” in *Proceedings IEEE International Symposyum Circuits and Systems*, 1996, pp. 584–587.

[33] Q. Yu and C. Sechen, “A unified approach to the approximate symbolic analysis of large analog integrated circuits,” *IEEE Transactions Circuits Systems*, vol. 43, pp. 656–669, Aug. 1996.

[34] F. V. Fernández, J. D. Martín, A. Rodríguez-Vázquez, and J. L. Huertas, “On simplification techniques for symbolic analysis of analog integrated circuits,” in *Proceedings IEEE International Symposyum Circuits and Systems*, 1992, pp. 1149–1152.

[35] G. Gielen and W. Sansen, Symbolic Analysis for Automated Design of Analog Integrated Circuits. *Norwell, MA: Kluwer Academic*, 1991.

[36] C.-J. Shi and X.-D. Tan, “Canonical symbolic analysis of large analog circuits with determinant decision diagrams,” *IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems*, vol. 19, no. 1, pp. 1–18, January 2000.

[37] S.-D. Tan and C.-J. Shi, “Efficient approximation of symbolic expressions for analog behavioral modeling and analysis,” *IEEE Transactions on Computer- Aided Design of Integrated Circuits and Systems*, vol. 23, no. 6, pp. 907– 918, June 2004.

[38] H. Yang, M. Ranjan, W. Verhaegen, M. Ding, R. Vemuri, and G. Gielen, “Efficient symbolic sensitivity analysis of analog circuits using element coefficient diagrams,” in

Proceedings Asia South-Pacific Design Automation Conference (ASPDAC), Yokohama, Japan, Jan. 2005, pp. 230–235.

[39] H. Yang, A. Agarwal, and R. Vemuri, “Fast analog circuit synthesis using multiparameter sensitivity analysis based on element-coefficient diagrams,” in *Proceedings IEEE Computer Society Annual Symposium on VLSI*, 2005.

[40]. J. Ureel and D. De Zutter, Shape sensitivities of capacitances of planar conducting surfaces using the method of moments, *IEEE Trans Microwave Theory Tech* 44 (1996), 198–207.

[41] J. Ureel and D. De Zutter, A new method for obtaining the shape sensitivities of planar microstrip structures by a full-wave analysis, *IEEE Trans Microwave Theory Tech* 44 (1996), 249–260.

[42] N.K. Georgieva, S. Glavic, M.H. Bakr, and J.W. Bandler, Feasible adjoint sensitivity technique for EM design optimization, *IEEE Trans Microwave Theory Tech* 50 (2002), 2751–2758.

[43] Yuan, T., Cheng-W. Q., Li, Le-W., Zouhda, S., Leong, Mook-S., "Sensitivity analysis of iterative adjoint technique for microstrip circuits optimization," *Microwave and Optical Technology Letters*, Vol. 49, pp. 607-609, March 2007

[44] Saltelli, A., K. Chan, and E. M. Scott. 2000. Sensitivity Analysis. *John Wiley & Sons*, New York.

[45] Feng, X. J., Hooshangi, S., Chen, D., Li, G., Weiss, R., Rabitz, H., "Optimizing genetic circuits by global sensitivity analysis," *Biophysical Journal*, vol. 87, pp. 2195-2202, Oct. 2004

[46] P. Wambacq, G. Gielen , and W. Sansen, “A new reliable approximation method for expanded symbolic network functions,” in *Proceedings IEEE International Symposyum Circuits and Systems*, 1996, pp. 584–587.

[47]A.C. Sanabria-Borbon, E. Tlelo-Cuautle, E., "Symbolic sensitivity analysis in the sizing of analog integrated circuits," *Electrical Engineering, Computing Science and Automatic Control (CCE), 2013 10th International Conference on* , vol., no., pp.440,444, Sept. 30 2013-Oct. 4 2013

[48] G. Shi, ”Graph-Pair Decision Diagram Construction for Topological Symbolic Circuit Analysis”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 2, pp. 275-288, Feb. 2013.

[49] M. Pierzchala and M. Fakhfakh, ”Transformation of LC-filters to active RC-circuits via the two-graph method”. *Microelectronics Journal*, vol. 42, no. 8, pp. 999-1005, Aug. 2011.

[50] S. Rodriguez-Chavez, A.A. Palma-Rodriguez, E. Tlelo-Cuautle, S. X.-D. Tan, ”Graph-based symbolic and symbolic sensitivity analysis of analog integrated circuits”, Analog/RF and Mixed-Signal Circuit Systematic Design, M. Fakhfakh, E. Tlelo-Cuautle, R. Castro-Lpez (Eds.), *Springer, Lecture Notes in Electrical Engineering*, Vol. 233, pp. 101-122, 2013.

[51] M. Fakhfakh, Y. Cooren, A. Sallem, et al., "Analog circuit design optimization through the particle swarm optimization technique," *Analog Integrated Circuits and Signal Processing*, vol. 63, no. 1, pp. 71–82, 2010.

