A Framework for Noise Analysis and Verification of Analog Circuits

Rajeev Narayanan

A Thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of Doctor of Philosophy at

Concordia University

Montréal, Québec, Canada

March 2012

CONCORDIA UNIVERSITY

Division of Graduate Studies

This is to certify that the thesis prepared

By: **Rajeev Narayanan**

Entitled: **A Framework for Noise Analysis and Verification of Analog Circuits**

and submitted in partial fulfilment of the requirements for the degree of

Doctor of Philosophy

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the final examining committee:

_____ Dr. Sedaghati, Ramin (Chair)

_____ Dr. Massicotte, Daniel (External Examiner)

_____ Dr. Cowan, Glenn (Examiner)

_____ Dr. Dolatabadi, Ali (External to Program)

_____ Dr. Zahangir, Kabir (Examiner)

_____ Dr. Tahar, Sofiène (Supervisor)

Approved by _____

Chair of the ECE Department

_____ 2012 _____

Dean of Engineering

ABSTRACT

A Framework for Noise Analysis and Verification of Analog Circuits

Rajeev Narayanan, Ph.D.

Concordia University, 2012

Analog circuit design and verification face significant challenges due to circuit complexity and short market windows. In particular, the influence of technology parameters on circuits, noise modeling and verification still remain a priority for many applications. Noise could be due to unwanted interaction between the various circuit blocks or it could be inherited from the circuit elements. Current industrial designs rely heavily on simulation techniques, but ensuring the correctness of such designs under all circumstances usually becomes impractically expensive. In this PhD thesis, we propose a methodology for modeling and verification of analog designs in the presence of noise and process variation using run-time verification methods. Verification based on run-time techniques employs logical or statistical monitors to check if an execution (simulation) of the design model violates the design specifications (properties). In order to study the random behavior of noise, we propose an approach based on modeling the designs using stochastic differential equations (SDE) in the time domain. Then, we define assertion and statistical verification methods in a MATLAB SDE simulation framework for monitoring properties of interest in order to detect errors. In order to overcome some of the drawbacks associated with monitoring techniques, we define a pattern matching based verification method for qualitative estimation of the simulation traces. We illustrate the efficiency of the proposed methods on different benchmark circuits.

To My Parents, Wife, Children, and In-Laws

ACKNOWLEDGEMENTS

First and foremost, I would like to thank my advisor Dr. Sofiène Tahar. I feel very fortunate that he took me on when I was left without a supervisor after my first year. I am very grateful for the opportunities he had provided me, right from presenting in prestigious conferences down to coaching an intern. I appreciate all his contributions of time and funding to make my Ph.D a successful one.

I am also thankful to the past and present members of the HVG group. The group has been a good sounding board, and a source of friendships. In particular, I would like to extend a special thanks to Dr. Mohammed Zaki, a past member of the HVG group for his encouragement, camaraderie and pointers. Zaki has played a key role in making my research experience a stimulating one. Some memorable people who have been inspirations in many ways would be Billy, Sanaz Afshar Khan, Naeem Abbasi and Liya Liu.

I am thankful to the people that I have had a chance to work with outside the HVG. In particular I would like to thank Dr. Behzad Akbarpour and Prof. Lawrence Paulson (University of Cambridge) for their collaboration. It was very motivating for me to work with them. Last but not least, I would like to thank my family for their support and encouragement. For my Father, who has been steadfast in his support of my Ph.D and my wife who has stood beside me through it all.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS

ABV	Assertion Based Verification
AC	Alternating Current
ACK	Acknowledgement
AMS	Analog Mixed Signal
ASIC	Application Specific Integrated Circuits
BDD	Binary Decision Diagram
BJT	Bipolar Junction Transistor
CAD	Computer-Aided Design
CDF	Cumulative Distribution Function
CT	Continuous-Time
dB	Decibel
DAE	Differential Algebraic Equation
DC	Direct Current
DT	Discrete-Time
FSM	Finite State Machine
HDL	Hardware Description Language
HOL	Higher Order Logic
IP	Intellectual Property
KCL	Kirchoff Current Law
KVL	Kirchoff Voltage Law
LCS	Longest Common Subsequence
LCSS	Longest Closest Subsequence
LPHN	Labeled Hybrid Petri Net

MNA	Modified Nodal Analysis
MOR	Model Order Reduction
MOS	Metal Oxide Semiconductor
MS	Mixed Signal
NF	Noise Figure
ODE	Ordinary Differential Equation
PCB	Printed Circuit Board
PDF	Probability Distribution Function
PLL	Phase Locked Loop
PWSN	Poisson White Shot Noise
RF	Radio Frequency
RMS	Root-Mean Square
RTL	Register Transfer Logic
RTV	Run-Time Verification
SDE	Stochastic Differential Equation
SPICE	Simulation Program with Integrated Circuit Emphasis
SoC	System on Chip
SRE	System of Recurrence Equations
SNR	Signal-to-Noise Ratio
SSNR	Segmental Signal-to-Noise Ratio
VCO	Voltage Controlled Oscillator

Chapter 1

Introduction

1.1 Motivation

"*Design is not just what it looks like and feels like. Design is how it works.*"

Steve Jobs (1955-2011).

The above statement has many connotations in science and engineering, especially, in the area of semiconductor (chip) design. Over the last decade, high performance System-on-Chip (SoC) [78] based printed circuit board (PCB) has played a pivotal role in the growth of consumer electronics (iphones, cameras, game console, laptops, etc.), embedded systems and computing servers. A typical SoC, shown in Figure 1.1, can be characterized by its interaction between different intellectual property (IP) units, advanced microprocessor, custom built radio frequency (RF), and analog and mixed signal (AMS) circuitry. An AMS circuit combines analog and digital units on a single chip and remains the backbone to any SoC design as it represents the interface to the external world. Apart from generating system reference clock (e.g., a phase locked loop (PLL)), front and back end AMS circuits are responsible for data translation across analog/digital domains (A/D, D/A converters),

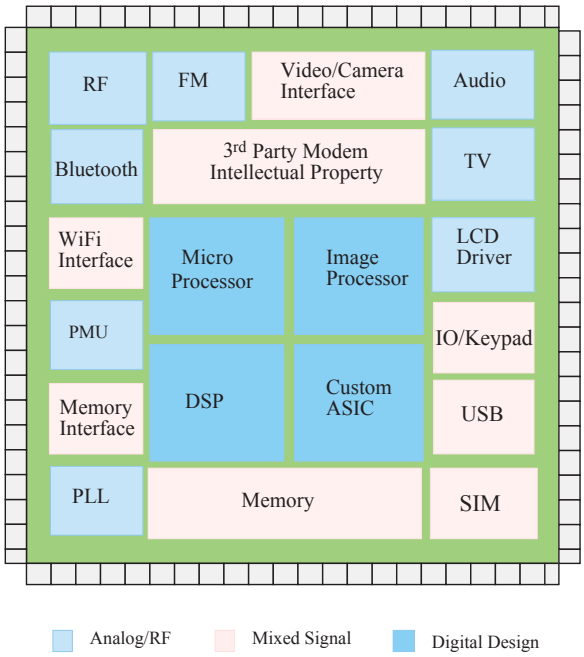


Figure 1.1: System-on-Chip [25].

and circuit biasing (e.g., a Band-Gap reference) which facilitates the correct and stable operation of the SoC. The complexity of an SoC continues to escalate against a backdrop of process scaling, tenacious competition among different vendors and aggressive time-to-market schedules. As most of the analog circuit within an AMS block is handcrafted, the fast paced product cycle has created a unique challenge to its traditional design and verification techniques.

A recent report [25], depicted in Figure 1.2, reveals that 70% of the product re-spin are due to functional errors, with industry/research groups spending more than 80% of effort on verification. Though, the combined analog/RF designs occupy less than 20-30% of the chip area, they contribute to more than 50% of the design failures [25]. This is not surprising, as in reality, analog design and verification process tend to focus on a much larger set of design specification [86] to find an optimal trade-off for better circuit performance and yield, as summarized in Figure 1.3.

Over the last decade, computer-aided design (CAD) tool development has seen a

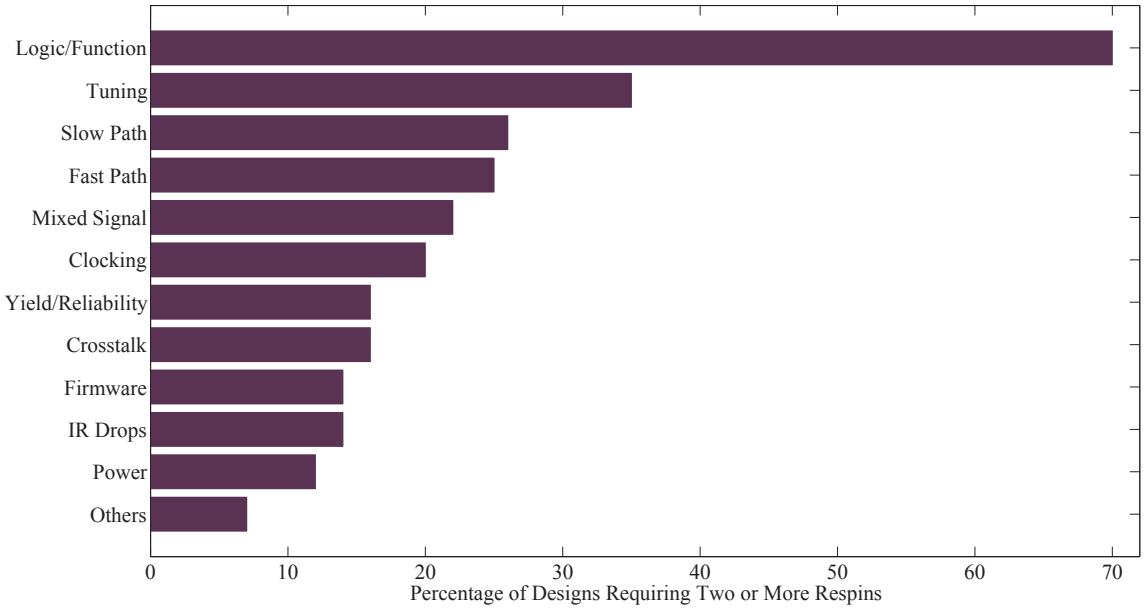


Figure 1.2: Respin in System on Chip [25].

tremendous growth in ensuring the correctness of analog designs influenced by noise, fluctuations, environment constraints, and manufacturing variations. Yet, the analog design and verification process remains a very complex and daunting process, and still lags its digital counterpart in many aspects such as, abstraction, automation, simulation run-times, and IP (Intellectual Property) reusability [79]. The analog design flow has remained essentially the same for the past twenty years [64]. This transistor-level analog design flow starts with the front-end design of the individual blocks using a schematic capture program (usually SPICE circuit simulator [15]), which are then verified through multiple simulation by combining noise and process variation details to form the overall design [64]. Even in the current state-of-the art, circuit parameters ($\frac{W}{L}$ ratio of transistors, power consumption, cutoff frequency, etc.) are determined manually or with little automation and then verified through visual inspection using simulation. Till date, performing system level verification at micromodel (transistor) level using SPICE may lead to weeks/months of labor intensive

circuit simulation to validate the design and can become impractically expensive.

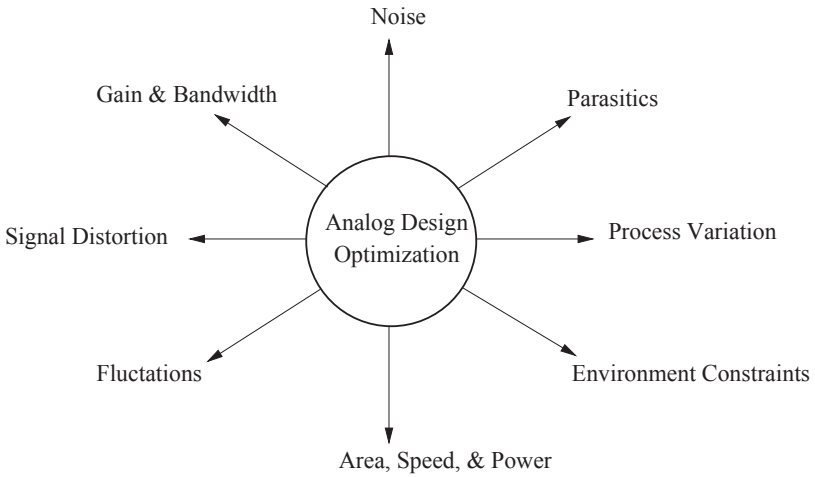


Figure 1.3: Challenges in Analog Designs [86].

To address the above challenges, industry/research groups [63] have complemented the traditional circuit level verification approach with the behavioral level modeling and verification at a higher level of abstraction. The advantage of such an approach is that the verification for the whole design can be automated and performed much faster. This speed-up, however, does not come without a price. The first cost is the accuracy of the behavioral model against the actual transistor-level designs. Secondly, the model has to account for physical (threshold voltage, leakage current, etc.), functional (noise, jitter) and environmental (temperature) constraints. Current modeling and verification methods using hardware description language (HDL) environments ([5], [121], and [58]) can be very effective to understand the functional behavior of the designs, but in reality, it does not guarantee that the design will maintain the same behavior with effects like noise, process variation, and so on. Therefore, there is a need for a new modeling and verification paradigm to complement the classical verification methods to handle noise, process variation and other constraints at a higher level of abstraction in order to avoid costly errors downstream.

1.2 Noise and Process Variation in Analog Circuits

Noise is a random phenomenon whose origin has been studied by many researchers for decades, yet it still remains a mysterious threat to any hardware systems [52]. The sources of noise could be due to unwanted interaction between different circuit blocks (e.g., cross-talk noise) or it could be inherited from the circuit elements (e.g., thermal, shot and flicker) [66]. Additionally, a large amount of simultaneously switching exhibited by digital signal can also cause noise in sensitive analog circuits and can result in unwanted oscillation or false spikes. Noise can also be transmitted around a chip through the power rails, the package and the substrate. Other types of noise common in deep sub-micron technology are the *excess* and the *telegraph* noise [81]. While the *excess* noise occurs due to the heating of charge carriers by the high lateral electric field, *telegraph* noise happens due to the trapped charges near the Fermi level [81].

Thermal noise is associated with the random thermal motion of carriers in any conducting material and in general considered to be independent of the conventional current [66]. The extent of the motion is proportional to the resistance of the material and its absolute temperature T (°K). As T approaches zero, thermal noise tends to die out. *Shot noise* is generally found in junction semiconductors, and its existence is attributed to the motion of charges across the junction between two semiconductor materials. For instance, in a semiconductor p-n junction, the movement of charge carriers into the depletion region generates a small pulse, this contributes to shot noise. In order to model shot noise, one has to understand the rate at which this pulse arrive and the corresponding amplitude. At any given time, such random pulses take Poisson distribution path with its amplitude to be fixed/varying [52]. Effective at lower frequencies, the *1/f noise*, also called *flicker noise* is due to impurities in a conductive channel , for instance, varying doping concentration in active devices [66]. Flicker noise is a general form of a power law noise or a $1/f^{\alpha}$, noise

where α is considered to vary between 0 and 2 [66]. *Cross-talk* noise, is due to capacitive and inductive coupling between the lines that run close to one another, meaning, the signal on one line will influence the behavior of the signal in the adjacent lines. This kind of interference effect depends on the frequency of the signal, the proximity of the two lines, and the total distance that the two lines run adjacent to one another [52].

Can we eliminate noise? The short answer is that it depends on the type of noise and its origin. For instance, with proper layout and shielding techniques between the analog and digital blocks or between two neighboring lines in a design, interference noise can be nullified [66]. On the other hand, the inherent (e.g., thermal, shot and flicker) noise can be reduced but cannot be eliminated completely. This is because, the dynamics of such noise are influenced by the way active/passive elements are manufactured and environment constraints that could totally alter its behavior.

With the reduction in feature dimensions to a nanotechnology process, analog designs are becoming more challenging to analyze and verify. The manufacturing steps such as *Local Oxidation, Photolithography, Ion Implantation,* and *Etching* present a completely different set of constraints on the design [124]. For instance, in a MOS transistor, *can we assume the ultra-thin oxide layer that separates the gate from the channel has a smooth surface?* Absolutely not, it is difficult to control the manufacturing process entirely [3] and hence process variation will create disparities at different points in a device [30]. The sources of variations can be classified as *interdie* or *intradie* variations [124]. While, *interdie* variation, also called *global* variation, assumes the device/circuit parameter discrepancies to be the same across *die-to-die* or *lot-to-lot* or *wafer-to-wafer* [124], *intradie*, also called *local* variation, reflects the mismatch in a component with reference to an adjacent component [124]. In this case, the devices in the same circuit might have different variations, thereby posing a serious threat on circuit performance and functionality.

Poly resistors that are built with poly layer deposited over field oxide are used widely to represent resistors in analog designs and its value depends on the sheet resistance (R_{sh}) associated with the poly layer. For a given process the variations in poly resistance are mainly due to fluctuation in film thickness, doping concentration, doping profile and annealing conditions [124]. A capacitor component in an analog circuit can be constructed at various levels (*poly-to-poly*, *metal-to-metal* and *junction*). However, a metal-oxide semiconductor (MOS) transistor based capacitor has an adverse impact on the performance of the circuit mainly due to the fact that it inherits a large deviation in the capacitance value [124]. This, in turn, alters the total input/output capacitance of the circuit, thereby, resulting in slower processing of the analog signal. For a MOS transistor, the process variation may cause a deviation in threshold voltage (V_t), length and width of the transistor (L and W), or oxide thickness (T_{ox}) , which may change, the device characteristics across the die/wafer.

1.3 Problem Statement

Analog circuits are increasingly evolving into abstract designs that rely heavily on the behavioral models and can yield simulation performance improvements that can make the full chip verification a reality. Much CAD literature were focussed on studying such possible system level modeling/verification frameworks for AMS designs. The verification approaches that have been developed (e.g., [50], [128], [79], [53], [11], [23]) in the recent years make use of mathematical models in the form of ordinary differential equations (ODE) or differential algebraic equations (DAE) to characterize the functional behavior of the designs. Unfortunately, these methods fall short in addressing the following real-world uncertainties associated with the circuit behavior due to:

- More often, analog designs act upon unpredictable environmental conditions, random noise effects, and semiconductor manufacturing disparities that may alter the functional characteristics of the circuit. Several work for modeling noise compute the power spectral density (PSD) response of a transistor-level circuit simulation to the collection of small noise sources. The noise analysis can be combined with process variation for a statistical estimation of the circuit failures. At the circuit-level, simulating and validating an analog design with noise and process variation is exceptionally costly both in terms of time and memory resource allocation. Finding a way to reduce the simulation time to verify the analog designs with noise and process conditions, while trading off some accuracy is extremely valuable in detecting circuit failures earlier in the design cycle.

- Verification of analog designs is still manual or semi-automatic. Current verification methods at a higher level of abstraction rely heavily on testbench structures that can report a design failure for a single bounded simulation trace. While this may be sufficient enough to validate the functional behavior, the unpredictable noise and process conditions require an exhaustive validation to quantify the failure in terms of circuit confidence level. To find a way that could provide a probability of failure at a higher level of abstraction can be very useful to estimate the overall design failures during the verification process.

1.4 State-of-the-Art

Verification methods for analog circuits in the presence of noise and process variation have been developed in theory and in practice primarily at the transistor level of abstraction [81]. More recently, advances have been made in the area of noise modeling and simulation using

stochastic differential equations (SDE) [107]. A SDE is an extension to ordinary differential equation (ODE) with stochastic term that could model thermal/shot noise behavior in an analog circuit. This has attracted CAD tool developers to complement the traditional small-signal noise analysis with the SDE based approach.

For instance, Synopsys has introduced a tool HSPICE RF [120] implementing SDE techniques to make a direct prediction on the statistical behavior of analog/RF circuits. The results include the usual deterministic transient analysis waveforms and the time-varying root-mean square (RMS) noise behavior. Likewise, an open source tool, *f*REEDA [47] provides a leverage to model and analyze noise using SDEs. Based on *f*REEDA, the authors in [74] have performed an SDE based phase noise simulation in the time domain. Interesting contribution through a time-domain numerical integration methods for behavioral noise analysis have been reported in [97], [40], and [122]. These methods evaluate an electronic oscillator with new physical descriptions of thermal noise by combining the non-equilibrium statistical mechanics with the SDE based Langevin approach [107].

Other specific attempts to bring the simulation of AMS circuit closer to logic simulation take advantage of analyzing noise at a higher level of abstraction using HDLs ([76], [79], [80] and [81]) or MATLAB/Simulink framework with the focus on measuring the frequency response of the circuit in terms of signal-to-noise ration (SNR) and noise figure (NF). SNR is a measure used to determine the quality of a signal that is corrupted by noise, and NF is a quality measure of SNR degradation [81]. For instance, the authors in [112] have developed behavioral models for a sigma-delta modulator that takes into account sampling jitter, and *kT/C* noise. In the end, the measured frequency response is compared with the specification for design validation.

For process variation, designers use a combination of *Worst-Case*, *MonteCarlo* or

Mismatch [124] analysis to verify analog circuits. The worst-case analysis method for analog circuits incorporates design models with pessimistic process corners. These worst-case variations are determined in the foundry design document, and their values are derived from certain parameter distribution. For instance, the process corners are constructed to maximize/minimize one specific performance of the device (e.g., speed, power, area, etc.) and can provide faster results [124]. However, the worst-case analysis that targets single device variation may increase the overall design effort and cost. The MonteCarlo method takes into account a predefined distribution (usually normal distribution) of the device parameters due to process variation. When defining normal distribution, the designers have to use certain standard deviation, usually, it is $\pm 3\sigma$ parameter distribution. Unlike worst-case that targets for single device performances, MonteCarlo methods use a repeated simulation technique for multiple device performance [124]. In the end, it provides a statistical estimate of the analysis with a certain confidence level, but at the cost of simulation run-time. Mismatch analysis [30] relies on the circuit configuration and the outcome may have different distributions. This is because there may be cases where there is no analytical dependence of the output on some of the device parameters such as, the threshold voltage V_t or the effective channel length L_{eff}. In such cases, the output curves carry different skews according to the mean value, which cannot be fitted to a normal or log-normal distribution function.

In summary, noise and process variations in analog circuits are analyzed through circuit level simulation, which is time consuming and do not facilitate system level verification. Attempts to handle noise at a higher level of abstraction still lags in providing a unified approach in quantifying the design failures.