[52] E. Tlelo-Cuautle, I. Guerra-Gomez, M.A Duarte-Villaseñor, et al., "Applications of evolutionary algorithms in the design automation of analog integrated circuits," *Journal of Applied Sciences*, vol. 10, no. 17, pp. 1859–1872, 2010.

[53] B. Liu, Y. Wang, Z.P. Yu and F.V. Fernandez, "Analog circuit optimization system based on hybrid evolutionary algorithms," *Integration - The VLSI Journal*, vol. 42, no. 2, pp. 137–148, 2009.

[54] I. Guerra-Gomez, E. Tlelo-Cuautle, L.G. de la Fraga, "Sensitivity analysis in the optimal sizing of analog ICs by evolutionary algorithms," *Evolutionary Computation (CEC), 2013 IEEE Congress on* , vol., no., pp.3161,3165, 20-23 June 2013

[55] G. Groeseneken *et al*., "Trends and perspectives for electrical characterization and reliability assessment in advanced CMOS technologies," in *ESSDERC*, 2010, pp. 64–72.

[56] E. Maricau, G. Gielen, G., "Computer-Aided Analog Circuit Design for Reliability in Nanometer CMOS," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems,* vol.1, no.1, pp.50,58, March 2011

[57] C. Hu *et al*., "Hot-electron-induced MOSFET degradation—Model, monitor and improvement," *IEEE Trans. Electron Devices*, vol. ED–32, no. 2, pp. 375–385, Feb. 1985.

[58] E. Maricau *et al*., "An analytical model for hot carrier degradation in nanoscale CMOS suitable for the simulation of degradation in analog IC applications," *Microelectron. Rel*., vol. 48, no. 8–9, pp. 1576–1580, 2008.

[59] A. Bravaix *et al*., "Hot-carrier acceleration factors for low power management in DC-AC stressed 40 nm NMOS node at high temperature," in *IRPS*, 2009, pp. 531–548.

[60] W. Wanget al., "Compact modeling and simulation of circuit reliability for 65-nm CMOS technology," *IEEE Trans .Device Mater. Rel*., vol. 7, no. 4, pp. 509–517, 2007

[61] D. K. Schroder *et al*., "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing," *J. Appl. Phys*., vol. 94, no. 1, 2003.

[62] G. Gielen et al., “Emerging yield and reliability challenges in nanometer CMOS technologies,” in *DATE*, 2008.

[63] B. Liu, F.V. Fernandez, G. Gielen, "Efficient and Accurate Statistical Analog Yield Optimization and Variation-Aware Circuit Sizing based on Computational Intelligence Techniques," *IEEE Trans on Computer- Aided Design of Integrated Circuits and Systems*, Vol. 30, no. 6, pp. 793–805, 2011.

[64] K. S. Eshbaugh, “Generation of correlated parameters for statistical circuit simulation,” *IEEE Transactions Computer-Aided Design Integr. Circuits Systems*, vol. 11, no. 10, pp. 1198–1206, Oct. 1992.

[65] M. Barros, J. Guilherme, and N. Horta, “Analog circuits optimization based on evolutionary computation techniques,” *Integr. VLSI J.*, vol. 43, no. 1, pp. 136–155, 2010.

[66] F. Schenkel, M. Pronath, S. Zizala, R. Schwencker, H. Graeb, andK. Antreich, “Mismatch analysis and direct yield optimization by spec-wise linearization and feasibility-guided search,” in *Proceedings DAC*, 2001, pp. 858–863.

[67] R. Schwencker, F. Schenkel, M. Pronath, and H. Graeb, “Analog circuit sizing using adaptive worst-case parameters sets,” in *Proceedings DATE*, 2002, pp. 581–585.

[68] M. Sengupta, S. Saxena, L. Daldoss, G. Kramer, S. Minehane, and J. Cheng, “Application-specific worst case corners using response surfaces and statistical models,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol.24, no.9, pp.1372,1380, Sept. 2005

[69] G. Gielen, T. Eeckelaert, E. Martens, and T. McConaghy, “Automated synthesis of complex analog circuits,” in *Proceedings 18th Eur. Conference Circuit Theory Design*, Aug. 2007, pp. 20–23.

[70] S. Basu, B. Kommineni, and R. Vemuri, “Variation-aware macromodeling and synthesis of analog circuits using spline center and range method and dynamically reduced design space,” in *Proceedings 22nd International Conference VLSI Design*, Jan. 2009, pp. 433–438.

[71] A. A. Mutlu, N. G. Gunther, and M. Rahman, “Concurrent optimization of process dependent variations in different circuit performance measures,” in *Proceedings International Symposium Circuits Systems*, May 2003, pp. 692–695.

[72] S. K. Tiwary, P. K. Tiwary, and R. A. Rutenbar, “Generation of yield aware Pareto surfaces for hierarchical circuit design space exploration,” in *Proceedings DAC*, 2006, pp. 31–36.

[73] M. Stein, “Large sample properties of simulations using Latin hypercube sampling,” *Technometrics*, vol. 29, no. 2, pp. 143–151, 1987.

[74] A. Singhee, S. Singhal, and R. A. Rutenbar, “Practical, fast Monte Carlo statistical static timing analysis: Why and how,” in *Proceedings IEEE ICCAD*, Nov. 2008, pp. 190–195.

[75] A. Singhee and R. Rutenbar, Novel Algorithms for Fast Statistical Analysis of Scaled Circuits. *Berlin, Germany: Springer*, 2009.

[76] K. T. Fang. K. Fang, L. Runze, "Design for Modeling for Computer Experiments," *CRC Press*, October 2005

[77] H. Cai, H. Petit, J.-F. Naviner, "A fast reliability-aware approach for analogue integrated circuits based on Pareto fronts," *New Circuits and Systems Conference (NEWCAS), 2013 IEEE 11th International* , vol., no., pp.1,4, 16-19 June 2013

[78] NIST/SEMATECH e-Handbook of Statistical Methods. [Online]. Available: http://www.itl.nist.gov/div898/handbook

[79] A. Singhee and R. A. Rutenbar, “From finance to flip-flops: a study of fast quasi-Monte Carlo methods from computational finance applied to statistical circuit analysis,” *ISQED*, 2007.

[80] A. Singhee, S. Singhal and R. A. Rutenbar, “Exploiting correlation kernels for efficient handling of intra-die spatial correlation, with application to statistical timing,” *DATE*, 2008.

[81] J.H.Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Nuremische Mathematik*, 2:84–90, 1960

Chapter 6

Conclusion

In this work we have presented a new approach for creating behavioral models of analog and mixed signal circuits based on partitioning. This methodology addresses the need for an automatic approach for behavioral modeling of any type of analog and mixed signal circuits. We developed a tool that can automatically create a set of partitions and detect intermediate behaviors based on netlist and transistor level simulation behavior. SVM models are created to predict intermediate behaviors which lead to the prediction of the final output behavior. We have shown the generality and feasibility of this approach on large circuits such as a PLL and ∑Δ-ADC. Our results indicate that we can obtain three orders of magnitude speedup over transistor level simulations while maintaining over 95% accuracy.

We then developed a methodology for applying statistical learning to the verification of analog circuits. The first methodology developed utilizes unsupervised learning, circuit partitioning, and event propagation to determine the minimal representative set of input events which describe the output space. A significant amount of simulation time is saved by only simulating the important inputs. The second methodology developed locates primitive elements with low complexity which can be modeled behaviorally instead of simulated. Unsupervised learning is used on the input and output of each cluster to determine if the primitive element increases, transfers, or decreases the information content. Distance calculations are used to determine the complexity of the events through the primitive element. Low complexity primitive elements maintain the distance ordering of the events. Behavioral models are created using supervised learning techniques for primitive elements with transferring or decreasing information content. We have shown the effectiveness of the method on four analog circuits where the simulation time is decreased by 55-75%.

Finally, we have shown that we can automatically detect critical transistors of a design in a quick and efficient way. This chapter addresses the need for an automated tool to assist a designer in understanding and identifying critical transistors within a design. The result of the tool is compared to an expert's critical node analysis. We have shown that we can produce a set of critical nodes based on sensitivity levels to the entire circuit. Each transistor identified by the expert designer was identified by the tool. The most sensitive of the transistors were identified and the results generated by the tool which supplies the designer with feedback based on the circuits' sensitivity or overdesign.

We have been able to show that we can efficiently and automatically produce tradeoff data for environmental and process variation. The method provides a designer or layout engineer with the power and variation sensitivities for specific transistors so that they can adjust the design or layout accordingly. This method can be incorporated into a feedback optimization or simulation sizing algorithms to automatically adjust the design based on sensitivity analysis. We have shown the validity of this technique on multiple large and complex analog and mixed signal circuits.

Chapter 7

Future Work

The work in Chapter 4 can be extended in multiple areas. The first of these areas is the modeling of the primitive elements. In the current form the modeling is done simply by creating lookup tables and applying linear interpolation. Time sensitive modeling of the behavior is a challenging problem which may be solved with polynomial chaotic models. In current literature, these chaotic models have been applied to simple analog circuits which may be conducive for primitive element modeling.

Event and waveforms parsing is done using simple evaluations of the waveforms. If the waveform is repeating, then each period is considered an event, otherwise the entire waveform is an event. This can be reevaluated to extract key information from each waveform by way of advanced signal analysis, both digital and analog. Instead of predicting the event, the attributes and be predicted and the waveform can be reconstructed based on the predicted attributes.

Input space clustering can be extended to include redundancy calculations for better sampling of complex regions. The redundancy measure will ensure that each cluster in the input space maps to the output space in a similar cluster.

The use of Kernel Density Estimation (KDE) can be employed to calculation information loss for information and complexity measures of primitive elements. Analyzing the soundness of the data for building behavioral models enhances the confidence in the behavioral model predictions.