1.5 Thesis Objectives

The objective of this thesis is the development of a unified behavioral modeling and verification framework for noise analysis and process variation in analog circuits. In particular, we aim at developing:

- Modeling techniques that could describe the continuous and discrete behavior of a circuit in the presence of noise. For example, by providing mathematical models to capture the statistical and stochastic behaviors at the different levels of design abstraction.

- Quantitative and qualitative validation techniques based on run-time verification that could allow us to integrate process variation to monitor deterministic/statisitical properties and to quantify the failures in the analog design.

1.6 Proposed Methodology

Figure 1.4 shows our proposed analog modeling and verification framework. Given an analog design, the first step is to describe the functional behavior as a system of ODEs. We use the Dymola modeling [37] or the MATLAB [84] tool environment to systematically transform the SPICE netlist to a set of ODEs. Depending on the circuit configuration, the next step is to include thermal and shot noise as a set of stochastic processes that adhere to certain probability distributions to describe the circuit noise behavior as shown in Figure 1.4. This is done through the use of Stochastic Differential Equation (SDE). As there are no known functions/procedures that can automatically incorporate stochastic processes, SDEs have to be generated manually.

To find an analytical solution for the SDEs special mathematical interpretation in the form of stochastic calculus to handle randomness is required [107]. If an analytical

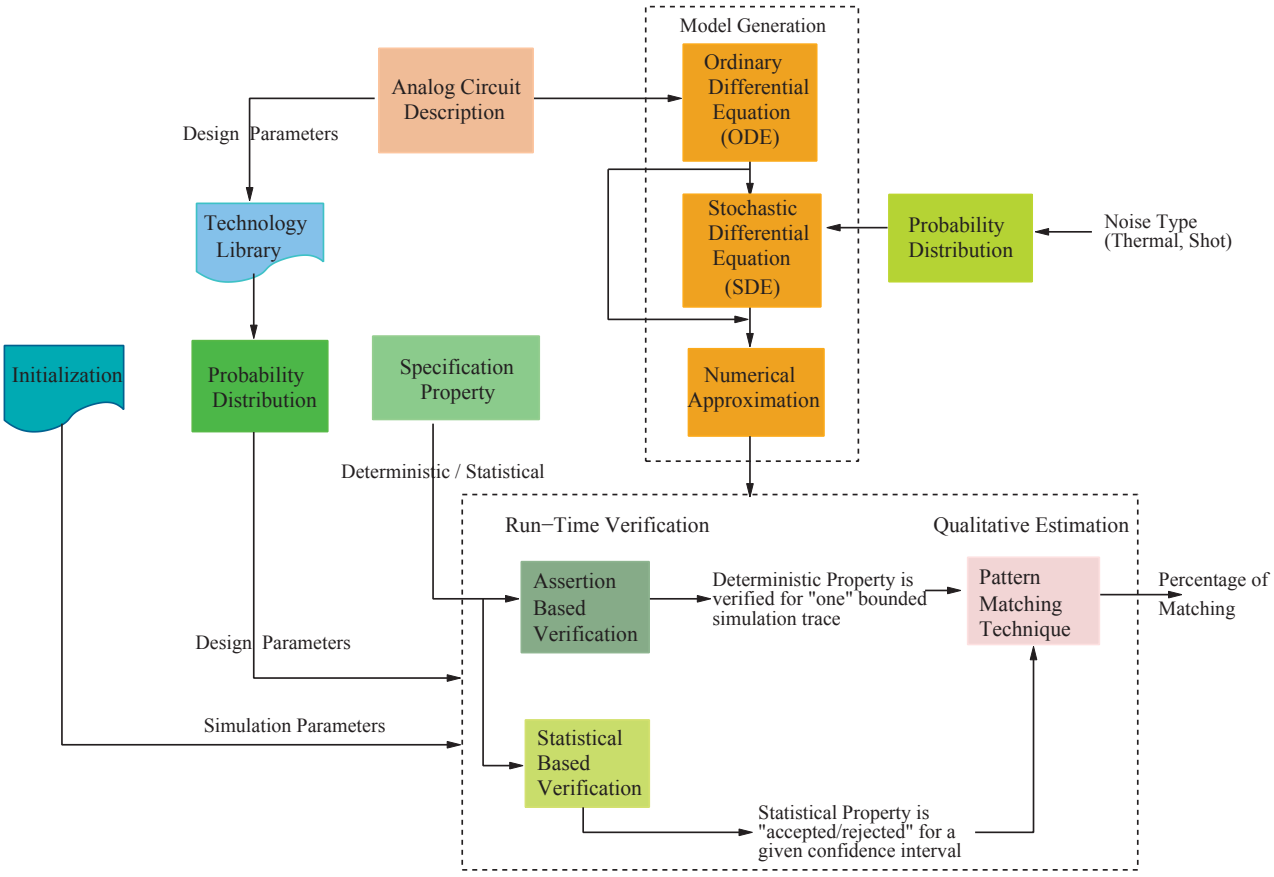


Figure 1.4: Analog Modeling and Verification Framework.

solution is possible, process variation can be integrated in a symbolic simulation [11] or formal verification [140] environment to verify the analog circuit with noise. However, as most analog circuits do not have a closed-form solution, the behavioral noise verification has to rely on well established numerical approximation methods for the SDEs. The details pertaining to the modeling technique based on analytical and numerical approximation of SDEs are presented in Chapter 2.

For process variation, technology vendors create a library of devices with different corners such as *slow, nominal* and *fast* [30]. Each process corner characterizes the device in terms of power consumption, speed, etc., thereby allowing designers to choose from a range of devices based on the application and design requirements. Based on the type of process,

various design parameters in the circuit are calculated using Gaussian distribution with a known $\pm 3\sigma$ deviation as shown in Figure 1.4 and are then passed on as design parameters during simulation..

Thereafter, the SDE numerical approximation of the analog circuit, process variation, and the initial condition of the circuit current and voltages are evaluated in a MATLAB simulation environment using a qualitative verification method in the form of pattern matching and a quantitative verification in the form of assertion/statistical based methods as shown in Figure 1.4. These methods are briefly described in the sequence.

1. Quantitative approaches are run-time verification methods for monitoring whether an execution of the design model violates the design specifications (properties). The deterministic quantitative method is based on finite-state machine (FSM) implementation of simple assertion [12]. These FSM that represent the property of interest are evaluated in a MATLAB environment with noise, process variation, and circuit initial conditions for *"one"* bounded interval, meaning one simulation trace. A simulation trace is defined as the output of any observation points in the analog circuit over a period of time. In the end, the monitor reports if the property has *passed* or *failed* as depicted in Figure 1.4. The details related to assertion based verification methodology are presented in Chapter 3.

2. The statistical technique relies on MATLAB based statistical monitors in the front-end and hypothesis testing in the back-end to verify statistical properties of the design. The property to be verified is represented as a null hypothesis and in the end, a circuit is *accepted/rejected* with a certain confidence level and error margin. The front-end monitors can be classified based on the sampling and re-sampling of the analog output with a known/unknown set of distribution. Popular finance methods such as the *MonteCarlo* [96] and *Bootstrap* [31] can be used to implement the monitors. The

details related to statistical monitors based on MonteCarlo, Bootstrap technique and hypothesis testing are presented in Chapter 4.

3. The missing qualitative analysis of the circuit acceptance/rejection is addressed through a pattern matching method as depicted in Figure 1.4. The pattern matching verification methodology is developed by modifying two popular dynamic programming algorithms [36]: the *"longest common subsequence"* (LCS) and the *"dynamic time warping"* (DTW). The idea of both these algorithms is to find the subsequence simulation trace between an ideal and a non-ideal analog signal and use the combination of MonteCarlo and hypothesis testing to determine the probability of acceptance/rejection as shown in Figure 1.4. The algorithm details are presented in Chapter 5.

1.7 Thesis Contributions

The primary focus of this thesis is on the idea of developing a framework for analog circuit verification in the presence of noise and process variation. The approach allows us to study some of the effects in a traditional analog design flow at a higher level of abstraction. This is quite useful and important for the performance evaluation of circuits for analog design exploration. The thesis makes the following contributions.

- A modeling and a quantitative estimation infrastructure that allows us to capture the noise dynamics in the form of SDEs and integrate process variation for the deterministic monitoring of the specification. We applied the technique on a Tunnel Diode oscillator, a Colpitts oscillator, and a Phase Locked Loop (PLL) circuit for a $0.18\mu m$ fabrication process. We have shown that the properties that are satisfied without noise have failed in the presence of noise and process variation, thereby making the method efficient in finding bugs.

- A statistical approach based on the combination of hypothesis testing with different monitoring (MonteCarlo and Bootstrap) techniques is developed which will increase the confidence level of the design/verification process. We illustrate the proposed approach on a Tunnel Diode oscillator, a Colpitts oscillator and a PLL circuit for a $0.18\mu m$ fabrication process. The method estimates the acceptance/rejection of the circuit with a certain confidence interval.

- A pattern matching based verification approach is developed for the qualitative analysis of the circuit simulation traces that have noise and process conditions to achieve a more meaningful quantification of circuit failures. We extend the LCS and DTW algorithms to handle set of simulation sequences derived from an analog circuit. We perform statistical techniques to estimate the probability of failure. The approach is illustrated on a Colpitts oscillator and a PLL circuit for a $0.18\mu m$ fabrication process. Advantages of the proposed methods are robustness and flexibility to account for a wide range of variations.

- The whole thesis framework is developed as a *AnalogSDE* MATLAB tool for automatic verification of noise and process variation in an analog circuit. The tool is developed using MATLAB based object-oriented approach in form of *object classes* and *functions*.

1.8 Thesis Organization

The rest of the thesis is organized as follows. In Chapter 2, we provide a brief introduction about stochastic differential equation to equip the reader with some notation and concepts that are going to be used in the rest of this thesis. We also discuss about statistical techniques that are necessary to understand Chapter 4. Chapter 3 describes the framework for

verifying the property specification of an analog design using assertion based technique. The effectiveness of this methodology is demonstrated for a Colpitts oscillator, a Tunnel Diode oscillator and a PLL based frequency synthesizer circuit. Chapter 4 presents the methodology for the quantitative analysis using statistical techniques. In this chapter, we compare the efficiency of MonteCarlo and Bootstrap algorithms based hypothesis testing for different benchmark circuits. Next, in Chapter5, we demonstrate the longest closest subsequence (LCSS) and the dynamic time warping (DTW) pattern matching algorithms to ensure the correctness of analog designs with noise and process variation. We illustrate the practical effectiveness of the proposed approach by successfully applying it for the verification of a Colpitts oscillator and a PLL circuit. Appendix A gives an overview of the developed tool, AnalogSDE, including class diagrams, functions and decision procedures. Finally, Chapter 6 concludes the thesis and outlines some future research directions.

Chapter 2

Preliminaries

In this chapter, we provide a brief introduction to the Stochastic Differential Equation (SDE) modeling technique for analog circuits and present an overview of the stochastic calculus needed to derive the analytical and numerical solution. We also present a general idea about different statistical technique that will be used as a part of the verification framework.

2.1 Stochastic Differential Equation (SDE)

An SDE is an ordinary differential equation (ODE) with a stochastic process that can model unpredictable real-life behavior of any continuous system [107]. The random process in the SDE can be purely additive or it may multiply with some deterministic term. The underlying difference between an ODE and an SDE lies in their solution, with the ODE following a smooth trajectory and the SDE will have some random disturbance as shown in Figure 2.1. Because of this, SDE has been the main modeling platform for understanding the outcome of stock prices, population growth and electronics systems [107].

Given a probability space ω [119], a stochastic process in an SDE with state space E is a collection $\{X_t; t \epsilon T\}$ of random variables X_t that takes values in E. If T is countable (T

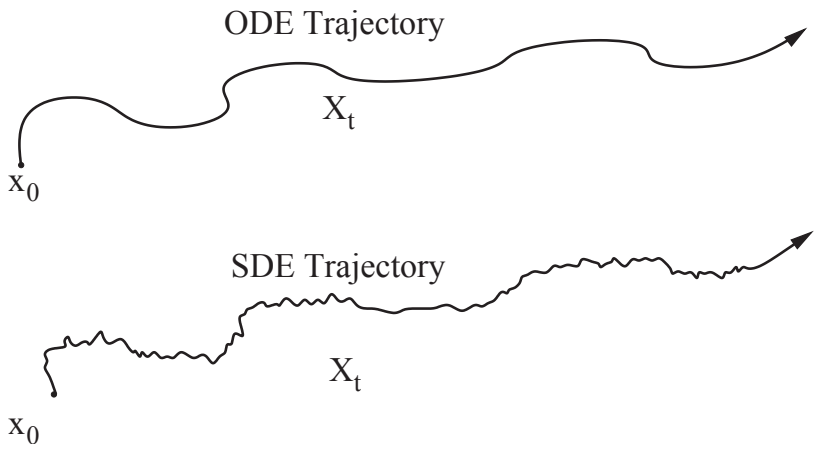


Figure 2.1: ODE vs. SDE [2].

= $\mathbb{N}$ = 0, 1, 2, ...), the process is said to be a *discrete parameter process*, else, a *continuous parameter process*. The random term in the SDE is incorporated as an uncorrelated *white gaussian noise* which can be contemplated as the derivative of Brownian motion [107] (or the Wiener process [56]). A Wiener process is a family of random variables W_t that can be used to model thermal noise in time domain as shown in Figure 2.2. It is indexed by nonnegative real numbers *t*, defined on a common probability space with the following properties:

- W_0 = 0.
- With probability 1, the function $t \rightarrow W_t$ is continuous in *t* as shown in Figure 2.2 (a).
- The process W_t has stationary, independent increments.
- The increment W_{t+s} - W_s has the Normal(0, t) distribution as shown in Figure 2.2 (b).

To understand better, let us consider the population growth model describe by the following ordinary differential equation

$$\frac{dN}{dt} = a(t)N(t); \quad N(0) = A \tag{2.1}$$

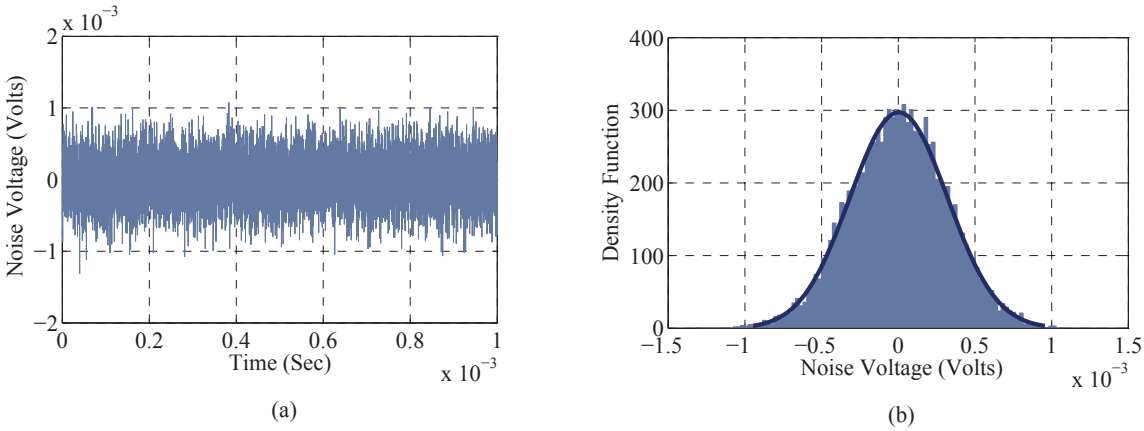


Figure 2.2: Thermal Noise in Time Domain [107].

where *N(t)* is the size of the population at time *t*, and *a(t)* is the relative rate of growth at time *t* and *A* is some initial constant. But, *a(t)* is unknown and is random in nature. Hence a reasonable mathematical interpretation of the randomness for the above equation can be described as

$$\frac{dN}{dt} = a(t)N(t) + \xi_t N(t); \quad N(0) = A \tag{2.2}$$

The term *a(t)N(t)* is the deterministic drift coefficient while the term $\xi_t N(t)$ represents the stochastic effect [107]. In SDE terminology, the above equation can be represented in two forms [107]: *Itô* or *Stratonovich*. Both these form depend on the limit interval that defines the integral of the noise term. As the noise term in an SDE is considered to fluctuate an infinite number of times with infinite variance, different choices of those time interval may lead to different stochastic calculi. For instance, let us consider the following function

$$\int_0^t f(t)dt = \lim_{n\to\infty} \sum_{j=1}^{n} f(\tau_j)(t_{j+1} - t_j) \tag{2.3}$$

where τ_j is in the interval (t_j, t_{j+1}). Then,

$$\begin{aligned} \tau_j &= t_j && \leftarrow Itô \\ \tau_j &= \frac{t_j + t_{j+1}}{2} && \leftarrow Stratonovich \end{aligned} \tag{2.4}$$

If we consider ξ_t in the SDE Equation (2.2) to be the path-wise derivative of Brownian motion (or Wiener process) dB_t, then Equation (2.2) can be written in *Itô* differential and integral form as given by

$$\begin{aligned} dN &= a(t)N(t)dt &&+N(t)dB_t \\ N &= \int_0^t a(s)N(s)ds &&+ \int_0^t N(s)dB_s \end{aligned} \tag{2.5}$$

However, to solve Equation (2.5), traditional calculus lack the structure to handle stochastic process, and hence there is a need for a special mathematical interpretation in the form of stochastic calculus to solve the equations involving Brownian motion [107]. If we consider the random term to be the approximation to continuously fluctuating noise and with finite memory, it is appropriate to use the *Stratonovich* representation. On the other hand, if the random term is considered as a finite pulse, it is suitable to use the *Itô* form. In addition, *Stratonovich* SDEs are easier to solve analytically, and the *Itô* SDEs are better handled using numerical schemes.

Stochastic calculus uses the concept of expectation and *Itô* isometry to solve SDEs. Expectation determines the behavior of any system in the absence of randomness and hence it is easy to conclude that the expectation of any random process (Brownian or Wiener) is zero. As Brownian motion cannot be solved using definite integral, the goal of *Itô* isometry is to replace the Brownian motion dB_s by a deterministic term ds for solving SDEs. Table 2.1 summarizes some of the theorems and axioms that are key for solving the SDEs [107].

For analog circuits, the noise modeling is a straight forward approach that relies on the extraction of the ODE and transforming them to SDEs. Once, the SDE models are generated, we can apply stochastic calculus to find an analytical solution. A detailed analysis of finding a closed form solution is described in the next section.

Table 2.1: SDE Formulas

Expectation of a Brownian motion	$\int_0^t F_s dB_s = 0$
Substitution-by-parts	$d(e^t X_t) = e^t X_t dt + e^t dX_t$
Itô Isometry property	$E\left(\left[\int_0^t F_s dB_s\right]^2\right) = E\left(\left[\int_0^t F_s^2 ds\right]\right)$
Noise	$N_t = X_t - E[X_t]$
Variance of the noise	$Var[N_t^2] = E[X_t^2] - E[X_t]^2$

2.1.1 Finding the Analytical Solution of SDE

Consider the mixer circuit as shown in Figure 2.3. If we assume that the transistor operates

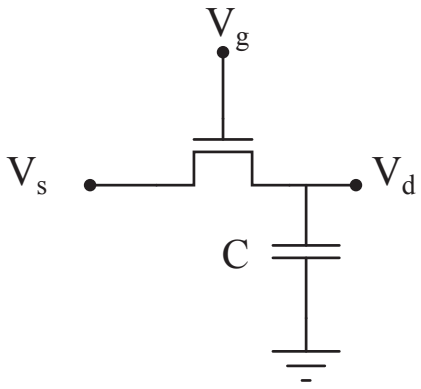


Figure 2.3: Sampling Mixer Circuit.

in the *triode* region, the voltage across the capacitor can be written as

$$C\frac{dV_d}{dt} = K(V_{gs} - V_t)V_{ds} - \frac{K}{2}V_{ds}^2 \tag{2.6}$$

where, $K = \mu C_{ox}\frac{W}{L}$. Rewriting Equation (2.6) and neglecting nonlinear terms we have

$$C\frac{dV_d}{dt} + K(V_g - V_t)V_d = K(V_g - V_t)V_s \tag{2.7}$$

Assuming the noise at the *gate*

$$K(V_g + \sigma\xi_t - V_t)V_d + C\frac{dV_d}{dt} = K(V_g + \sigma\xi_t - V_t)V_s \tag{2.8}$$

Replace $\xi_t dt = dW_t$; $V_d = X_t$; $g_t = K(V_g - V_t)$; $V_s = u_t$; in Equation (2.8) we have,

$$g_t X_t dt + K\sigma X_t dW_t + C dX_t = g_t u_t dt + K\sigma u_t dW_t \tag{2.9}$$

$$\implies C dX_t = g_t(u_t - X_t)dt + K\sigma(-X_t + u_t)dW_t \tag{2.10}$$

Hence, from Equation (2.10) we have

$$dX_t = \frac{g_t}{C}(u_t - X_t)dt + \frac{K\sigma}{C}(-X_t + u_t)dW_t \tag{2.11}$$

Equation (2.11) is the SDE for the mixer. Integrating both sides of Equation (2.11), we have

$$X_t = X_0 + \int_0^t \frac{g_t}{C}(u_s - X_s)ds + \int_0^t \frac{K\sigma}{C}(-X_s + u_s)dW_s \tag{2.12}$$

Taking Expectation on Equation (2.12), we have

$$E[X_t] = E[X_0] + E\left[\int_0^t \frac{g_t}{C}(u_s - X_s)ds\right] \tag{2.13}$$

Since

$$E\left[\int_0^t \frac{K\sigma}{C}(-X_s + u_s)dW_s\right] = 0, \tag{2.14}$$

differentiating Equation (2.14) and rearranging it, we have

$$\frac{dE[X_t]}{dt} + \frac{g_t}{C}E[X_t] = \frac{g_t}{C}u_t \tag{2.15}$$

Equation (2.13) describes the mean of the output process which is the output of the circuit without noise. To find $E[X_t^2]$ from Equation (2.13) we use the following theorem based on stochastic calculus

The Quadratic Variance is given by

$$\langle W_t \rangle = \lim \sum_{i=1}^{n} \left(W_{t_i} - W_{t_{(i-1)}}\right)^2 \tag{2.16}$$

For Wiener process $\langle W_t \rangle = t$ and based on *Ito* isometry, we have

$$\left\langle \int_0^t f(W_s, s) dW_s \right\rangle = \int_0^t f^2(W_s, s) ds \tag{2.17}$$

Based on stochastic integration definition, we have

$$\phi(X_t) = \phi(X_0) + \int_0^t \phi^{'}(X_s) dX_s + \frac{1}{2} \int_0^t \phi^{''}(X_s) d\langle X_s \rangle \tag{2.18}$$

where $\phi(x)$ is any twice differential form function with continuous second derivative. *Ito* allows non-linear transformation of stochastic process, we need some stochastic calculus to solve for $E[X_t^2]$.

Assuming the second order continuous function is $\phi(x) = x^2$, then we have

$$\phi^{'}(x) = 2x; \quad \phi^{''} = 2; \tag{2.19}$$

Then Equation (2.18) becomes

$$X_t^2 = X_0^2 + \int_0^t 2X_s dX_s + \langle X \rangle_t \tag{2.20}$$

Now the goal is to remove $\langle X \rangle_t$. Integrating Equation (2.11), we have

$$X_t = \int_0^t \frac{g_t}{C}(u_s - X_s) ds + \int_0^t \frac{K}{C}(u_s - X_s) \sigma dW_s \tag{2.21}$$

Based on *Ito* isometry property, we have

$$\langle X_t \rangle = \frac{\sigma^2 K^2}{C^2} \int_0^t (u_s - X_s)^2 ds \tag{2.22}$$

Hence, substituting Equation (2.22) and (2.11) in (2.20) we have

$$E[X_t^2] = E[X_0^2] + \int_0^t \frac{2g_t}{C} \left(E[X_s] u_s - E[X_s^2] \right) ds + \int_0^t \frac{\sigma^2 K^2}{C^2} (E[u_s - X_s])^2 \tag{2.23}$$

Let us take output noise $N_t = X_t - E[x_t]$ then we have

$$E[N_t^2] = E[X_t^2] - E[X_t]^2 \tag{2.24}$$

Differentiating both sides of Equation (2.24) we have

$$\frac{dE[N_t^2]}{dt} = \frac{dE[X_t^2]}{dt} - 2E[X_t]\frac{dE[X_t]}{dt} \tag{2.25}$$

Differentiating Equation (2.23) we have,

$$\frac{dE[X_t^2]}{dt} = \frac{2g_t}{C}\left(E[X_t]u_t - E[X_t^2]\right) + \frac{\sigma^2K^2}{C^2}(E[u_t - X_t])^2 \tag{2.26}$$

From Equation (2.23) and substituting Equations (2.26), and (2.15), we have

$$\frac{dE[N_t^2]}{dt} = \left(\frac{-2g_t}{C} + \frac{\sigma^2K^2}{C^2}\right)E[X_t^2] - \left(\frac{2u_t\sigma^2K^2}{C^2}\right)E[X_t] + \frac{2g_t}{C}E[X_t]^2 + \frac{u_t^2\sigma^2K^2}{C^2} \tag{2.27}$$

Rearranging we have

$$\frac{dE[N_t^2]}{dt} + \left(\frac{2g_t}{C} - \frac{\sigma^2K^2}{C^2}\right)E[N_t^2] = -\left(\frac{2u_t\sigma^2K^2}{C^2}\right)E[X_t] + \frac{\sigma^2K^2}{C^2}E[X_t]^2 + \frac{u_t^2\sigma^2K^2}{C^2} \tag{2.28}$$

Rearranging the above equation, we have

$$\frac{dE[N_t^2]}{dt} + \left(\frac{2g_t}{C} - \frac{\sigma^2K^2}{C^2}\right)E[N_t^2] = \left(\frac{2u_t\sigma^2K^2}{C^2}\right)(E[X_t] - u_t) \tag{2.29}$$

In summary, the equation for mean and variance of the output process in the presence of noise at the gate is given by

$$\begin{aligned} &\frac{dE[X_t]}{dt} + \frac{g_t}{C}E[X_t] = \frac{g_t}{C}u_t \\ &\frac{dE[N_t^2]}{dt} + \left(\frac{2g_t}{C} - \frac{\sigma^2K^2}{C^2}\right)E[N_t^2] = \left(\frac{2u_t\sigma^2K^2}{C^2}\right)(E[X_t] - u_t) \end{aligned} \tag{2.30}$$

2.1.2 Numerical Approximation of the SDE

The methods based on numerical analysis are reported in [71], which involve discrete time approximation in a finite time interval over the sample paths. To realize SDE based numerical method, let us apply to a tunnel diode oscillator circuit.

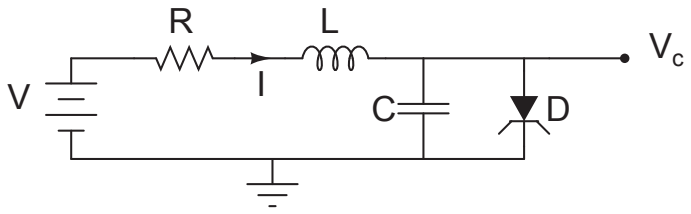


Figure 2.4: Tunnel Diode Oscillator

The current through the resistor and inductor I and the voltage across the capacitor V_C can be described by

$$\begin{aligned} \dot{V_C} &= \frac{1}{C}(-I_d(V_C) + I) \\ \dot{I} &= \frac{1}{L}(-V_C - \frac{1}{G}I + V) \end{aligned} \tag{2.31}$$

where $I_d(V_C)$ describes the non-linear tunnel diode behavior. If we consider thermal noise in the passive elements (*R, L, C*) and shot noise in the diode *D*, a reasonable mathematical interpretation of the randomness for Equation (2.31) can be described as

$$\begin{aligned} \dot{V_C} &= \overbrace{\frac{1}{C}(-I_d(V_C) + I_L)}^{A^1} + \alpha\xi_1(t) + \zeta(t) \\ \dot{I} &= \overbrace{\frac{1}{L}(-V_C - RI + V)}^{A^2} + \sum_{k=2}^{3} \alpha\xi_k(t) + \zeta(t) \end{aligned} \tag{2.32}$$

where $\sum_{k=1}^{3} \alpha\xi_k$ represents the thermal noise model for the passive elements with certain amplitude α and $\zeta(t)$ represents Poisson white shot noise (PWSN) [70] that has random pulses, which occurrence is based on Poisson distribution. The strengths of the pulses takes a white noise (gaussian) distributed independent values as shown in Figure 2.5 (a), (b).

During noise analysis, choosing a fixed amplitude for such random pulse does not make the evaluation completely random. Hence, models based on PWSN as shown in Figure 2.5 allows Poisson distribution for the random pulses and a white noise (Gaussian) distribution for its amplitude. Mathematically, the probability that a random sequence of *k*

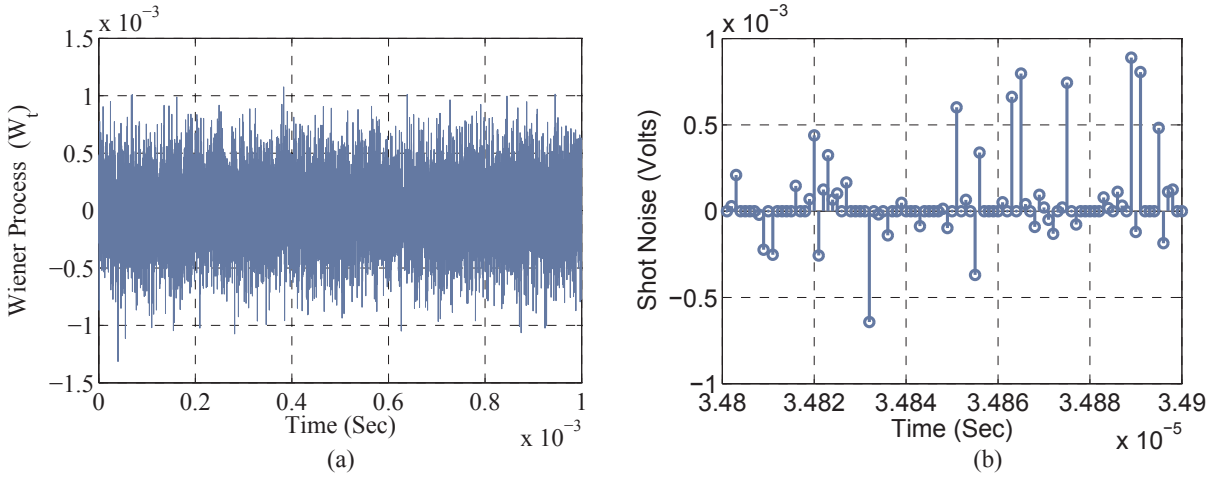


Figure 2.5: Poisson White Shot Noise (PWSN) [70].

pulses occurs in the interval $(0, t)$ is given by

$$Prob\{n(t) = k\} = \frac{(\lambda t)^k e^{-\lambda t}}{k!} \tag{2.33}$$

If we consider dB_t and dB_{st} to represent stochastic processes for the thermal and shot noise, respectively, then Equation (2.32) can be rewritten as

$$\begin{aligned} dV_C &= \frac{1}{C}(-I_d(V_C) + I)dt + \alpha dW_t + dW_{st} \\ dI &= \frac{1}{L}(-V_C - RI + V)dt + \sum_{k=2}^{3} \alpha dW_{kt} + dW_{kst} \end{aligned} \tag{2.34}$$

Based on the simplest *Euler-Maruyama* time discretization approach [71], Equation (2.34) can be rewritten as

$$\begin{aligned} V_{C_{n+1}} &= V_{C_n} + \frac{\Delta_n}{C}(-I_d(V_{C_n}) + I_n) + \alpha \Delta W_{1n} + \Delta W_{sn} \\ I_{n+1} &= I_n + \frac{\Delta_n}{L}(-V_{C_n} - RI_n + V) + \sum_{k=2}^{3} \alpha \Delta W_{k_n} + \Delta W_{sn} \end{aligned} \tag{2.35}$$

where for time step τ,

$$\Delta_n = \tau_{n+1} - \tau_n; \;\; \Delta W_n = \Delta W_{sn} = W_{\tau n+1} - W_{\tau n} \tag{2.36}$$

for *n=0,1,2*$\cdots$*N-1*; and for maximum *N* simulation steps.

In general, any SDE that takes a form as in Equation (2.35) is suited to represent the additive noise behavior in an analog circuit. Higher order numerical approximation such as the *Milstein* method [71] uses multiple stochastic integrals in terms of several Wiener processes and can be used to model the multiplicative noise behavior. To better understand the *Milstein* method of noise model, let us consider the tunnel diode oscillator shown in Figure 2.4. If we consider noise to exist in multiplicative form, then, rewriting Equation (2.34) in matrix form, we get

$$dY = \begin{pmatrix} dV_C \\ dI \end{pmatrix} = \begin{pmatrix} A^1 \\ A^2 \end{pmatrix} dt + \begin{pmatrix} I \\ V_C \end{pmatrix} dW_t^1 + \begin{pmatrix} V_C \\ I \end{pmatrix} dW_t^2 \tag{2.37}$$

with

$$b^1 = \begin{pmatrix} b^{(1,1)} \\ b^{(2,1)} \end{pmatrix} = \begin{pmatrix} I \\ V_C \end{pmatrix}; b^2 = \begin{pmatrix} b^{(1,2)} \\ b^{(2,2)} \end{pmatrix} = \begin{pmatrix} V_C \\ I \end{pmatrix};$$

A general Milstein approximation for the SDE can be written as

$$Y_{n+1}^k = Y_n^k + a^k \Delta_n + \sum_{j=1}^{M} b^{j,k} \Delta W^j + \sum_{j_1,j_2=1}^{M} L^{j_1} b^{k,j_2} I(j_1, j_2) \tag{2.38}$$

Applying Equation (2.38) to Equation (2.35), we get

$$\begin{aligned} \begin{pmatrix} V_{C_{n+1}} \\ I_{n+1} \end{pmatrix} &= \begin{pmatrix} V_{C_n} \\ I_n \end{pmatrix} + \begin{pmatrix} A^1 \\ A^2 \end{pmatrix} \Delta_n + \begin{pmatrix} I_n \\ V_{C_n} \end{pmatrix} \Delta W_n^1 \\ &+ \begin{pmatrix} V_{Cn} \\ I_n \end{pmatrix} \Delta W_n^2 + \sum_{j_1,j_2=1}^{2} L^{j_1} b^{k,j_2} I(j_1, j_2) \end{aligned} \tag{2.39}$$

where L^j [71] is the partial differential operator as defined by, $L^j = \sum_{k=1}^{N} b^{k,j} \frac{\partial}{\partial x^k}$ and

$I(j_1, j_2)$ is the *Ito* integral. Expanding L^j for $j = 1, 2$, we get

$$\begin{aligned} L^1 &= \sum_{k=1}^{2} b^{k,1} \frac{\partial}{\partial x^k} = b^{1,1} \frac{\partial}{\partial x^1} + b^{2,1} \frac{\partial}{\partial x^2} \\ L^2 &= \sum_{k=1}^{2} b^{k,2} \frac{\partial}{\partial x^k} = b^{1,2} \frac{\partial}{\partial x^1} + b^{2,2} \frac{\partial}{\partial x^2} \end{aligned} \tag{2.40}$$

Hence, the final Milstein numerical approximation for Equation (2.40) is given by

$$\begin{aligned} \begin{pmatrix} V_{C_{n+1}} \\ I_{n+1} \end{pmatrix} &= \begin{pmatrix} V_{Cn} \\ I_n \end{pmatrix} + \begin{pmatrix} \frac{1}{C}(-I_d(V_{Cn}) + I_n) \\ \\ \frac{1}{L}(-V_{Cn} - \frac{1}{G} I_n + V) \end{pmatrix} \Delta_n \\ &+ \begin{pmatrix} I_n \\ V_{Cn} \end{pmatrix} \Delta W_n^1 + \begin{pmatrix} V_{Cn} \\ I_n \end{pmatrix} \Delta W_n^2 + \begin{pmatrix} V_{Cn} \\ I_n \end{pmatrix} I(1,1) \\ &+ \begin{pmatrix} I_n \\ V_{Cn} \end{pmatrix} I(1,2) + \begin{pmatrix} I_n \\ V_{Cn} \end{pmatrix} I(2,1) + \begin{pmatrix} V_{Cn} \\ I_n \end{pmatrix} I(2,2) \end{aligned} \tag{2.41}$$

where the *Ito* integral $I(j_1, j_2)$ [71] can be expressed as

$$I(1,1) = I(2,2) = \int_{t_n}^{t_{n+1}} \int_{t_n}^{t} dW_s^{j_1} dW_t^{j_2} = \frac{1}{2} \left((\Delta W_n^{j_1})^2 - \Delta_n \right)$$

$$I(1,2) = I(2,1) = \int_{t_n}^{t_{n+1}} \int_{t_n}^{t} dW_s^{j_1} dW_t^{j_2} = \frac{1}{2} \left(\Delta W_n^{j_1} \Delta W_n^{j_2} \right)$$

Equation (2.41) represents the numerical approximation for the tunnel diode oscillator. Unlike analytical solution, numerical approximation tends to have some error. Mathematically, this absolute error at the final time instant "T" is defined as,

$$\varepsilon(\delta) = E(|X_T - Y_N|) \leq \sqrt{E(|X_T - Y_N|)^2} \tag{2.42}$$

The absolute error determines how close the numerical solution "Y" is with respect to the analytical solution "X". For the case where the analytical solution cannot be determined,

Table 2.2: Statistical Estimation Error

	H_0 **is True**	H_1 **is True**
Accept H_0	Correct Decision	Wrong Decision - Type II Error
Reject H_0	Wrong Decision - Type I Error	Correct Decision

then the absolute error can be calculated as the absolute difference between the numerical approximation solutions that are derived with different step-size Δ_n and $\frac{\Delta_n}{2}$ [2].

2.2 Statistical Hypothesis Testing

Hypothesis testing [96] is the use of statistics to make decision about acceptance or rejection of some statements based on the data from a random sample, meaning, to determine the probability that a given hypothesis is true. Hypothesis testing in general has two parts:

1. *Null hypothesis*, denoted by H_0, which is what we want to test (e.g., $jitter\_period \leq$ *3.2 ns*), and

2. *Alternative hypothesis*, denoted by H_1, which is what we want to test against the null hypothesis (e.g., $jitter\_period >$ *3.2 ns*).

If we reject H_0, then the decision to accept H_1 is made. The conclusion is drawn with certain probability of error for a specific confidence interval as summarized in Table 2.2. The error associated with such statistical estimate for a given confidence interval can be classified to be [85]:

Type I or False positive - H_0 is rejected when it is in fact true with error α.

Type II or False negative - H_0 is true when it is in fact false with error β.

The quantification of error can be made by measuring the probability of accepting/rejecting H_0 when it is actually true/false, respectively. If α and β denote such probabilities then, mathematically they can be represented as

$$\begin{aligned} \alpha &= \textit{Pr}\{\ reject\ H_0 \mid H_0\ is\ true\ \} \\ \beta &= \textit{Pr}\{\ accept\ H_0 \mid H_0\ is\ false\ \} \end{aligned} \tag{2.43}$$

The choice to accept or reject is determined by the direction with which the null hypothesis is proved to be true or false. This direction is decided based on a one-tailed test (*upper* or *lower*) or a *two-tailed* test as shown in Figure 2.6.

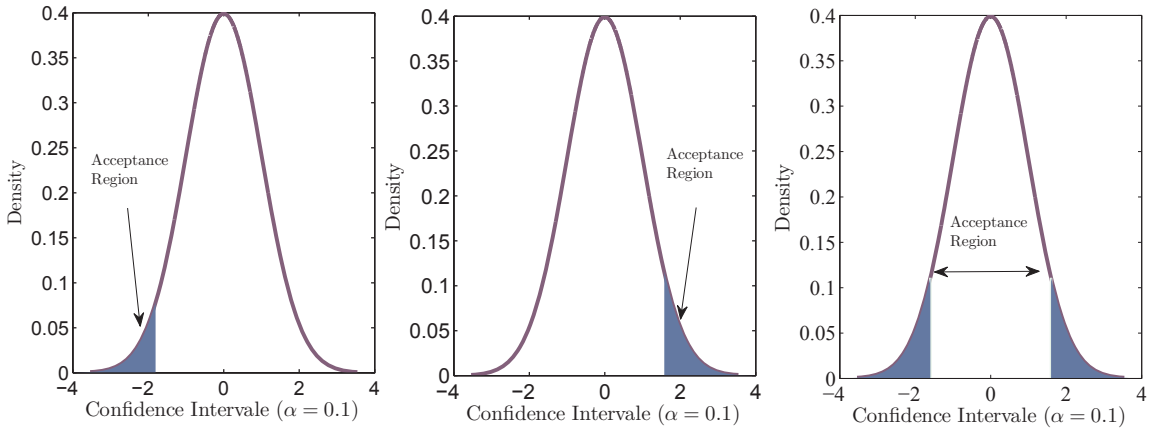


Figure 2.6: Accept/Reject Hypothesis Testing

The *upper tail* distribution represents the rejection region for the case where a large value of the test statistic provide evidence for rejecting H_0. On the other hand, a *lower* tail distribution is used if only a small value of the test statistic show proof of H_0 rejection [49].

The *bounded hypothesis testing* [49] also called the *two-tailed* test is determined by a bounded region $[x_1, x_2]$, such that such that H_0 satisfies the following:

$$H_0 : P(x_1 < X < x_2) = P(X < x_2) - P(X < x_1) = 1 - \alpha \tag{2.44}$$

For instance, $\alpha = 0.05$ and $\alpha = 0.01$ refer to the confidence level of 95% and 99%, respectively. For the case, where the confidence interval is divided equally between the lower and upper bounds, the probability can be determined as follows:

$$\begin{cases} P(X < x_1) = \dfrac{\alpha}{2} = 0.05 \\ P(X < x_2) = 1 - \dfrac{\alpha}{2} = 0.95 \end{cases} \tag{2.45}$$

In any of the above hypothesis testing measures, if the observed sample data over a given interval is within some critical region, then we reject the null hypothesis H_0, else we accept H_0 as shown by the shaded region in Figure 2.6. In general, the steps in statistical hypothesis testing can be summarized as follows:

1. State the null and alternative hypothesis.
2. Take a random sample from the population of interest.
3. Estimate the statistical measure related to the null hypothesis.
4. Interpret the results to make a decision about acceptance/rejection of the null hypothesis using *critical value* or *p-value* approach with certain standard error.

The *critical-value* approach [49] determines a critical region in which the null hypothesis will be rejected. It depends on the type of tail test (*upper lower* or *two tailed*), observed value and the significant level α. The observed value T_{obs} is calculated based on the sample mean $\bar{x}$, the mean value under the null hypothesis and standard error $\bar{\sigma}$ as described below,

$$T_{obs} = \frac{\bar{x} - \mu_0}{\bar{\sigma}} \tag{2.46}$$

If the observed value T_{obs} is greater than the critical value, we reject the null hypothesis H_0 otherwise, we retain H_0. The *P-value* approach [49] involves defining the probability of the test statistic to be in the direction of the alternative hypothesis, when the null hypothesis is true. If the derived P-value tends to be smaller it is more likely to reject H_0.

The accuracy of the hypothesis testing depends on how good the sample statistics (*mean*, *variances* and *percentiles*) that determines the standard error are estimated. Sampling by far is concerned with the selection of a subset of the observed data to make a desired statistical inference. Based on the sampling method used one may be able to derive different standard errors and hence the accuracy of the results may vary during hypothesis testing. Some of the popular techniques such as the *MonteCarlo* [96], and *BootStrap* [31] are widely used in the financial sector and the extent of their application for analog circuit verification are detailed in Chapters 4 and 5.

Chapter 3

Verification using Deterministic Monitors

This chapter presents a framework for verifying the property specification of an analog and MS designs using assertion based technique. The framework allows us to model and verify the deterministic property in the presence of shot noise, thermal noise, and process variations. The idea is to use stochastic differential equations (SDE) to model noise in additive and multiplicative form and then combine process variation in a runtime verification environment. The practical effectiveness of the proposed framework are compared for Colpitts oscillator, Tunnel Diode oscillator and a Phase Locked Loop (PLL) based frequency synthesizer circuit.

3.1 Introduction

Verification approaches that increase the probability of designs being correct the first time is the key to a successful tapeout, and methodologies that could be easily integrated into the existing verification flow can lead us to reduction in debugging time and cost. Traditionally,

designs were verified based on a constrained random test environment. The idea is to use stimulus generation (testbench) that verify a specific functionality of the design [132]. This approach is known as run-time verification [80]. High-level HDLs provide mechanisms to create complex stimulus patterns and facilitate the re-use of the testbench models.

For a robust verification environment, every test should facilitate a way to detect and isolate bugs automatically and dynamically. This can be accomplished efficiently using assertions in a run-time verification. Assertion Based Verification (ABV) [127] is a debugging technique that has played a central role in the verification of SoC designs. An assertion is a simple description of a property specification for identifying the design failures. For instance, if a property that is being monitored does not behave appropriately, the assertion fails and the user is notified [46]. Depending on the severity of the failure, it could even stop the simulation. The biggest advantage of writing assertions is that, it could be re-used for future designs and can also be used successfully with formal verification. For instance, to check for any violation between two mutually exclusive signals A and B, following assertion can be used,

```
if (A and B) then
  Violation = '1';
end if
```

An assertion based environment is shown in Figure 3.1. An assertion can be constructed as a finite state machine (FSM) [72] with a set of timers constraints specified on each state location. The timer constraints are defined over a set of design variables that form the stimulus to the design under test (DUT). The monitors can also be constructed using FSM, with the acceptance condition to verify the property.

Both the stimulus and monitors can be implemented using any HDLs. The whole

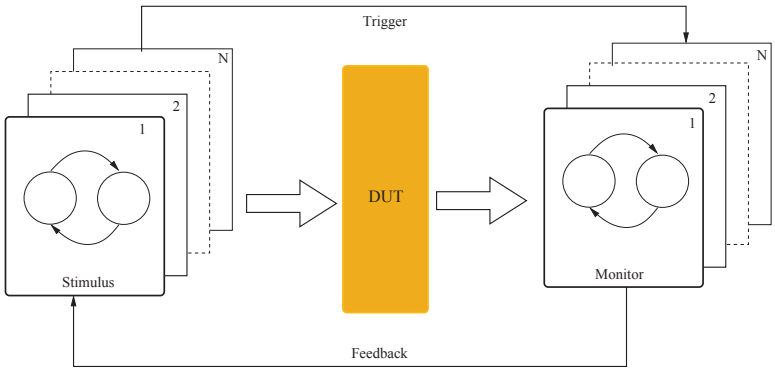


Figure 3.1: Assertion Based Environment

environment can be simulated using any standard simulator to perform run-time verification [46]. A communication mechanism can be established in an automatic fashion between the stimulus generator and the monitor. This is especially helpful when regression test is carried with the feedback signals from the monitor guiding the stimulus to choose next test vector.

The run-time verification environment for AMS designs is still emerging and in the current state-of-the art methodologies have been developed to verify the functional aspect of the design. Due to the lack of a unified verification environment, the uncertainties due to noise and process variation are seldom handled using top-level simulation. To address this, we propose in this chapter a run-time verification framework for monitoring the property of an analog/MS circuit with process variation, thermal/shot noise in additive and multiplicative form.

3.2 Assertion Based Verification Methodology

Figure 3.2 shows the overall assertion based run-time verification methodology. Thereafter, given an analog design described as a system of *ODEs*, the idea is to include a stochastic

process that describes the noise behavior. Due to the statistical behavior of the noise, we propose to use stochastic differential equations (SDE) as an analog noise model in additive and multiplicative forms as described by the tunnel diode oscillator example in Chapter 2.

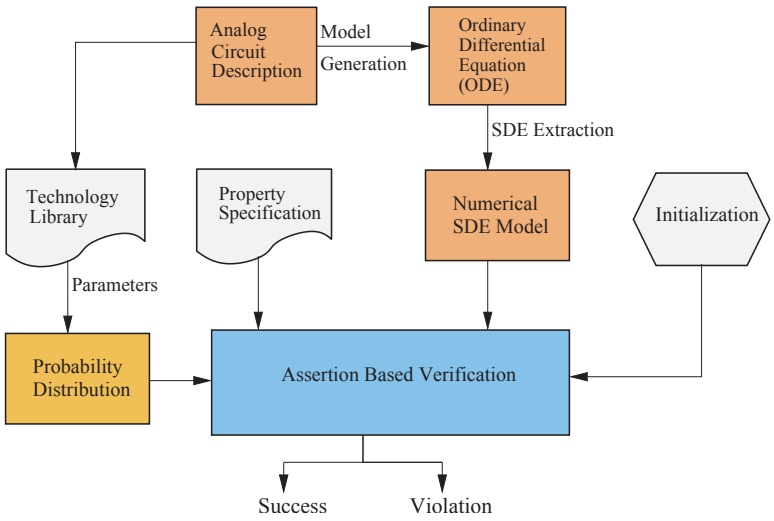


Figure 3.2: Assertion Based Run-Time Verification

The implementation of the time-domain thermal noise model based on Wiener process is described in Algorithm 3.1.

Algorithm 3.1 Wiener Process Generation

Require: $\Delta T, SEED$
Ensure: $\Delta T > 0$
if $SEED = 0$ **then**
  $randn(state, 0)$
else if $SEED = 1$ **then**
  $randn(state, sum(clock))$
else
  $Undefined\ SEED!$
end if
$DW = \sqrt{\Delta T} \times randn$
$W = W + DW$
return W, DW

This time-domain generation depends on the simulation step-size ΔT and the type of $SEED$ used. $SEED$ is a control parameter for generating pseudo-random numbers. Based on the $SEED$ value, the algorithm generates either a fixed ($SEED$=0) or a variable ($SEED$ =1) pseudo random number (lines 1-7). This is followed by an incremental noise generation (lines 8-9). The implementation of PWSN as described in Chapter 2 is shown in Algorithm 3.2. The algorithm uses the built-in MATLAB Poisson function for generating the random pulses (line 1). Then, the amplitude of those pulses are determined using Gaussian distribution (line 2).

Algorithm 3.2 Shot Noise Generation

Require: $\Delta T, N$
Ensure: $\Delta T,\ N\ >\ 0$
1: $Temp = random('Poisson', N)$
2: $V_{Shot}(Temp) = sqrt(\Delta T) \times randn$
3: **return** V_{Shot}

For process variation, different circuit parameters are derived using Gaussian distribution with a known $\pm 3\sigma$ deviation as described in Algorithm 3.3. Technology vendors provide the *lower* and the *upper* bound associated with the circuit parameter variation. Based on the given upper/lower limits, the algorithm generates "n'' different values for the circuit parameters with the pseudo random number $randn$ (line 3). The parameter generation has the probability density function (PDF) that take a Gaussian distribution (lines 1-2).

Algorithm 3.3 Process Parameter Variation

Require: *lower_bound, nominal_bound, upper_bound, randn, inc, sigma_bound, n*
Ensure: $n\ >\ 0$
1: $Dist \leftarrow lower\_bound : inc : upper\_bound$
2: $PDF \leftarrow (1/(\sqrt{(2 \times \pi)} \times sigma\_bound)) \times exp(\frac{-(Dist - nominal\_bound)^2}{(2 \times (sigma\_bound)^2)})$
3: $Param \leftarrow sigma\_bound \times randn(n, 1) + nominal\_bound$
4: **return** $Param$

For environment constraints, this may include the amplitude of the noise, initial conditions of the circuit current and voltages. The environment constraints are passed as a parameter to the design under verification during simulation along with process variation. The SDE numerical approximation of the design, along with the properties to be monitored, and the environment constraints are evaluated using assertions in a MATLAB [123] simulation environment for "one" simulation trace. The implementation of finite state machine (FSM) based assertion and the corresponding algorithm is shown in Figure 3.3 and Algorithm 3.4, respectively.

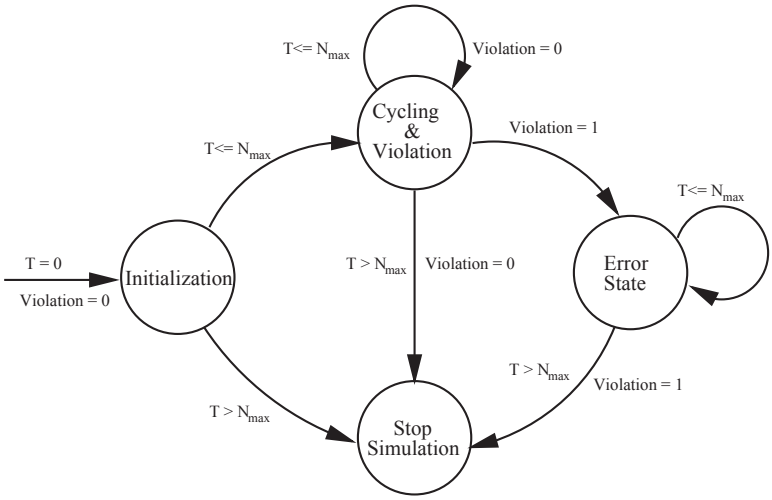


Figure 3.3: Assertion Monitoring

The FSM has four states namely, *Initialization, Cycling & Violation, Error and Stop Simulation*. The maximum simulation time, N_{max}, and inputs like initial voltage, current and output violation are set in the *Initialization* state (lines 2-4 in Algorithm 3.4). Here, N_{max} represents a single simulation trace for which the design will be evaluated. As soon as the simulation starts, the FSM goes to the *Cycling & Violation* state and remains there until the time $T \leq N_{max}$ and $Violation = 0$ (lines 5-6 in Algorithm 3.4). An assertion is a piece of code that evaluates the outputs of the simulator and checks whether the property satisfies the design specification. If the property is satisfied, the monitor reports the

satisfaction. Otherwise, the monitor can assert *violation = 1* and can possibly enter to an *Error* state (lines 7-8 in Algorithm 3.4) or terminate the simulation using *exit* command (*Stop Simulation* state) (lines 9-11 in Algorithm 3.4) as shown in Figure 3.3. The monitor could be as simple as observing a current or voltage, or could be more complicated, taking several signals, processing and then comparing them against the expected results. The monitors could be constructed so that signals could be observed in an *online* or *offline* fashion [132]. While the online monitoring is more practical when simpler properties are needed to be verified and violations are identified as soon as they occur, offline monitors allow the verification of more complex properties but require the gathering of simulation results which can cost a lot of memory resources.

Algorithm 3.4 Assertion Based Verification:

Ensure: N_{max}, T
Ensure: $N_{max} > 0, T > 0$
1: **for** $i \leftarrow 1$ *to* N_{max} **do**
2: **if** $(State = Initialization \ \ and \ \ T \leq N_{max})$ **then**
3: $Violation \leftarrow \ '0'$
4: $I \leftarrow AssignInputs$
5: **else if** $(State = Cycling \ \& \ Violation \ \ and \ \ T \leq N_{max})$ **then**
6: $Violation \leftarrow Evaluate_{trace}(Violation, \ I, \ Sim_{trace})$
7: **else if** $(State = Error \ \ and \ \ T \leq N_{max} \ \ and \ \ Violation = \ '1')$ **then**
8: $Violation \leftarrow \ '1'$
9: **else if** $(State = Stop \, Simulation \ \ or \ \ T > N_{max})$ **then**
10: $Violation \leftarrow \ '1'$
11: $Exit$
12: **else**
13: {"INVALID STATE"}
14: **end if**
15: **end for**
16: **return** $Violation$

3.3 Applications

To illustrate the efficiency of the proposed methodology, we have applied it on several benchmark circuits, including a tunnel diode oscillator [57], a Colpitts oscillator [69] and a PLL based frequency synthesizer in a MATLAB environment. Experiments were run on a Windows Vista OS, AMD Dual-Core Processor with 4GB RAM.

3.3.1 Tunnel Diode Oscillator

The circuit diagram of a tunnel diode oscillator is shown in Figure 2.4 (Chapter 2). The tunnel diode exploits a phenomenon called resonant tunneling due to its negative resistance characteristic at very low forward bias voltages. This means that for some range of voltages, the current decreases with increasing voltage as shown in Figure 3.4. This characteristic makes the tunnel diode useful as an oscillator.

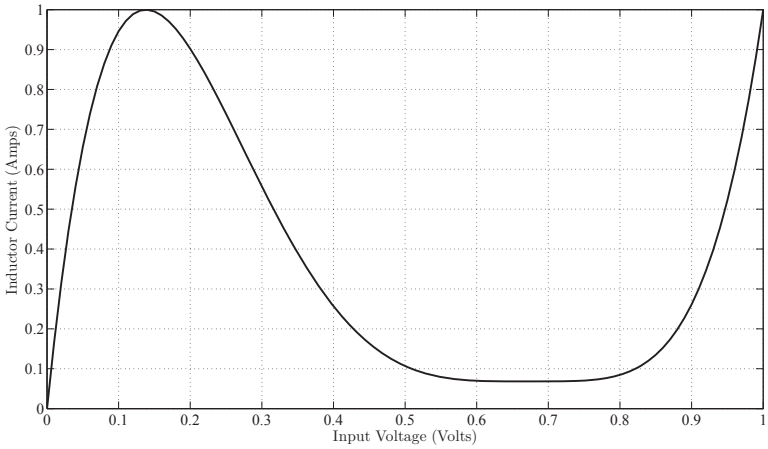


Figure 3.4: Negative Resistance Region [29].

The numerical model of the SDE in additive and multiplicative form, presented in Chapter 2 is simulated in a MATLAB environment and using $0.18\mu m$ process parameters. The process variation is considered for the resistor and the capacitor elements only. Due to lack of available process data, only the nominal values are assumed. However, if the

manufacturing variations are known for the inductors the methodology can be scaled to adopt those changes. As any underlying assumption on the distribution cannot be made for the initial condition (V_0), in all the cases it is considered to be a constant.

Property Observations

In general, for tunnel diode oscillation, the kind of properties we are interested to verify are: *"Is the system behavior the same for the set of initial condition?"* or *"For which set of parameters values, the circuit oscillates or dies?"* The properties that we verify are the oscillation and no oscillation for different *0.18μm* process corners shown in Table 3.1.

Table 3.1: Tunnel Diode Oscillator Parameters for Property 1

Parameter	Slow Process Corner	Nominal Process Corner	Fast Process Corner
Sheet Resistance(R_{sh})$\Omega/\square$	6.715	6.32	5.925
Resistance (R)Ω	0.425	0.4	0.375
Inductor (L) H	1e-6	1e-6	1e-6
Capacitor (C) F	1200e-12	1000e-12	800e-12
V_0 Volts	0.131	0.131	0.131
I_0 Amps	0.04e-3	0.04e-3	0.04e-3

Property 1: We verify that for the set of parameters given in Table 3.1, there is no oscillatory behavior. The behavior in question is stated as the bounded safety property, meaning for no oscillation property to be satisfied, if for the given simulation time step a certain threshold will not be reached then the property is violated thereby enabling a *violation signal*. The implementation of the assertion as a FSM for verification of *no oscillation* property is shown in Figure 3.5.

The FSM has five states namely, *initialization*, *cycling*, *violation* & *cycling, error and stop simulation*. The maximum simulation time, N_{max}, and inputs like initial voltage, current and output violation are set in the *initialization* state. As soon as the simulation starts,

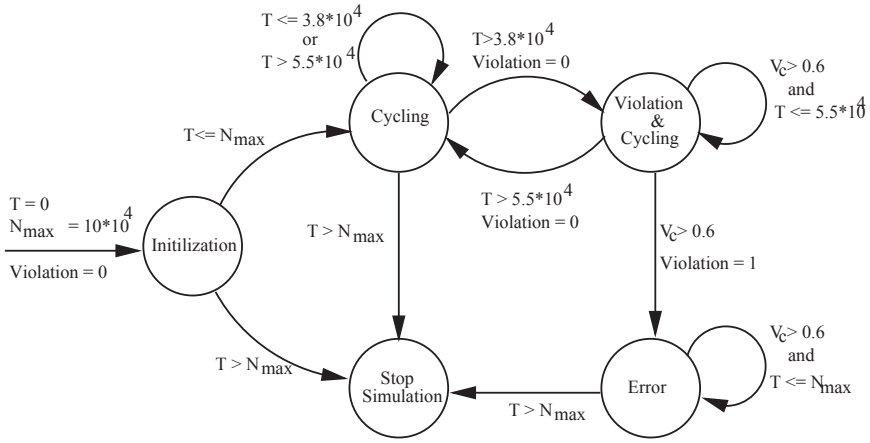


Figure 3.5: Property 1 FSM

the FSM goes to the *cycling* state and remains until $T < 3.8 * 10^4$ or $T > 5.5 * 10^4$, where the output voltage $V_c(t)$ is just reported and not observed for any violation. This is because, though the simulation is done from $T = 0$ to $T = N_{max}$, the *no oscillatory* property is verified for the bounded interval $T > 3.8 * 10^4$ to $T \leq 5.5 * 10^4$. As *T* becomes greater than $3.8 * 10^4$, the FSM goes into the *violation* & *cycling* state where the property is verified for any violation, meaning if $V_C(t) < 0.6$, the property is satisfied or else the violation signal is asserted and the FSM enters into the *error* state where it remains there till $T \leq N_{max}$, and then goes to the *stop simulation* state. The results for the verification of Property 1 is shown in Figure 3.6. The results are obtained by simulating the numerical approximation of the SDEs and the assertion using MATLAB. However, the more interesting question that has to be answered is *"For the given set of initial conditions and bounded region, how does the influence of noise and process variation affect the oscillatory behavior of the tunnel diode oscillator?"* meaning will the tunnel diode oscillator, which has been proved to be stable and non oscillating, produce the same stable result in the presence of noise?

We simulated the tunnel diode oscillator for three different process corners (*slow, nominal* and *fast*) as shown in Figure 3.6. The noise is modeled and simulated as a Wiener process as shown in the Figure 3.6 (a). From the simulation results, Figures 3.6 (b) and (c),

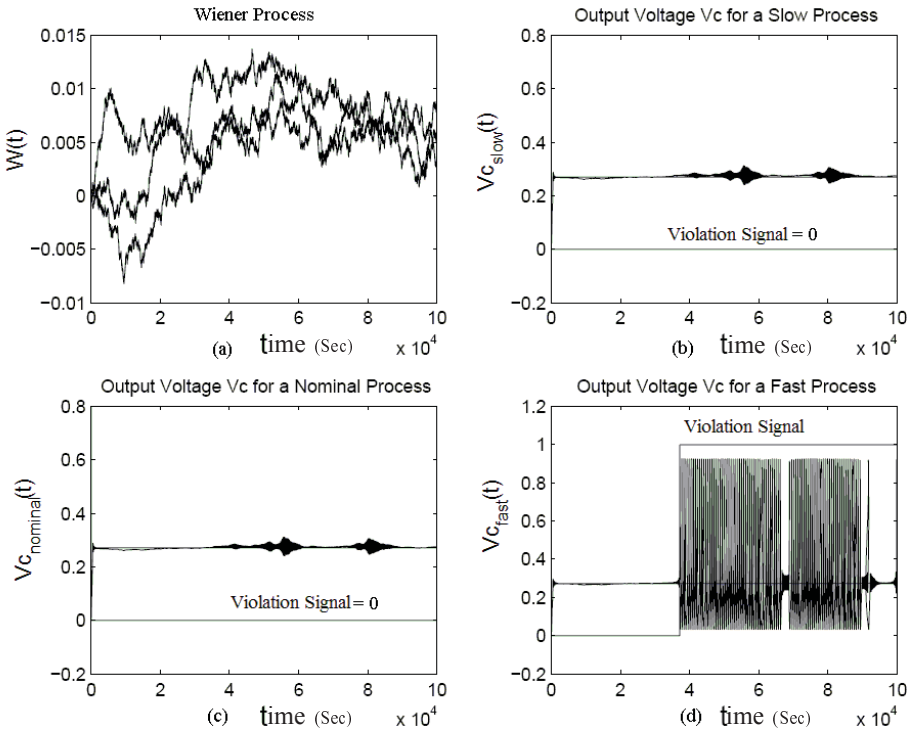


Figure 3.6: Property 1 Simulation Results

we note that for the given set of parameters, the property is satisfied for *slow* and *nominal* process corners. However, for the *fast* process corner and $T > 3.8 * 10^4$ (Figure 3.6 (d)) the output has a stable oscillation, thereby detecting a violation. The additive noise W_2 and W_3 along with the changes in resistor and capacitor due to process variation in the voltage equation $V_c(t)$ causes the tunnel diode oscillator circuit to move to negative resistance region, thereby creating oscillation.

In summary, for the given set of initial conditions and device parameters, though the authors in [57] have verified the *no oscillation* property in the absence of noise and process variation, we demonstrated that the property fail with noise and process variation.

Property 2: We verify that for the set of parameters and initial conditions given in Table 3.2, the tunnel diode produces a stable oscillation.

Table 3.2: Tunnel Diode Oscillator Parameters for Property 2

Parameter	Slow Process Corner	Nominal Process Corner	Fast Process Corner
Sheet Resistance(R_{sh})$\Omega/\square$	6.715	6.32	5.925
Resistance (R)Ω	0.17	0.16	0.15
Inductor (L) H	1e-6	1e-6	1e-6
Capacitor (C) F	1200e-12	1000e-12	800e-12
V_0 Volts	0.131	0.131	0.131
I_0 Amps	0.04e-3	0.04e-3	0.04e-3

The oscillation property can be understood as within the time interval [0, T] on every computation path, whenever the *Vc* amplitude will reach [0.9v, 1.0v], it will reach this value again until the simulation stops. We show that within a bounded region, we prove whether the oscillation dies in the presence of noise, meaning, no oscillatory behavior, even though in the noiseless model it was proved to oscillate [57]. The implementation of the assertion as an FSM for verifying the absence of oscillation is shown in Figure 3.7. The details follow

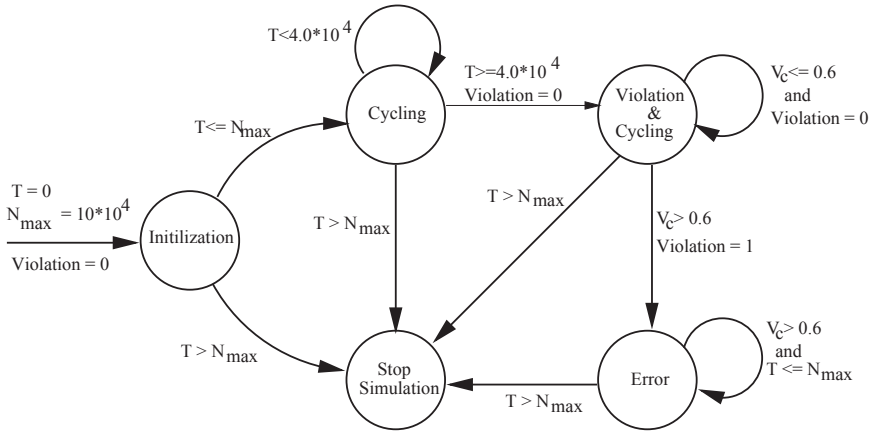


Figure 3.7: Property 2 FSM

exactly like in Property 1 except that the bounded region for verification of *no oscillatory* behavior is between $T \geq 4.0 * 10^4$ until *T*=N_{max}. The simulation results for the verification of Property 2 are shown in Figure 3.8. The dotted line represents the output oscillation in the absence of noise, while the bold line represents the output oscillation in the presence of noise and process variation.

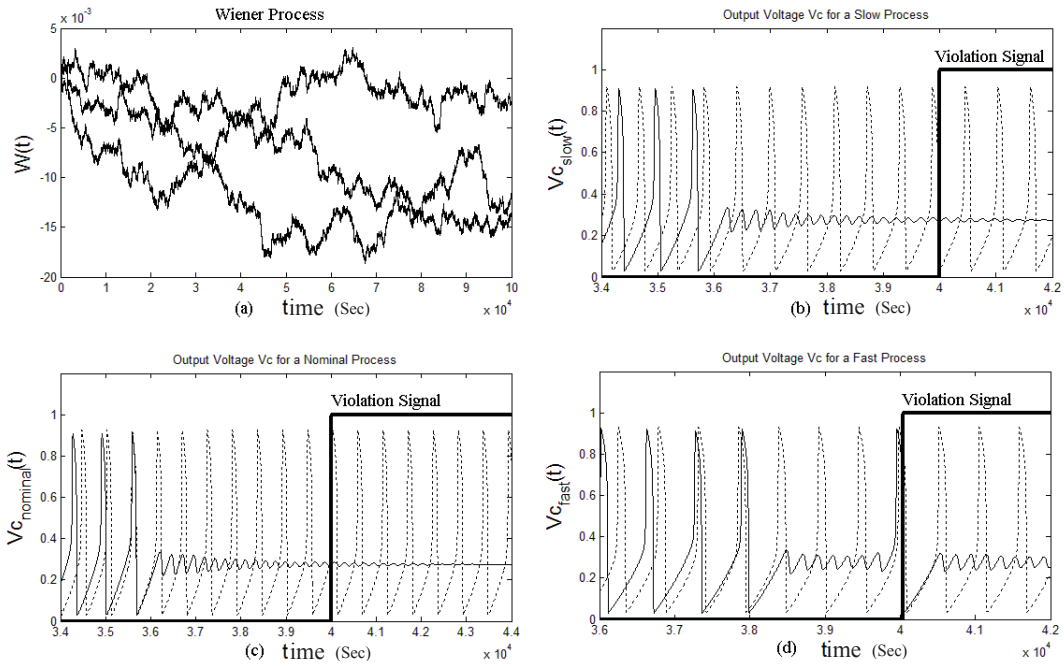


Figure 3.8: Property 2 Simulation Results

From the simulation results, we notice that the tunnel diode produces a stable oscillation in the absence of noise. However, in the bounded region from $T \geq 4.0 * 10^4$ until $T = 10.0 * 10^4$, the oscillatory behavior dies out in the presence of noise for all the process corners, thereby detecting a violation as shown in Figures 3.8(b), (c) and (d). This shows that the noise and process variation has an adverse effect on the performance of the design under verification. Moreover, we demonstrated that the oscillatory behavior which has been proved in [57] does not hold under noisy and process variation conditions, thereby making our methodology robust in detecting errors.

3.3.2 Colpitts Oscillator

The circuit diagram for a MOS transistor based Colpitts oscillator [69] is shown in Figure 3.9 (a) with the small-signal shown in Figure 3.9 (b). For the correct choice of component values the circuit will oscillate. This is due to the bias current and negative resistance of the

passive tank.

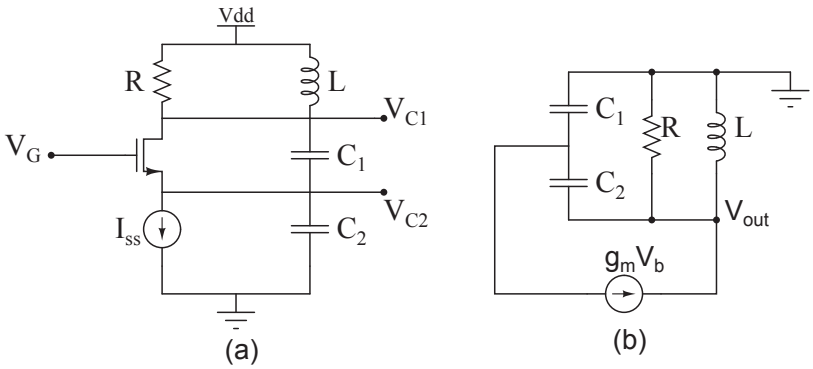


Figure 3.9: Colpitts Oscillator

The simplified system of equations that describe the behavior of the Colpitts oscillator is given by [69]:

$$\begin{aligned} \dot{V}_{C1} &= \frac{1.2-(V_{C1}+V_{C2})}{RC} + \frac{I_L}{C} - \frac{I_{ds}}{C} \\ \dot{V}_{C2} &= \frac{1.2-(V_{C1}+V_{C2})}{RC} + \frac{I_L}{C} - \frac{I_{ss}}{C} \\ \dot{I}_L &= \frac{1.2-(V_{C1}+V_{C2})}{L} \end{aligned} \tag{3.1}$$

where, for $V = V_{C1} + V_{C2}$

$$I_{ds} = \begin{cases} 0 \text{ if } V_{C2} > 0.3 \\ K\frac{W}{L}((0.3-V_{C2})(V_{C1}) - 0.5(V_{C1})^2) \text{ if } V < 0.3 \\ K\frac{W}{L}(0.3-V_{C2})^2 \text{ if } V \geq 0.3 \end{cases}$$

If thermal noise is considered for the passive components and shot noise for the MOS transistor, then Equation (3.1) can be extended to SDE form as given below

$$\begin{aligned}
\dot{V}_{C1} &= \frac{1.2-(V_{C1}+V_{C2})}{RC} + \frac{I_L}{C} - \frac{I_{ds}}{C} + \sum_{k=1}^{2} \alpha\xi_k(t) \\
\dot{V}_{C2} &= \frac{1.2-(V_{C1}+V_{C2})}{RC} + \frac{I_L}{C} - \frac{I_{ss}}{C} + \sum_{k=1}^{2} \alpha\xi_k(t) \\
\dot{I}_L &= \frac{1.2-(V_{C1}+V_{C2})}{L} + \alpha\xi_3(t) + \zeta(t)
\end{aligned} \tag{3.2}$$

where $\sum_{k=1}^{3} \alpha\xi_k$ represents the thermal noise model for the passive elements with certain amplitude α and $\zeta(t)$ represents poisson white shot noise (PWSN) due to random carrier motion (current) in the MOS transistor.

The above SDE model is numerically approximated using Euler/Milstein technique and simulated with process variation in a MATLAB simulation environment.

Property Observations

The property that we are interested in analyzing is *"Whether for the given parameters and initial conditions (Iss, Vdd, transconductance) the circuit will oscillate?"* The simulation results in Figure 3.11 show the variation of output voltages Vc_1 and Vc_2 with and without noise. The property that is verified is the no oscillation for different circuit parameters shown in Table 3.3. The behavior in question is stated as the bounded safety property, meaning for the given simulation time step oscillation will not occurs if the current cannot exceed a certain threshold.

For the no oscillation property to be satisfied, the current through the inductor I_L should be bounded within $[-0.004, 0.004]$. If verified to true, the property is satisfied else a violation signal is enabled. The implementation of the assertion as an FSM for verification of no oscillation property is shown in Figure 3.10.

Table 3.3: Colpitts Oscillator Parameters

Parameter	Slow Process Corner	Nominal Process Corner	Fast Process Corner
Sheet Resistance(R_{sh})$\Omega/\square$	6.715	6.32	5.925
Resistance (R)Ω	408	384	360
Inductor (L) H	3e-6	3e-6	3e-6
Capacitor ($C_1 = C_2 = C$) F	24e-12	20e-12	16e-12
Transconductance ($\frac{mA}{V^2}$	0.0067	10.0	0.0133
V_{dd} Volts	1.2	1.2	1.2
I_{SS} Amps	100e-6	100e-6	100e-6

The FSM has four states namely, *initialization*, *cycling*, *error* and *stop simulation*. The maximum simulation time, N_{max}, and output violation are set in the *initialization* state. As soon as the simulation starts, the FSM goes to the *cycling* state and remains until $T \leq N_{max}$ and there are no violations observed. If the inductor current crosses the bounded threshold, the FSM asserts the *violation* signal and goes into the $error$ state where it remains there till $T \leq N_{max}$ and then goes to the *stop simulation* state.

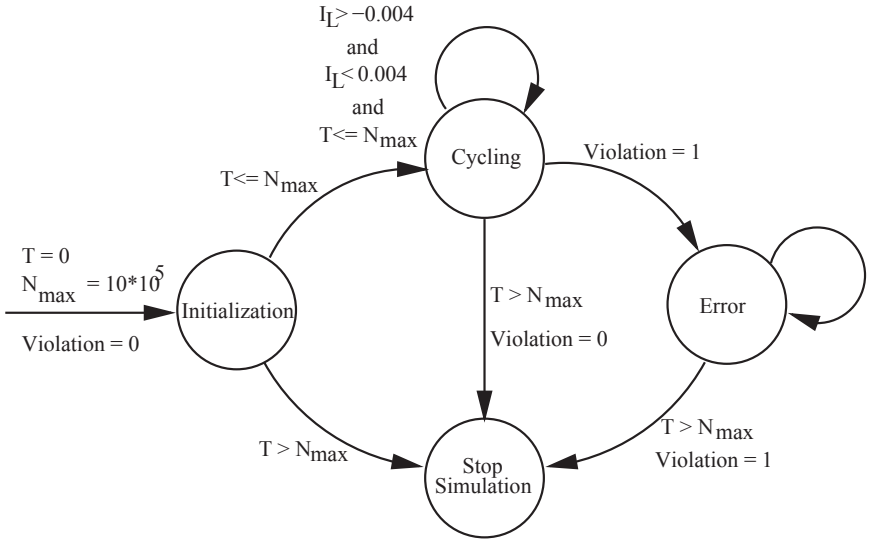
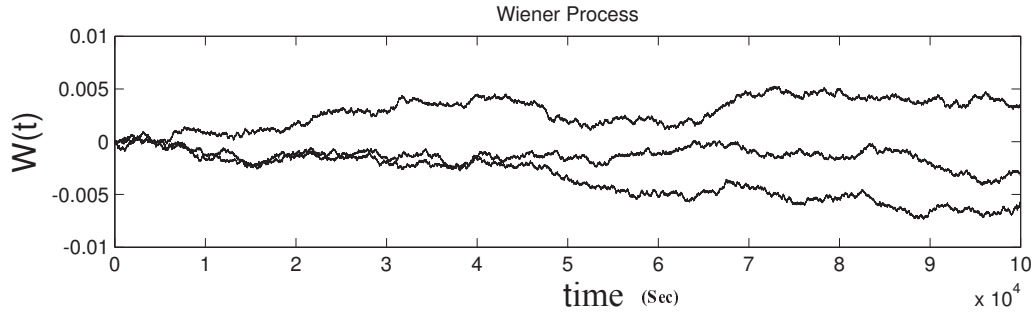


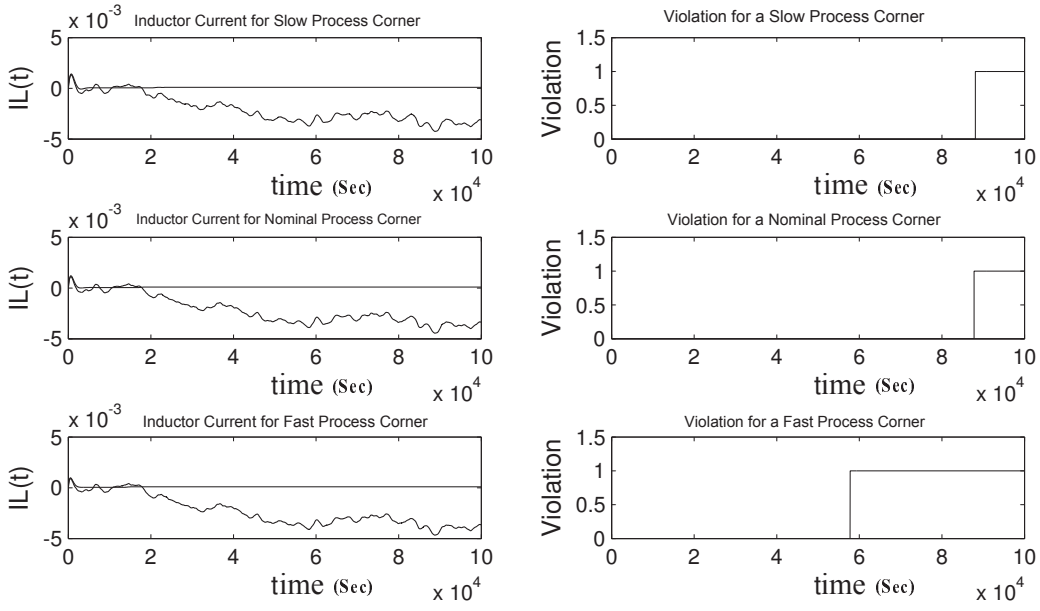
Figure 3.10: No Oscillation Property FSM

From the simulation results, we notice that the Colpitts oscillator does not oscillate in the absence of noise (solid line). However, for the *slow, nominal* and *fast* process corners in

the bounded region from T=$5.8 * 10^4$ until T=$10.0 * 10^4$, the variation in device parameter and additive noise in the inductive current equation has caused an increase in the inductive current, thereby detecting violation at T=$9.0 * 10^4$, T=$8.9 * 10^4$ and T=$5.9 * 10^4$ as shown in Figure 3.11 (b), respectively. This shows that the noise and process variation has an adverse effect on the performance of the design under verification. The simulation result does not mean that the Colpitts oscillator is oscillating but, shows that the inductor current is large enough to trigger other circuits when connected to a bigger designs.



(a)



(b)

Figure 3.11: Simulation Result of Colpitts Oscillator

3.3.3 PLL Based Frequency Synthesizer

A PLL based frequency synthesizer [21] is a classical AMS design that is commonly seen in communication systems for clock generation and recovery. Figure 3.12 shows a typical PLL based frequency synthesizer. It is composed of two comparators, a phase/frequency detector, charge pump, analog filter, voltage controlled oscillator (VCO) and a divider. Based on the input reference signal, the phase detector compares the reference signal, injected to the loop, to the VCO's output and produces a signal which varies in proportion to the difference in their phases. This output passes through a low pass filter to be used as a control signal to drive the voltage controlled oscillator. Thereafter, the VCO will lock to the reference signal, thereby producing a periodic signal.

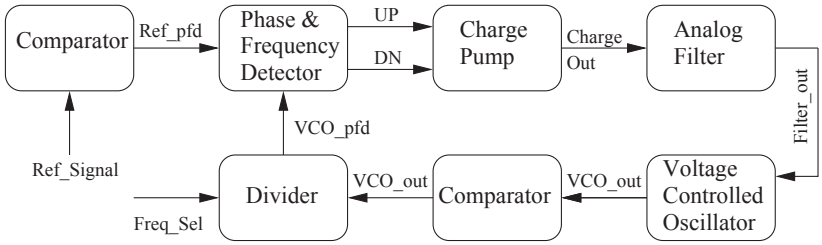


Figure 3.12: PLL Based Frequency Synthesizer

The reference signal (*Ref_Signal*) at the input is a simple sinusoidal wave with frequency ω_0. The VCO output (*VCO_out*) is a cosine signal with frequency *N+1* times of the reference frequency, where *N* is determined by the frequency select signal (*Freq_Sel*). If the *Freq_Sel* is '0', then the frequency of the reference input and VCO output will be the same or else the frequency will be divided accordingly based on the divider.

For this application, digital blocks are implemented as a difference equation [68] in a MATLAB simulation environment. For continuous time components the formulation is based on the semi-automatic generation of recurrence equation models as described in [11]. The behavior of the low-pass filter is modeled as an ODE and the SDE representation of its

noisy (additive and multiplicative) behavior can be described as

$$\dot{F}_o = \frac{1}{RC}(CP_o(t) - F_o(t)) + \sum_{k=1}^{2} \alpha\xi_k(t)$$
$$\dot{F}_o = \frac{1}{RC}(CP_o(t) - F_o(t)) + \alpha\xi_k(t)F_o(t) \tag{3.3}$$

where, F_o and CP_o represents the filter and charge-pump output, respectively, R and C represents the resistor and capacitor component in the filter circuit. The next step is to apply the Euler/Milstein scheme described in Chapter 2 to generate the following numerical model:

$$F_{O_{n+1}} = F_{O_n} + \left(\frac{\Delta_n}{RC}\right)(CP_o(n) - F_o(n)) + \alpha\Delta W_{sn}$$
$$F_{O_{n+1}} = F_{O_n} + \left(\frac{\Delta_n}{RC}\right)(CP_o(n) - F_o(n)) + \alpha\Delta W_n + \tag{3.4}$$
$$\frac{1}{2}\left((\Delta W_n)^2 - \Delta_n\right)F_{O_n}$$

where, $F_o(n)$ and $CP_o(n)$ are the discrete representation of the filter and charge-pump output, respectively,

Property Observations

A critical property of a frequency synthesizer is the "lock-time", meaning, if the *Freq_Sel* is activated, the PLL will lock at the desired frequency within a certain time as identified in the specification. The verification challenge is *"Will the above property hold true in the presence of noise and process variation?"*

The first step is to model the PLL and simulate it in a MATLAB simulation environment. The simulation result at the VCO output is shown in Figure 3.13. The dotted/bold waveforms represent the VCO output without noise and with noise, process variation, respectively.

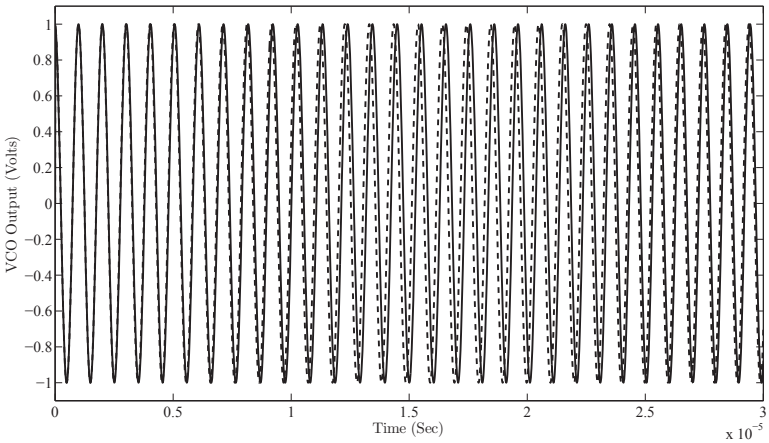


Figure 3.13: VCO Output with/without Noise

Property 1:

The lock-time is considered a safety property and is measured by the changes to the filter output with respect to the $freq\_sel$ signal. The implementation of the assertion as an FSM for verification of no oscillation property is shown in Figure 3.14.

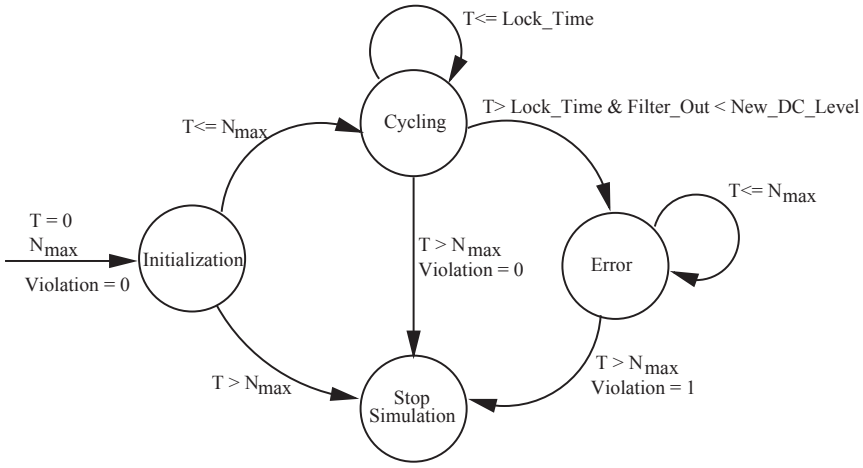


Figure 3.14: Lock-Time Property FSM

The FSM has four states namely, *initialization*, *cycling*, *error and stop simulation*. The maximum simulation time, N_{max}, and output violation are set in the *initialization* state. As soon as the simulation starts, the FSM goes to the *cycling* state and remains until $T \leq N_{max}$ and there are no violations observed. If the PLL fails to lock with 0.001 sec as given

in the specification, the FSM asserts the *violation* signal and goes into the $error$ state where it remains there till $T \leq N_{max}$ and then goes to the *stop simulation* state. The snapshot to verify the above property can be described as

```
while (freq_sel[i]==1 && freq_sel[i-1]==0)
 for n = 1: Nmax
   if (filter_out[n]== New_DC_Level && T_sample*(n-i)
       <=Lock_time) then
       Violation = 0; // Property is Satisfied
   else
       Violation = '1'; // Property is Violated
   end if
 end for
end while
```

The simulation results for Property1 are shown in Figure 3.15. It is interesting to note that the authors in [136] have shown that the lock-time property is satisfied. However, that is not the case when noise and process variation are considered. It is obvious that there will some effect of noise and the filter output will not stable. To accommodate this, the assertion is modified to check if the New_DC_Level falls within certain range p. In this case, for $p = \pm 1\%$, the filter output exceeds the threshold level. However, by increasing p, the property will be satisfied, but at the cost of accuracy. Choosing the value of p will depend on the type of application and designers choice.

It should be noted that the property fails because of the additive noise at the filter output and process variation in the resistor and the capacitor elements. Since, noise and process variation are random quantities, the assertion in the above case has failed due to high level of noise and process conditions. In reality, one may not come across such a level

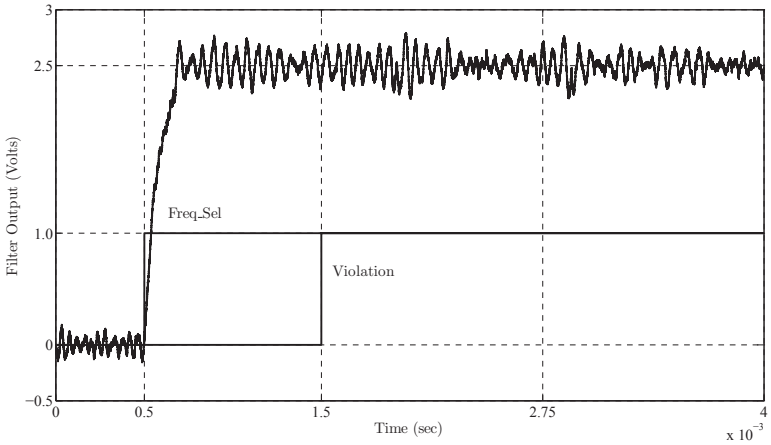


Figure 3.15: Lock-Time Property Results

of noise and manufacturing conditions.

To check the consistency of MATLAB based SDE analysis with the circuit level noise analysis, we conducted an experiment for a RC low-pass filter as shown in Figure 3.16. Here $R = 1\Omega$ and $C = 1F$ are the resistor and the capacitor of the filter circuit, respectively. *X(t)* is the white noise source representing the thermal noise in the resistor.

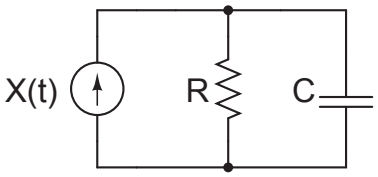


Figure 3.16: Lock-Time Property Results

Table 3.4: RC- Low Pass Filter

Noise Quantity	**Circuit Simulator**	**MATLAB SDE Analysis**
Average Power (P_{avg}) $watts$	0.5	0.57
Noise Bandwidth Hz	0.25	0.29

The simulation results are summarized in Table 3.4. It can be noted that the SDE analysis results at a higher-level of abstraction does not match with the circuit-level simulation results. The difference is due to white noise source generator, as each simulator

tend to use different pseudo-random number generator. Therefore, the average noise power generated at a given time will be different.

3.4 Summary

In this chapter, we have presented a practical assertion based verification methodology for noise and process variation in analog designs. The approach is based on modeling the noise using SDEs and numerically simulating, in MATLAB, the model with a given fabrication process parameter variations, and monitor the property of interest in an online fashion. We have used the proposed methodology to verify the oscillatory behavior of a Tunnel diode and a Colpitts oscillator circuits and the "lock-time" property of a PLL based frequency synthesizer. We showed that the properties that are satisfied without noise, have failed in the presence of noise and process variation, thereby proving that the proposed verification environment is efficient in finding bugs. This process is much more reliable than manual (visual or textual) inspection of simulation traces which will cost lots of time.

The above simulation results were derived for one particular set of Wiener process and for $0.18\mu m$ process technology. The FSM for verifying the property of interest is constructed using *if-then-else* MATLAB constructs. The methodology could be easily extended for other technologies by calculating device parameters based on process variation for different process corners or using probability distribution. The values of the Wiener process depend on the random number generator of the system and so we may find different sets of W_1, W_2 and W_3 during each simulation run. Therefore we conclude that, for this particular set of parameter values of W_1, W_2 and W_3 and initial conditions, the properties in the Tunnel diode, Colpitts oscillators and the PLL are violated, but, one can get a different set of values for the Wiener processes for which the property holds. Hence, the verification has to be done for multiple trajectories before concluding the correctness of the design.

The downside of assertion based approach is that the design is evaluated for one simulation trace. So, the question is *"What happens if we don't detect any violation on that trace?"*. *"Does that mean the circuit will work under all conditions?"*. In general, simulation based approach cannot provide a complete coverage on the design evaluation. However, the verification method can incorporate additional constraints to improve the confidence level of the design. This is very important while dealing with process variation and noise, as the design may encounter various operating conditions. Hence, to gain high confidence in the circuit, the design has to be evaluated statistically through multiple simulation and with different noise and process conditions. In the end, the acceptance/rejection of a circuit has to be measured from all simulation samples that may involve hypothesis testing and probabilistic measures. Therefore, we propose in the next chapter, a statistical based run-time verification environment to ensure the correctness of an analog circuit with certain confidence interval. This quantitative approach will help designers to assess the circuit with certain confidence at a higher level of abstraction and can facilitate system-level verification.

Chapter 4

Quantitative Analysis using Statistical Techniques

This chapter presents a framework for the statistical analysis of analog circuits. The framework allows us to model and verify the statistical property of an analog designs in the presence of shot noise, thermal noise, and process variations. The idea is to use stochastic differential equations (SDE) to model noise in additive and multiplicative form and then combine process variation in a statistical runtime verification environment. To illustrate the practical effectiveness of the proposed framework, the efficiency of MonteCarlo and Bootstrap statistical techniques are compared for Colpitts oscillator, Band-Gap reference generator and a Phase Locked Loop (PLL) based frequency synthesizer circuit.

4.1 Introduction

Assertion based method discussed in Chapter 3 is a powerful mechanism to verify the functional properties of the design without any uncertainties. Due to the stochastic nature of the noise, the assertion based verification technique cannot provide a greater insight to gain

confidence on the circuit. This is because, assertion based verification results are derived by simulating the design for a single trace and then looking for any violation.

To address the above problem, we propose in this chapter a run-time verification framework for monitoring the statistical property of an analog circuit. Statistical run-time verification combines hypothesis testing [96] and MonteCarlo/Bootstrap simulation for monitoring the statistical behavior in an analog circuit. The framework is developed to handle uncertainties in an analog design due to noise and process variation.

4.2 Statistical Verification Methodology

Figure 4.1 shows the overall statistical run-time verification methodology. Thereafter, given an analog design described as a system of *ODEs*, the idea is to generate SDEs that express the noise behavior. For the case of the circuits that do not have closed form solution, the approach is to numerically approximate the SDE's based on Euler-Maruyama technique as described in Chapter 2. For process variation, technology vendors create a library of devices with different corners [30] that characterize the device in terms of power, speed, etc. This allows designers to choose from a range of devices based on the application and requirements. For a 0.18μm process, different circuit parameters are derived using Gaussian distribution with a known $\pm 3\sigma$ deviation.

For environment constraints, this may include the amplitude of the noise, initial conditions of the circuit current and voltages. The SDE model, process variation, and the environment constraints are evaluated using MonteCarlo/Bootstrap statistical technique in a MATLAB simulation environment.

Statistical run-time verification combines hypothesis testing and resampling methods for monitoring the statistical behavior in an analog circuit. The basic idea behind the resampling methods is to simulate the SDE model and sample them in order to calculate the

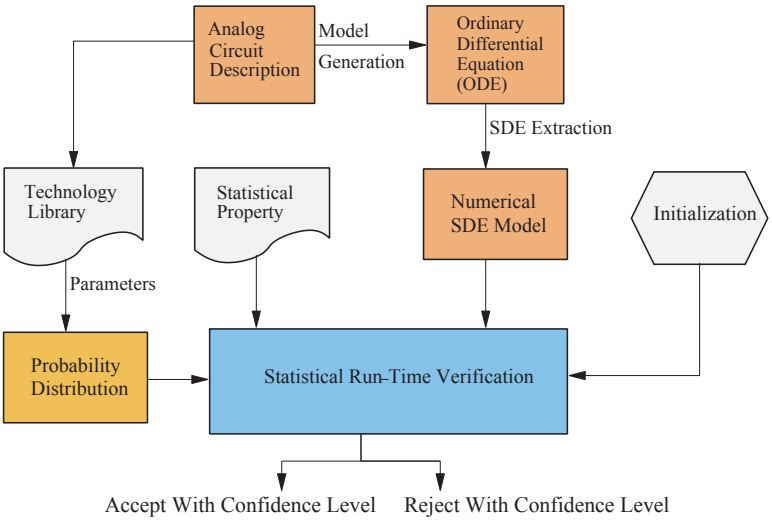


Figure 4.1: Statistical Run-Time Verification Methodology

desired statistics for a given confidence level δ.

Figure 4.2 shows the methodology for statistical simulation procedure based on hypothesis testing. The statistical property, is expressed as a null hypothesis H_0, while the alternative hypothesis H_1 becomes the counterexample naturally. For the given numerical SDE model and the specified tail test, MonteCarlo or Bootstrap monitoring is carried out based on the given confidence level δ and the calculated significance level α. The statistical property is verified if the null hypothesis H_0 is accepted, else, the monitor reports the violation of the property. In all cases, an error margin ϵ is generated as shown in Figure 4.2.

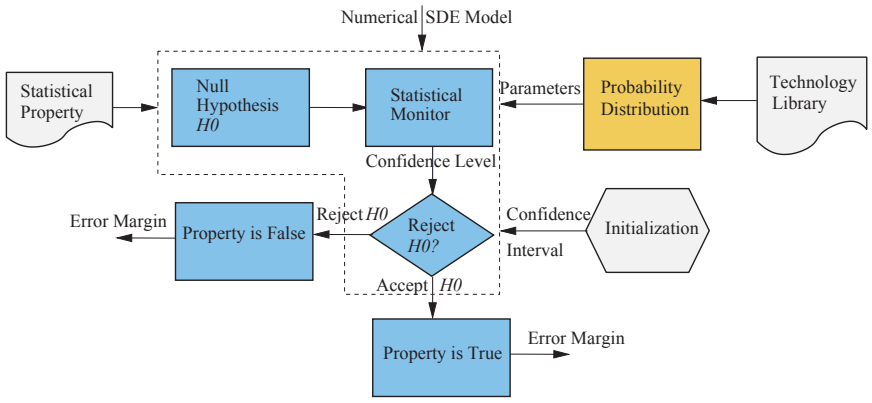


Figure 4.2: Statistical Hypothesis Testing

4.2.1 MonteCarlo Algorithm

The MonteCarlo method refers to a technique of solving problems using random variables. It is widely used to investigate statistical problems such as inference statistics of a given population of interest. The basic idea behind the MonteCarlo method is to sample the given population model for M trials and then calculate the desired statistics (such as mean, median, variance, skewness, etc.). To apply MonteCarlo based hypothesis testing, it is necessary that the distribution of the sampling population is known in advance [85].

One of the most important components of MonteCarlo simulation is the use of deterministic algorithm to generate a normally distributed unbiased pseudo random number. These random numbers are then used to sample the true population of interest, in this case the analog circuit output. In general, there is no theory that governs the number of trials in MonteCarlo simulation. However, a trade off exists between those numbers and the simulation run-times. The higher confidence can be gained by choosing a larger number of trials, but at the cost of run-times [96].

The detailed procedure for Monte-Carlo hypothesis testing for an analog circuit is illustrated in Algorithm 4.1, where $output\_vector$ denotes the observed output of an analog circuit with noise and process variation. M represents the number of MonteCarlo trials, α a chosen significant level and $type\_test$ represents the type of test to be performed (*upper*, *lower*, or *two-tailed*).

The initialization steps (lines 1-4) are followed by the computation of the standard score to determine the observed analog output T_{obs} (loop between lines 5 and 9). This calculation is done with certain standard error margin as defined to be

$$E = \frac{\sum_{i=1}^{n}(x_i - \bar{x})^2}{N(N-1)} \tag{4.1}$$

where $X = (x_1, x_2, ..., x_N)$ represents the MonteCarlo sample, μ is the mean $\bar{x}$ the pseudo

Algorithm 4.1 MonteCarlo Based Hypothesis Testing:

Require: $output\_vector$, M, α, $type\_test$

$V \leftarrow output\_vector$
$N \leftarrow length(V)$
$mu \leftarrow mean(V)$
$sig \leftarrow standard\_error(V)$
for $i \leftarrow 1$ *to* M **do**
  $r \leftarrow random\_number\_generator(N)$
  $MC_{sample} \leftarrow sig * r + mu$
  $T_{obs}(i) \leftarrow \frac{mean(MC\_sample)-mu}{sig}$
end for
while $type\_test =$ "$upper\ tail\ test$" **do**
  $critical\_value = quantile(1-\alpha)$
  if $critical\_value \geq T_{obs}$ **then**
    $Accept\ H_0$
  else
    $Reject\ H_0$
  end if
end while
while $type\_test =$ "$lower\ tail\ test$" **do**
  $critical\_value = quantile(\alpha)$
  if $critical\_value \leq T_{obs}$ **then**
    $Accept\ H_0$
  else
    $Reject\ H_0$
  end if
end while
while $type\_test =$ "$two\ tail\ test$" **do**
  $critical\_value\_low = quantile(\frac{\alpha}{2})$
  $critical\_value\_up = quantile(\frac{1-\alpha}{2})$
  if $critical\_value\_up \leq T_{obs} \parallel critical\_value\_low \geq T_{obs}$ **then**
    $Reject\ H_0$
  else
    $Accept\ H_0$
  end if
end while

random sample mean, and E defines the standard error of the population under the hypothesis that H_0 is true. The next step is to compute the critical value in order to specify the rejection region (alternative hypothesis). Depending on the type of the test, the *quantile* procedure [96] (lines 11,19, 27 and 28) can be used to determine the critical value. With the estimated critical value, the rejection region under the assumption of H_0 being true can be determined for each of the tail test as defined in Table 4.1.

Table 4.1: Rejection Region for Different Tail Test

Tail Test	**Rejection Region**
Upper	$[(100 \times (1-\alpha))\%, +\infty]$
Lower	$[-\infty, (100 \times \alpha)\%]$
Two-Tailed	$[-\infty, (100 \times \frac{\alpha}{2})\%] \cup [(100 \times (1-\frac{\alpha}{2}))\%, +\infty]$

If the observed value T_{obs} is greater than the critical value, the null hypothesis H_0 is rejected as described in Algorithm 4.1 (lines 10-32). One of the drawbacks of the MonteCarlo simulation is that the assumption on the distribution which leads directly to the question *"What will happen when the assumption of the underlying distribution is violated or unknown?"* In such cases, techniques such as the Bootstrap will provide a perfect platform to reason about the statistical inference of the population.

4.2.2 Bootstrap Algorithm

The bootstrap is a general purpose method for estimating the statistical property without making any assumptions about the underlying distribution of the population [31]. In this sense, it is considered as a non parametric technique or distribution free. The basic idea behind the bootstrap technique can be described as follows [31]: *"Given a random sample of N data $X = (x_1, x_2, ..., x_N)$ from an unspecified distribution F, the maximum likelihood*

estimator of F is the distribution that puts an equal point probability of $\frac{1}{N}$ to each data of X".

The detailed procedure for Bootstrap based hypothesis testing for an analog circuit is illustrated in Algorithm 4.2, where $output\_vector$ denotes the observed output of an analog circuit with noise and process variation. B represents the number of bootstrap samples, α a chosen significant level and $type\_test$ represents the type of test to be performed (*upper*, *lower*, or *two-tailed*).

The first step is to draw randomly B samples with replacement from the simulated circuit output of size N (line 4). This is followed by test statistic estimation for each bootstrap replication in order to measure discrepancy between the data and H_0. The results are then in T_{boot} as a vector (line 5). The *quantile* procedure is then used to compute the critical value by type of test:

- The $1 - \alpha$ quantile of the empirical distribution for an upper tail test as shown in line 9.
- The α quantile of the empirical distribution for a lower tail test as mentioned in line 17.
- The $\frac{\alpha}{2}$ and $(1 - \frac{\alpha}{2})$ quantile of the empirical distribution for a two sided test as given in lines 25 and 26.

Once the critical value is determined, a decision regarding the violation of a statistical property is done using hypothesis testing (lines 16-31). For instance, in the case of a lower tail test (lines 16-23), if the observed value T_{boot} is lower than the computed critical value than we reject H_0, meaning the statistical property has failed.

Algorithm 4.2 Bootstrap Based Hypothesis Testing:

Require: $output\_vector$, B, α, $type\_test$

$V \leftarrow output\_vector$
$N \leftarrow length(V)$
for $i \leftarrow 1$ *to* B **do**
  $rep \leftarrow Resample\_Bootstrap(V, N)$
  $T_{boot}(i) \leftarrow Compute\_test\_statistic(rep)$
end for
$T_{sorted} \leftarrow sort\_ascending\_order(T_{boot})$
while $type\_test =$ "$upper\ tail\ test$" **do**
  $critical\_value = T_{sorted}(B * (1 - \alpha))$
  if $critical\_value \geq T_{obs}$ **then**
    $Accept\ H_0$
  else
    $Reject\ H_0$
  end if
end while
while $type\_test =$ "$lower\ tail\ test$" **do**
  $critical\_value = T_{sorted}(B * \alpha)$
  if $critical\_value \leq T_{obs}$ **then**
    $Accept\ H_0$
  else
    $Reject\ H_0$
  end if
end while
while $type\_test =$ "$two\ tail\ test$" **do**
  $critical\_value\_low = T_{sorted}(B * \frac{\alpha}{2})$
  $critical\_value\_up = T_{sorted}(B * \frac{1-\alpha}{2})$
  if $critical\_value\_up \leq T_{obs} \parallel critical\_value\_low \geq T_{obs}$ **then**
    $Reject\ H_0$
  else
    $Accept\ H_0$
  end if
end while

4.3 Applications

To illustrate the efficiency of the proposed methodology, the approach is illustrated on a tunnel diode, Colpitts oscillator and PLL circuits. The effect of thermal noise on passive components (Resistor, Capacitor and Inductor) and shot noise on the transistors has been analyzed in additive and multiplicative form. The first step in noise analysis is to identify and incorporate the sources of noise as a stochastic process in the form of SDE. Thermal and Shot noise are defined based on the method described in Chapter 2. The experiment results are derived separately for additive and multiplicative noise in a statistical based MATLAB simulation environment on a Windows Vista OS (AMD Dual-Core, 4GB RAM) machine. Unlike assertion based verification method discussed in Chapter 3 where the process conditions are considered for only three corners (*slow, nominal* and *fast*), for statistical verification, based on MonteCarlo/Bootstrap trials, many independent circuit parameters are generated using Gaussian distribution.

4.3.1 Colpitts Oscillator

The circuit diagram for a MOS transistor based Colpitts oscillator is shown in Figure 3.9 (a). For the correct choice of component values, the circuit will oscillate due to the bias current and negative resistance of the passive tank. The frequency of oscillation is determined by L, C_1 and C_2.

The SDE model presented in Chapter 3 is numerically approximated using Euler/Milstein technique and simulated with process variation in a MATLAB simulation environment. The deterministic property that was verified is *"Whether for the given parameters and initial conditions, the inductor current is within a certain bound or not for oscillation?"* The simulation results in the Figure 4.3 show the variation of inductor current $I_L(t)$ with (bold line) and without noise (dotted line).

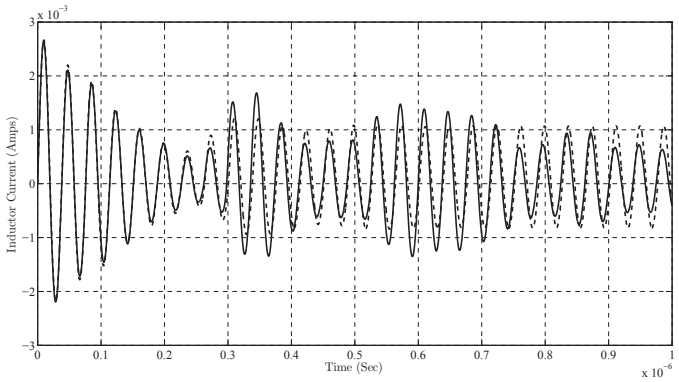


Figure 4.3: Simulation Result of Colpitts Oscillator

Statistical Property Observation

For statistical run-time verification one would be interested to know *"Whether for the given confidence level* α*, process variation and* M/B *MonteCarlo/Bootstrap trials, what is the probability of acceptance and rejection of oscillation for multiple trajectories* $Trac$*?"* where $Trac$ represents different kind of the same analog circuit, but with different noise and process conditions. For the oscillator, the current through the inductor I_L should be bounded within $[-0.004, 0.004]$. As a result, the null hypothesis H_0 and the alternative hypothesis H_1 of this property can be expressed as:

$$\begin{aligned} H_0 &: -0.004 \leq I_L \leq 0.004; \\ H_1 &: I_L > 0.004 \parallel I_L < -0.004; \end{aligned} \tag{4.2}$$

Both the MonteCarlo and Bootstrap experiments were conducted for the confidence level $\alpha = 0.05$ for different tail tests, with shot/thermal noise in the circuit elements, and with the circuit parameter generation using a normally distributed process variation model. The results are summarized in Table 4.2.

From Table 4.2, it can be noted that, irrespective of the tail test, that the MonteCarlo technique exhibits false violation (column 3). In the MonteCarlo method, first the mean of the output is derived, followed by the creation of different sampling points based on normal

Table 4.2: Statistical Runtime Verification Results for Colpitts Oscillator.

Additive Noise (TRAC = 200, M = MonteCarlo Trials, B = Bootstrap Trials, P.V = Process Variation, A = Accept, R = Reject, $\alpha = 0.05$)																	
M=	**Tail**	**No Shot/Thermal Noise & P.V**				**Shot/Thermal Noise Only**				**P.V Only**				**Shot/Thermal Noise & P.V Only**			
		MonteCarlo		**Bootstrap**		**MonteCarlo**		**Bootstrap**		**MonteCarlo**		**Bootstrap**		**MonteCarlo**		**Bootstrap**	
B=	**Test**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**
1000	Lower	192	8	200	0	179	21	191	9	180	20	187	13	147	53	184	16
	Upper	194	6	200	0	173	27	194	6	178	22	189	11	153	47	182	18
	Two	190	10	200	0	169	31	189	11	175	25	181	19	138	62	180	20
10000	Lower	188	12	200	0	164	36	193	7	172	28	185	15	151	49	179	21
	Upper	187	13	200	0	163	37	194	6	180	20	187	13	143	57	184	16
	Two	183	17	200	0	164	36	191	9	175	25	181	19	132	68	179	21
50000	Lower	188	12	200	0	159	41	189	11	177	23	182	18	122	78	174	26
	Upper	187	13	200	0	161	39	188	12	174	26	181	19	127	73	179	21
	Two	181	19	200	0	159	41	183	17	171	29	179	21	120	80	173	27
Multiplicative Noise (TRAC = 200, M = MonteCarlo Trials, B = Bootstrap Trials)																	
1000	Lower	194	6	200	0	188	12	196	4	180	20	187	13	189	11	194	6
	Upper	197	3	200	0	190	10	193	6	178	22	189	11	185	15	196	4
	Two	193	7	200	0	185	15	194	6	175	25	181	19	177	23	189	11
10000	Lower	199	1	200	0	181	19	191	9	172	28	185	15	180	29	184	16
	Upper	197	3	200	0	183	17	193	7	180	20	187	13	177	23	181	19
	Two	194	6	200	0	188	12	196	4	175	25	181	19	179	21	184	16
50000	Lower	197	3	200	0	187	13	191	9	177	23	182	18	181	19	188	12
	Upper	197	3	200	0	182	18	193	7	174	26	181	19	171	29	185	15
	Two	195	5	200	0	186	14	191	9	182	18	187	13	164	36	181	19

distribution with a known standard deviation. Such a process may sometimes lead to a value that is out of bound with the observed value thereby, giving rise to false violation. It can also be seen that process variation (columns 11-14) in all the passive components has a greater effect on the acceptance/rejection of the circuit and with the combined effect of noise and process variation (columns 15-18) the hypothesis testing exhibited considerable failure of the statistical property.

The effect of process variation and noise on the statistical results can be visualized using shmoo plots as shown in Figure 4.4. Though the process variation is considered in all circuit elements, the figure is shown only for the process variation in capacitor with respect to the resistor. The capacitance values are generated based on Gaussian distribution as described in Chapter 3. The standard deviation in this case is 10% of the mean value as specified in the *0.18*μm technology library document [1]. At each capacitance value, the resistor is swept based on the values generated using normal distribution and the hypothesis testing result is analyzed by writing '1' for acceptance or '0' for rejection. It can be noted that higher values of capacitance, can make the oscillator stable, meaning non-oscillating.

In addition, the number of MonteCarlo trials has an adverse effect on the outcome of

the acceptance, but, a large Bootstrap trial have made little impact on the outcome. This is because, the data generation process for the Bootstrap does not assume any distribution of the output data. For this experiment, the worst case run-time for $M/B = 50000$ is around *5-6* hrs, which though is considerably less than the simulation done at the circuit level.

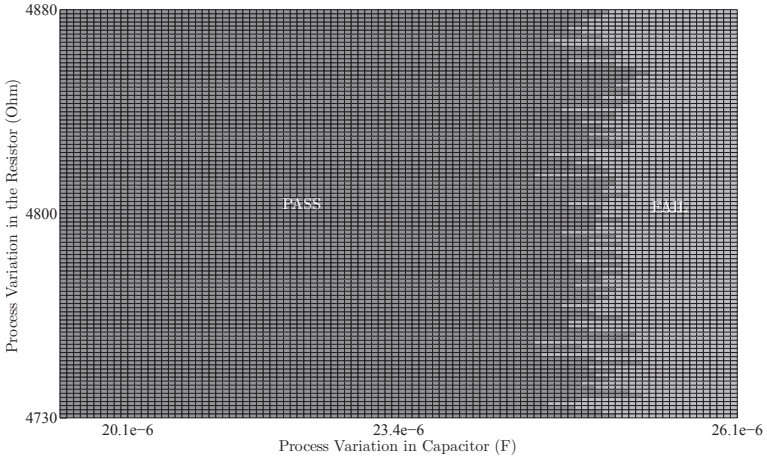


Figure 4.4: Shmoo Plotting of Colpitts Oscillator Results.

4.3.2 Band-Gap Reference Generator

For any biasing circuits, one of the most important performance issue is their dependence on temperature. The variation in temperature, noise and process variation attributes to the fractional change in the output voltage/current, thereby affecting the functionality of the design [52]. Figure 4.5 shows a BJT based reference generator biasing circuit, and the question is *"How does the variation of noise with respect to temperature affect the behavior of the circuit?"*

The output voltage is based on the summation of the voltage across base-emitter (V_{BE}) and the reference voltage (V_T). The behavior of the above circuit can be described as

$$\frac{dV_o}{dT} = (\gamma - \beta)\frac{V_T}{T}\left(\frac{T_0 - T}{T}\right) \tag{4.3}$$

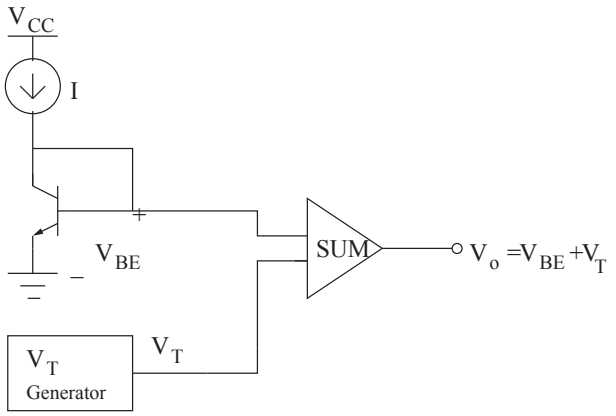


Figure 4.5: Band Gap Reference Circuit [52].

where V_T is the input voltage. If we consider a temperature varying shot noise process $\zeta(T)$ in the transistor, Equation (4.3), can be rewritten to incorporate randomness in additive and multiplicative form as:

$$\begin{aligned} \frac{dV_o}{dT} &= (\gamma - \beta)\frac{V_T}{T}\left(\frac{T_0 - T}{T}\right) + \zeta(T) \\ \frac{dV_o}{dT} &= (\gamma - \beta)\frac{V_T}{T}\left(\frac{T_0 - T}{T}\right) + \zeta(T)V_o(T) \end{aligned} \tag{4.4}$$

where γ and β are temperature independent constants [52] and T is the temperature. The shot noise process in Equation (4.4) is modeled as a PWSN numerical model with a white noise (Gaussian) distribution for the noise amplitude. For additive/multiplicative SDEs in the form of Equation (4.4), the SDE time discretization Euler/Milstein scheme with a step-size Δ_n is applied to generate the following numerical model:

$$V_{o_{n+1}} = V_{o_n} + (\gamma - \beta)\frac{V_T}{T}\left(\frac{T_0 - T}{T}\right)\Delta_n + \Delta W_{sn}$$

$$V_{o_{n+1}} = V_{o_n} + (\gamma - \beta)\frac{V_T}{T}\left(\frac{T_0 - T}{T}\right) + \Delta W_{sn} + \tag{4.5}$$

$$\frac{1}{2}\left((\Delta W_{sn})^2 - \Delta_n\right)V_{o_n}$$

In the statistical analysis presented for the band-gap reference generator, since manufacturing techniques for BJT are different from those of CMOS, the effect of process variation for BJT's are not considered. The effect of the variation in the input voltage (V_T) is also studied.

Statistical Property Observation

The property of interest is: *"Whether for the given set of parameters and variation in temperature T, will the output voltage V_o be greater than a certain threshold voltage?"* The analysis was done only for thermal noise in additive form and does not provide a statistical estimate to gain confidence in the circuit verification.

For statistical run-time verification, it would be intriguing to extend the above property to *"Whether for the given confidence level α, M MonteCarlo trials and B Bootstrap trials, what is the probability of acceptance and rejection of the output voltage V_o for multiple trajectories $TRAC$ and varying input voltage V_T?"* Here, $TRAC$ is used to depict the band-gap reference circuit under different shot noise processes. For instance, if $TRAC = 100$, it represents "100" band-gap reference circuit models that have independent shot noise characteristics. For this case, the output voltage V_o should be bounded within $[V_o \geq 3.13mV]$ [52]. As a result, the null hypothesis H_0 and the alternative hypothesis H_1 of this property can be, respectively, expressed as

$$\begin{aligned} H_0 &: V_o \geq 3.13e{-}3; \\ H_1 &: V_o < 3.13e{-}3; \end{aligned} \tag{4.6}$$

Both the MonteCarlo and Bootstrap experiments were conducted for the confidence level $\delta = 0.95$ ($\alpha = 0.05$) for different tail tests, with shot noise only and with $TRAC = 200$ and with varying input voltage ($20mV \leq V_T \leq 29mV$). The results are summarized in Table 4.3.

Table 4.3: Statistical Runtime Verification Results for Band-Gap Reference Generator.

Additive Noise (TRAC = 200, M = MonteCarlo Trials, B = Bootstrap Trials, $\alpha = 0.05$)									
M =	**Tail**	**No Noise**				**With Shot Noise and V_T**			
		MonteCarlo		**BootStrap**		**MonteCarlo**		**BootStrap**	
B =	**Test**	**Accept**	**Reject**	**Accept**	**Reject**	**Accept**	**Reject**	**Accept**	**Reject**
1000	Lower	197	3	200	0	151	49	185	15
	Upper	193	7	200	0	159	41	187	13
	Two	191	9	200	0	147	53	176	24
10000	Lower	198	2	200	0	142	58	181	19
	Upper	199	1	200	0	151	49	182	18
	Two	193	7	200	0	152	48	178	22
50000	Lower	198	2	200	0	133	67	186	14
	Upper	198	2	200	0	127	83	190	10
	Two	197	3	200	0	121	89	179	21
Multiplicative Noise (TRAC = 200, M = MonteCarlo Trials, B = Bootstrap Trials)									
1000	Lower	200	0	200	0	181	19	199	1
	Upper	199	1	200	0	181	19	199	1
	Two	199	1	200	0	179	21	194	6
10000	Lower	199	1	200	0	188	12	199	1
	Upper	198	2	200	0	183	17	199	1
	Two	198	2	200	0	171	29	198	2
50000	Lower	198	2	200	0	193	7	197	3
	Upper	199	1	200	0	191	9	197	3
	Two	197	3	200	0	191	9	191	9

From Table 4.3, it is interesting to see that even in the absence of noise (column 3), irrespective of the tail test, the MonteCarlo technique produces some rejection as shown by the shaded region. This is because of the MonteCarlo theoretical assumption of normal distribution of the output voltage V_o has resulted in this false rejection. As the Bootstrap technique does not take into account any assumption on the output distribution, it has 100% acceptance of the output in the absence of noise (column 5). In cases where the shot noise and V_T variation are considered (columns 6-10), it can be seen that Bootstrap has more acceptance than MonteCarlo because of their resampling method. Also, the effect of additive shot noise is greater than that of multiplicative noise. This is because, the amplitude of shot noise is in the order of millivolts and when multiplied by the output voltage the effect is almost negligible.

The effect of shot noise and V_T on the statistical results can be pictured using shmoo plots as shown in Figure 4.6. For each V_T, the circuit is evaluated for different shot noise models by sweeping the amplitude appropriately. In the end, the total statistical result

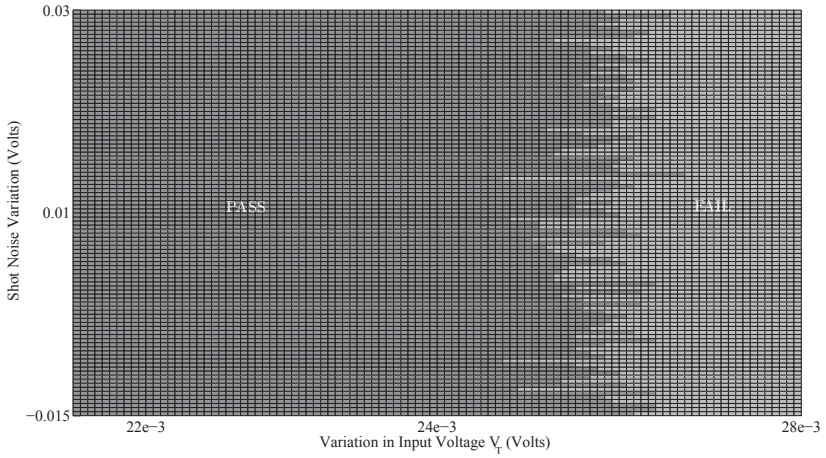


Figure 4.6: Shmoo Plotting of Band-Gap Reference Generator Results.

represents the number of passed/failed circuits that equal the total number of trajectories $TRAC$. The hypothesis testing result is analyzed by writing '1' for acceptance or '0' for rejection. The plot is shown for the MonteCarlo statistical results.

For this circuit, the worst case run-time for $M/B = 50000$ is around *3-4* hrs, which though is considerably less than the simulation done at the circuit level.

4.3.3 PLL Based Frequency Synthesizer

One of the major challenges for the verification of an AMS design, such as the PLL is evaluating the uncertainties due to short-term frequency perturbation known as the *jitter* [39]. Jitter, a time-domain measure, is an unwanted contraction or expansion in the output oscillating signal from its ideal position. Such instability can result in wrong synchronization of the AMS design and eventually lead to the loss of data.

The authors in [136] have made use of the jitter models from [39] and have combined hypothesis testing and MonteCarlo to provide a statistical estimate of the jitter property

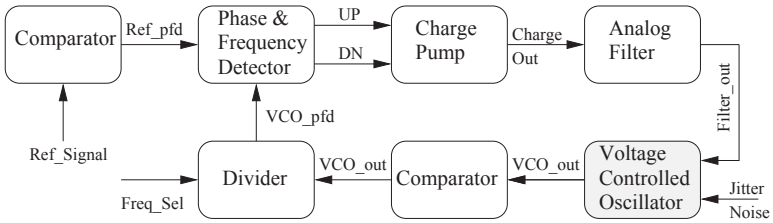


Figure 4.7: PLL Based Frequency Synthesizer

specification. Unfortunately, they failed to address the issue related to noise in the filter circuit and process variation associated with the circuit elements. Figure 4.7 shows a PLL based frequency synthesizer with jitter associated with the voltage controlled oscillator (VCO)[1].

The jitter model from [136] is used to address the issue of period jitter associated with the VCO. In general, it is modeled as a variation in the frequency of the VCO as shown in Figure 4.8.

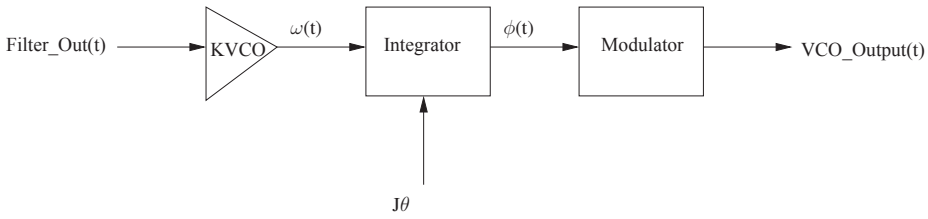


Figure 4.8: VCO Output with/without Noise

If T is the period of the ideal signal, then the frequency of the jittery signal can be defined as [76]

$$Jitter_{freq} = \frac{f}{1 + J\theta f} \tag{4.7}$$

where J is the jitter deviation, f is the input frequency and θ is the Gaussian random process. Using the VCO gain K_{vco} and the phase equation of the VCO from [76], the VCO output for a reference signal $Acos\omega t$ can be derived as

$$VCO\_output = Acos\left(\omega t + K_{vco}\int_0^t \frac{Filter_{out}(\tau)}{1 + \frac{J\theta \cdot K_{vco} \cdot Filter_{out}(\tau)}{2\pi}} d\tau + \phi_0\right) \tag{4.8}$$

[1]Please refer to Chapter 3 for a more detailed explanation.

With the jitter been defined, the next step is to represent the SDE behavior of the filter with additive and multiplicative thermal noise. This SDE representation can be described as

$$\dot{F_o} = \frac{1}{RC}(CP_o(t) - F_o(t)) + \sum_{k=1}^{2} \alpha\xi_k(t)$$
$$\dot{F_o} = \frac{1}{RC}(CP_o(t) - F_o(t)) + \alpha\xi_k(t)F_o(t) \tag{4.9}$$

where, F_o and CP_o represent the filter and charge-pump outputs, respectively, and R and C represents the resistor and capacitor components in the filter circuit, respectively. The next step is to apply the Euler/Milstein scheme described in Chapter 2 to generate the following numerical model:

$$F_{O_{n+1}} = F_{O_n} + \left(\frac{\Delta_n}{RC}\right)(CP_o(n) - F_o(n)) + \alpha\Delta W_{sn}$$
$$F_{O_{n+1}} = F_{O_n} + \left(\frac{\Delta_n}{RC}\right)(CP_o(n) - F_o(n)) + \alpha\Delta W_n + \tag{4.10}$$
$$\frac{1}{2}\left((\Delta W_n)^2 - \Delta_n\right)F_{O_n}$$

Statistical Property Observation

The lock-time is an isolated property for all PLL based frequency synthesizers, i.e., once the PLL gets locked, the VCO will start oscillating until there is a change to the *Freq_Sel* signal. The method of verifying the "lock time" property is to check if the output of the low-pass filter has reached a new DC value within the lock time as shown in Figure 4.9. In [135], the authors have verified the property without accounting for jitter in VCO and thermal noise in the filter.

For statistical run-time verification, the lock-time property is *"For the given confidence level α, M Monte Carlo trials, B Bootstrap trials, and multiple trajectory $TRAC$, what is the probability of acceptance and rejection that the PLL meet the lock-time of*

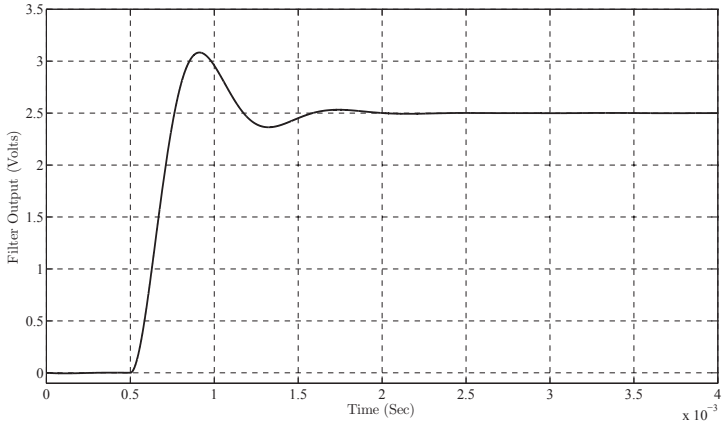


Figure 4.9: PLL Lock-Time Verification

0.001sec.?" Hence, the null hypothesis H_0 and the alternative hypothesis H_1 of this property can be, respectively, expressed as

$$\begin{aligned} H_0 &: lock\_time \leq 0.001; \\ H_1 &: lock\_time > 0.001; \end{aligned} \tag{4.11}$$

The simulation was carried out under the significance level $\alpha = 0.05$ and the jitter deviation as a normally distributed model as shown in Figure 4.10. The results for "200" trajectories are summarized in Table 4.4.

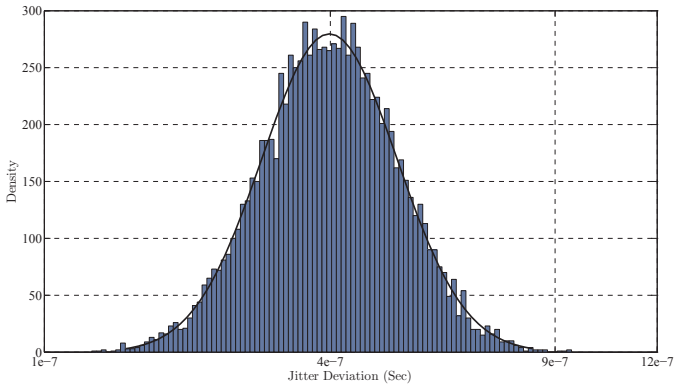


Figure 4.10: Jitter Deviation in VCO.

From Table 4.4, the combined jitter/thermal noise and process variation (columns 15-18) have substantially increased the PLL rejection, meaning PLL has failed to lock. The

Table 4.4: Statistical Runtime Verification Results for the PLL Lock-Time Property.

Additive Noise (TRAC = 200, M = MonteCarlo Trials, B = Bootstrap Trials, P.V = Process Variation, A = Accept, R = Reject, $\alpha = 0.05$)																	
M=	**Tail**	**No Noise & P.V**				**Noise Only**				**P.V Only**				**Noise & P.V Only**			
		MonteCarlo		**Bootstrap**		**MonteCarlo**		**Bootstrap**		**MonteCarlo**		**Bootstrap**		**MonteCarlo**		**Bootstrap**	
B=	**Test**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**	**A**	**R**
1000	Lower	152	48	200	0	129	71	181	19	180	20	187	13	137	63	174	26
	Upper	154	46	200	0	123	77	184	16	178	22	189	11	133	67	172	28
	Two	150	50	200	0	129	71	189	11	175	25	181	19	131	69	170	30
10000	Lower	168	32	200	0	124	76	173	27	172	28	185	15	121	79	169	31
	Upper	167	33	200	0	123	77	174	26	180	20	187	13	123	77	167	33
	Two	163	37	200	0	124	76	171	29	175	25	181	19	122	78	164	36
50000	Lower	148	52	200	0	119	81	177	23	177	23	182	18	112	88	164	36
	Upper	147	53	200	0	111	89	178	22	174	26	181	19	117	83	169	31
	Two	141	59	200	0	107	93	171	29	171	29	179	21	110	90	161	39
Multiplicative Noise (TRAC = 200, M = MonteCarlo Trials, B = Bootstrap Trials)																	
1000	Lower	154	46	200	0	128	72	189	11	180	20	187	13	139	61	177	26
	Upper	157	43	200	0	120	80	185	15	178	22	189	11	135	65	176	24
	Two	153	47	200	0	125	75	181	19	175	25	181	19	137	63	175	25
10000	Lower	159	41	200	0	121	79	175	25	172	28	185	15	130	70	174	26
	Upper	157	43	200	0	123	77	174	26	180	20	187	13	127	73	171	29
	Two	154	46	200	0	118	82	171	29	187	13	188	12	124	76	169	31
50000	Lower	147	53	200	0	107	93	171	29	175	25	181	19	111	89	168	32
	Upper	147	53	200	0	102	98	172	28	177	23	182	18	111	89	165	35
	Two	145	55	200	0	106	94	169	31	171	29	179	21	114	86	161	39

presence of jitter/thermal noise alone has shown higher rejection. This is obvious that the effect of thermal noise is reflected through the filter output, and at the VCO input, which again adds up the jitter noise. As the VCO is considered to be very sensitive, even a slight change to the input may cause substantial changes to its output. In some cases, the failure to lock does not mean that the VCO is not oscillating but, the oscillation is either "ugly" or delayed.

It is also obvious that the case of process variation only (columns 11-14) for additive/-multiplicative noise have resulted in the same number of rejection. This is because, both these cases have been simulated with the same process variation parameters. Also, both the additive/multiplicative noise have an equal influence on the overall rejections. This is because of the sensitive nature of the VCO, and even a millivolt drift in the input can cause substantial changes to the oscillation. A shmoo plot representing the pass/fail based on the lock-time is shown in Figure 4.11. For this circuit, the worst case run-time for

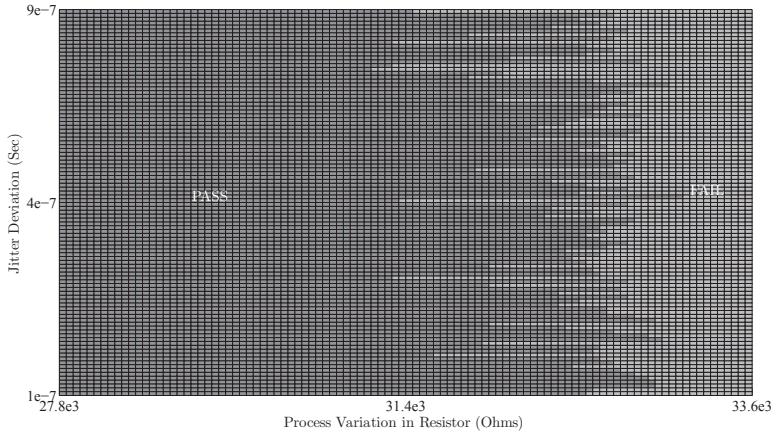


Figure 4.11: Shmoo Plotting of PLL Results.

$M/B = 50000$ is around *7-8* hrs, which is substantially high compared to previous circuits.

4.4 Summary

This chapter presented a methodology for the statistical verification of noise and process variation in an analog circuit. The approach is based on thermal and shot noise modeling in additive and multiplicative form using SDEs, and then integrating the device variation due to the *0.18*μm fabrication process in an SDE based simulation framework for verifying the statistical properties of the design. Our approach is illustrated on a Colpitts Oscillator, Band-Gap Reference generator and a PLL based frequency synthesizer circuit.

The statistical run-time verification method involves repeated simulation and can consume a lot of time and memory resources. The idea is to build a certain level of confidence in the circuit by analyzing the results from a large sample. The total number of samples depends on the values of M and B, and it is obvious that the higher those values are, the higher will be the confidence. As 100% confidence cannot be achieved using the run-time verification approach, it is necessary to complement them with other methods. Many enhancements can be made by combining run-time verification with formal methods to prove

properties of a given circuit.

So far in Chapters 3 and 4, we have established a quantitative based method for evaluating the analog circuit behavior with noise and process variation. By quantitative measurement, we mean that the circuit is evaluated either for "one" simulation trace (assertion) or multiple simulation traces (statistical). One of the key issues that remain unanswered is the quality of the simulation results. At transistor level simulation, the circuit quality is measured based on different criteria such as, SNR, NF and so on. But, if we look closely at those results, both SNR and NF represent a measurable quantity that may vary for different process conditions. However, the quality of a circuit can be determined by the quality of the simulation trace(s) it generates. From a verification perspective, such a metric remains to be a critical component, as it can determine if the simulation traces are "good", "bad" or "ugly". In the current state-of-the art, the quality of the simulation trace(s) are determined through visual inspection, which can be less productive and prone to human errors. Both, assertion/statistical based methods do not address this issue. Therefore, we propose in the next chapter a pattern matching based verification technique that could ensure not only the correctness of the analog design, but also the quality of its simulation trace.

Chapter 5

Qualitative Estimation Using Pattern Matching

From a functional verification perspective, the methods that have been discussed in Chapters 3 and 4 present a quantitative way to measure the circuit behavior. Unfortunately, assertion/statistical based verification approaches can sometimes exhibit violation that may not be associated with real design failures. For instance, if the output trace of a non-ideal circuit follow the trace of an ideal circuit for say 99.9% and violates for just 0.1% of the simulation time due to false spike in the simulator, then assertion/statistical methods will report a bug in the design. In such cases, the quantification methods fall short to enumerate the method of failure for the circuit behavior appropriately. To determine the quality of an analog circuit it is necessary to have methods that can estimate the overall quality of the circuit based on the simulation trace(s). This chapter relies on the combination of two pattern matching algorithms at a higher level of abstraction for the qualitative verification of an analog circuit influenced by random jitter conditions. The first algorithm, is the *longest closest subsequence* (LCSS), a variant of the longest common subsequence (LCS)

that is effective in estimating the percentage of matching between ideal and non-ideal circuit simulation traces influenced by noise and process variations. The second algorithm is a modification of *dynamic time warping* (DTW) that is used to handle instabilities in the simulation traces due to jitter conditions. The underlying idea of both these algorithm is to find the subsequence simulation trace between a set of analog signal traces and then use the combination of MonteCarlo and hypothesis testing to determine the probability of acceptance/rejection of those traces. The practical effectiveness of the proposed methodology is displayed using a Colpitts oscillator and a Phase Locked Loop (PLL) based frequency synthesizer circuit.

5.1 Introduction

Verification of analog designs is faced with immense challenges with the uncertainties due to short-term frequency perturbation known as the *jitter* [59] and the effect due to noise and process variations. The sources of jitter could be inherited from the circuit elements or from unwanted interaction between different analog/digital blocks. The amplitude associated with the jitter can be either bounded (*deterministic jitter*) or unbounded (*random jitter*) with respect to time [61]. The quantification of jitter relies on the kind of measurement used between the jittery and the ideal signal. Such enumeration is done with respect to the *phase*, *edge-to-edge* or the *period* [61] of jittery signal. The *phase* jitter is the edge timing difference between the ideal and the non-ideal signal. *Period* jitter is the difference between the phase jitter of the current cycle and that of the previous one. *Edge-to-edge*, also known as *cycle-to-cycle* jitter, is the two consecutive period deviation from the corresponding ideal signal period. All three jitter metrics are interrelated, meaning, the cycle-to-cycle jitter is considered to be the first difference function of period jitter and the second difference function of the phase jitter [87].

The jitter model, the verification environment and the class of AMS circuit are key components for evaluating the jitter. Due to its random nature, the first step in jitter measurement is to characterize its behavior as a probabilistic function and the most prominent approach is the use of a Gaussian distributed model [87]. This chapter addresses the above issues, by ensuring the correctness of analog circuits in the presence of noise (*jitter, thermal, and shot*) and process variations using the concept of pattern matching.

The pattern matching techniques are commonly applied to the characterization and validation of high-speed analog and digital circuits. Quite often they are associated with the study of crosstalk, coupling, delays in the data transmission lines [60] during the post-layout and board-level signal integrity (SI) analysis. CAD tools for SI analysis (e.g., [65]) provide a unique waveform comparison capability that can ensure a reliable high-speed data transmission, achieved through interconnect characterization and lab measurements [18]. In the current state-of-the-art, SI analysis can be performed only on the circuit-level simulation traces and board-level design waveforms. In general, any SPICE based simulator can generate SI analysis models for an analog circuit which then could be ported to any standard CAD tool to determine the quality of the simulation traces through waveform comparison. Such a specific trace comparison method can assist the designers to examine the design failures for validity.

To resolve the issue of false violation, the approach based on quantitative methods has to be complemented with a more meaningful analysis of the circuit simulation trace. In the current design/verification flow, the missing qualitative assessment of an AMS design at a higher level of abstraction can be achieved by extending the pattern matching concepts developed in SI analysis to the functional verification. As depicted in Figure 5.1, the verification based on pattern matching will also help to address the question of *"How to decide on the acceptance/rejection of a circuit simulated with N different process conditions and*

by N different designers?"

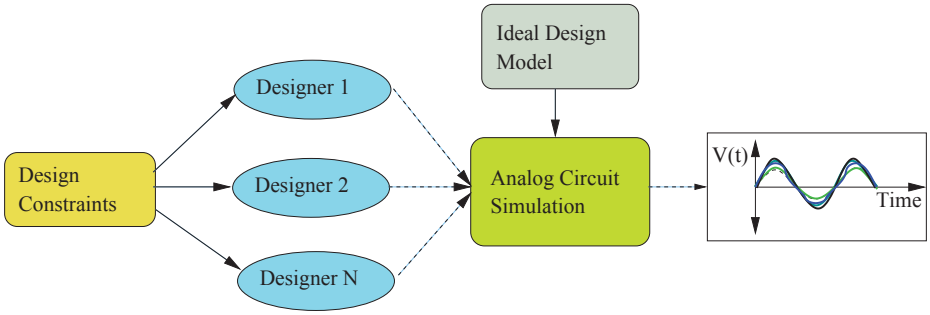


Figure 5.1: Analog Verification.

As most of the SI based waveform comparison algorithms are propriety to the tool developers, the challenge is to develop an algorithm that is tailored towards the AMS verification. This chapter takes a look at two popular pattern matching algorithms that are based on dynamic programming [36]. The underlying idea of these algorithms is to find the subsequence simulation trace between a set of analog signal traces and combine hypothesis testing to determine the probability of failures. Hypothesis testing [96] is the use of statistics to determine the probability that a given hypothesis is true. The statistical property, is expressed as a null hypothesis and in the end, a circuit is *accepted/rejected* with a certain confidence level and error margin.

The Longest Common Subsequence (LCS) is a pattern matching algorithm that finds its applications in computational biology, chip layout design, and so on [129]. In DNA matching, the idea is to find the longest subsequence common to all sequences in a set of sequences [36]. As opposed to the traditional approach of comparing the output of the design to its specification value, we can extend the LCS algorithm to estimate in terms of percentage the exact (100%) or *"closely"* matched simulated output relative to the ideal circuit output. By doing so, instead of blindly rejecting the circuit that violates the specification, designers will have more information during the evaluation and hence can make viable decisions.

The LCSS algorithm is efficient in finding the percentage of the closest match between a set of simulation trace that operates with the same time-period (frequency). Any variation to the operating frequency between the ideal and the non-ideal signal (e.g., jitter) will produce erroneous matching results. Alternatively, Dynamic Time Warping (DTW) [115] is a pattern matching algorithm that finds its applications in audio, video, and graphics designs to cope for different signal speeds. DTW is a method that finds an optimal path between two given time series sequences. Similar to LCSS, the DTW algorithm can be extended to measure the best possible closest match between the jittery output signal and the specification. Therefore, it facilitates a unified approach for verifying jitter in an AMS design. The main advantage of the pattern matching based approach is that the whole verification process is independent of the circuit models and can be applied to perform a qualitative assessment of any black-box design.

In analog designs, a *"closely"* matched relation can be defined in many different ways. Let us consider V_1 and V_2 as the output sequences of an ideal and a non-ideal circuit, respectively. First, we say that two output sequences of values V_1 and V_2 are similar if one is a subsequence of the other [36]. Alternatively, another way to measure the similarity between V_1 and V_2 is by finding a third longest sequence of values V_3 that appear in each of the sequences V_1 and V_2 [36]. In reality, it is difficult to find a one-to-one mapping between V_1 and V_2 and hence, designers have to define an acceptable tolerance range for the output as a part of the specification. Thereafter, the LCSS and DTW algorithms are defined to quantify the simulated output relative to a specification template. In the next section of this chapter we discuss details of the proposed methodology and its application to analog circuits. For noise, the idea is to apply SDE to model design and integrate device variation in a MATLAB simulation environment. The efficiency of LCSS algorithm is illustrated on a Colpitts oscillator circuit to study the effect of noise and process conditions. The influence

of jitter noise on the "lock-time" property of a phase locked loop (PLL) based frequency synthesizer is analyzed through DTW algorithm.

5.2 Proposed Methodology

Figure 5.2 shows the overall verification methodology based on pattern matching algorithm. Given an analog design described as a system of *ODEs*, the idea is to generate SDEs that describe the noise behavior. For the case of circuits that do not have a closed form solution, the approach is to numerically approximate the SDE's based on Euler-Maruyama technique as described in the Chapter 2. For process variation, the different circuit parameters are derived using Gaussian distribution as described in Algorithm 3.3. The SDE model, process variation, and the environment constraints are evaluated using MonteCarlo simulation in a MATLAB environment. We then generate a set of sequences, one considered to be the sequence of an ideal circuit (without noise and process variation) and the rest to be of a non-ideal circuit (with noise and process variation). These sequences along with the given tolerance level (p) are evaluated using the LCSS and DTW algorithms in a MonteCarlo simulation environment to decide on the probability of accepting or rejecting of the circuit as shown in Figure 5.2. The decision to use LCSS or DTW is determined by evaluating the frequency match between the ideal and non-ideal signals. If the frequency matches then the LCSS algorithm is used, else the DTW algorithm as depicted in Figure 5.2.

In general, the relative error of MonteCarlo method can be monitored by the figure of merit [25]. For example, 90% accuracy and 90% confidence level can be achieved when figure of merit equals to 0.1. However, a trade-off exists between the number of trials and the simulation run times [96].

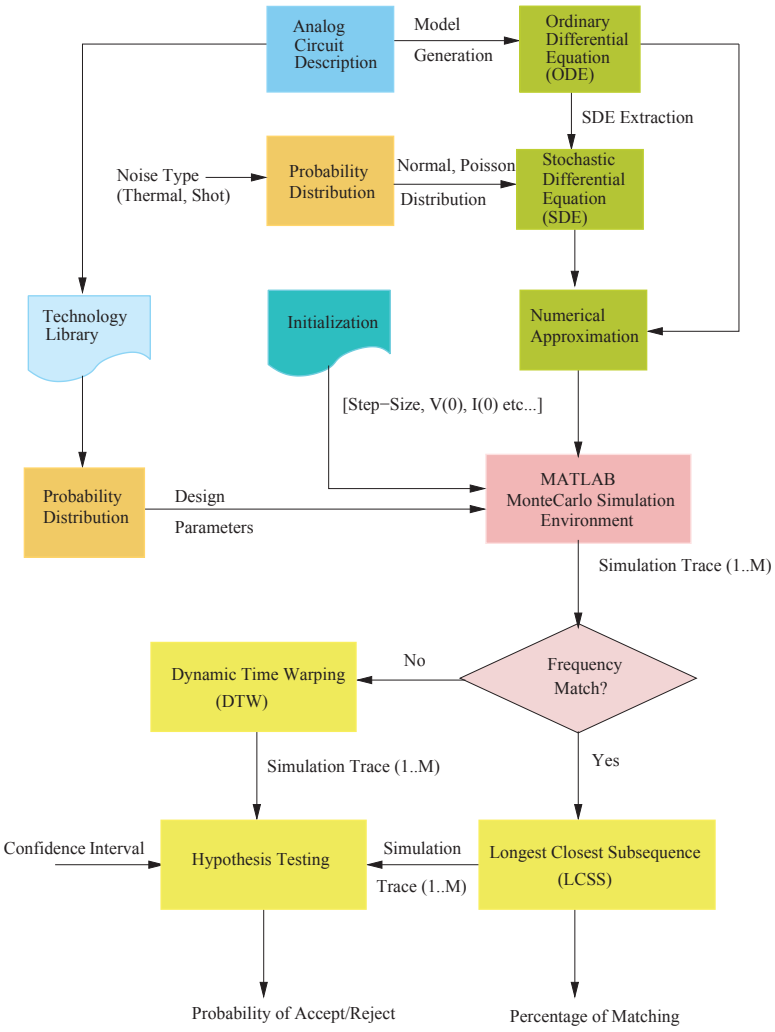


Figure 5.2: Overview of Pattern Matching Verification Methodology

5.2.1 Longest Closest Subsequence (LCSS)

Given two sequences of analog circuit output values $X = \{X_1, X_2, \ldots X_m\}$ and $Y = \{Y_1, Y_2, \ldots Y_n\}$, then $\exists Z = \{Z_1, Z_2, \ldots Z_k\}$, an increasing maximum-length common subsequence, if Z is a subsequence of both X and Y. Here, we choose X to be the output sequence of an ideal circuit and Y is the output of the non-ideal circuit. It can also be noted that the length of $k \leq$ length of m/n. To extend the LCS theorem [36] for analog circuits, we need to define a tolerance parameter *p* that describes the allowable boundary conditions for the sequence Y with respect to X. The recursive solution will detect only the values

that are not in the region as defined by $X - p$ and $X + p$. A large value of p means less accuracy and vice-versa.

Property 1 *Let* $X = \{X_1, X_2, \ldots X_n\}$ *and* $Y = \{Y_1, Y_2, \ldots Y_m\}$ *be sequences, and let* $Z = \{Z_1, Z_2, \ldots Z_k\}$ *be any LCSS of* X *and* Y*. Then, for the given* p*,*

(1) If $Y_n \leq (X_m$+$p)$ *and* $Y_n \geq (X_m$-$p)$*, then* Z_k *is an LCSS of* X_m *and* Y_n*.*

(2) If $X_m \neq Y_n$ *and* $Z_k \neq X_m$*, then* Z *is an LCSS of* X_{m-1} *and* Y*.*

(3) If $X_m \neq Y_n$ *and* $Z_k \neq Y_n$*, then* Z *is an LCSS of* X *and* Y_{n-1}*.*

A brute-force method to compute LCSS(X, Y) involves computing all subsequences of X, checking if they are subsequences of Y and be the longest. For a given sequence (X and Y) of length m and n, respectively, we have 2^m subsequences of X and it takes $\mathrm{O}\left(n \cdot 2^m\right)$ to compute the LCSS. As we can see from Property 1, there are overlapping subproblems in finding the LCSS of X and Y_{n-1} and X_{m-1} and Y, hence, it is efficient to implement the above property recursively with the computation time of $\mathrm{O}\left(m \cdot n\right)$. The recursive solution to the LCSS problem is based on finding an intermediate length $C[i, j]$ of LCSS(X,Y), where i and j represent the i^{th} and j^{th} position of X and Y, respectively. If either $i = 0$ or $j = 0$, one of the sequences has length zero, so the LCSS has length zero as defined in Property 2.

Property 2 *Require:* $i,j \neq 0$

$$C[i,j] = \begin{cases} C[i-1, j-1] + 1; & \textit{if } Y_j \in [X_i - p,\ X_i + p] \\ max\{C[i, j-1], C[i-1, j]\}; & \textit{otherwise} \end{cases}$$

To better understand the Properties 1 and 2, let us apply them to the Chua circuit [32] shown in Figure 5.3. The behavior of chaos is caused by the non-linear resistance *RL*. If the

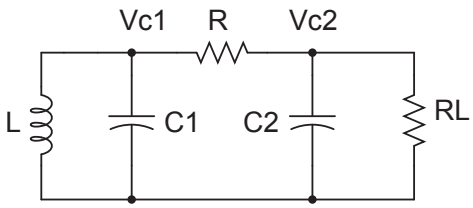


Figure 5.3: Chua Circuit.

value of the non-active circuit components are chosen properly, instead of chaos, the circuit will demonstrate a stable oscillation as shown in Figure 5.4 (a).

The dotted waveform (X) represents the simulation result of the circuit in the absence of noise, and the bold line represents the result in the presence of noise (Y) and process variation for *R, L*, C_1 and C_2 with a distribution shown in Figures 5.4 (b) to (d). For illustration purposes, we have taken four sample points and a tolerance level of p = 4% from the simulation results (Figure 5.4 (a)) to find the LCSS. The first step is to apply the recursive solution to create a matrix table for the two sequences X and Y as shown in Figure 5.5.

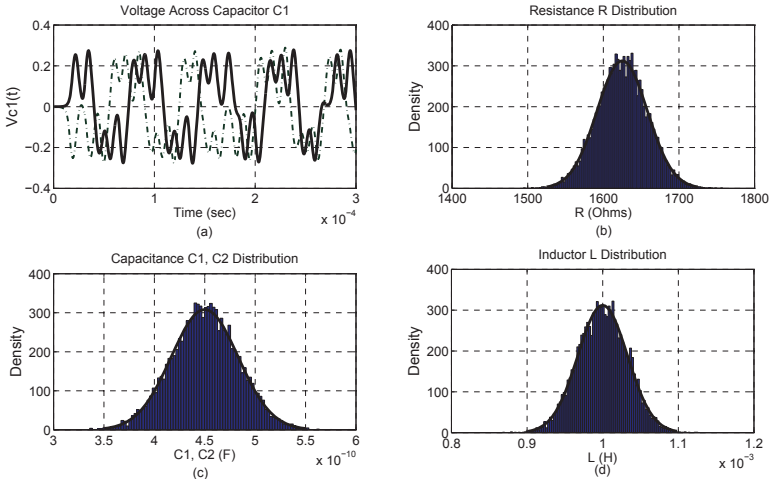


Figure 5.4: Chua Simulation Result with Process Variation.

The implementation of the LCSS matrix table in Figure 5.5 is described in Algorithm 5.1. First, we define an intermediate LCSS $C[i, j]$ and initialize it to zeros (lines

3-8). If sequence Y_j does match with X_i, then $C[i,j]$ is computed by adding "1" to the value in the position $C[i-1,j-1]$ (lines 11-12) as shown by the entries "B", "D" and all the dotted arrows in Figure 5.5. If the sequence Y_j does not match with X_i, then based on the comparison (bold arrows in Figure 5.5) between $C[i-1,j]$ and $C[i,j-1]$ (line 14) we determine the corresponding value in table. For instance, if the computation of $maxC[i-1,j], C[i-1,j]$ (line 14) resulted in $C[i-1,j] > C[i-1,j]$, then we make $C[i,j] = C[i-1,j]$, if not $C[i,j-1]$ is marked by the entry "A" or "C" in Figure 5.5. The algorithm continues until we reach the end of the sequence, and it returns the matrix table $C[i,j]$ (line 18).

Algorithm 5.1 : LCSS algorithm

Require: $X,\ Y,\ p$
1: $m \leftarrow Length[X]$
2: $n \leftarrow Length[Y]$
3: **for** $i \leftarrow 0\ to\ m$ **do**
4: $C[i,0] \leftarrow 0$
5: **end for**
6: **for** $j \leftarrow 0\ to\ n$ **do**
7: $C[0,j] \leftarrow 0$
8: **end for**
9: **for** $i \leftarrow 1\ to\ m$ **do**
10: **for** $j \leftarrow 1\ to\ n$ **do**
11: **if** ($Y_j \leq (X_i$ + p)) &&($Y_j \geq (X_i$ -p)) **then**
12: $C[i,\ j] \leftarrow C[i-1,\ j-1]+1$
13: **else**
14: $C[i,\ j] \leftarrow max\{C[i-1,\ j],\ C[i,\ j-1]\}$
15: **end if**
16: **end for**
17: **end for**
18: **return** C

Now, using the matrix table $C[i,j]$ we can trace back the path to determine the longest closest sequence between the two analog output sequences. The elements of the LCSS are encountered in reverse order by this method as described in Algorithm 5.2, which performs an inverse operation of Algorithm 5.1 to determine the values associated with the index.

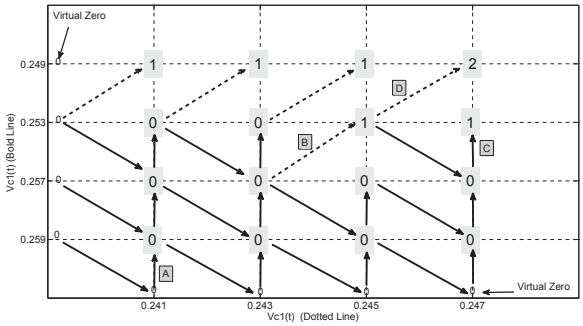


Figure 5.5: LCSS Table of Computation.

The starting point is $C[m, n]$ and each row and column are parsed until $C[i, j]$ changes to a new value (lines 7-12). By doing so, we are in a position to detect the index of matched values (or its deleted ones) (line 13). In the case of the Chua circuit for the given tolerance level, we found only two closely matching values of Y with X for the chosen four sequence points as shown by the circled points in Figure 5.6.

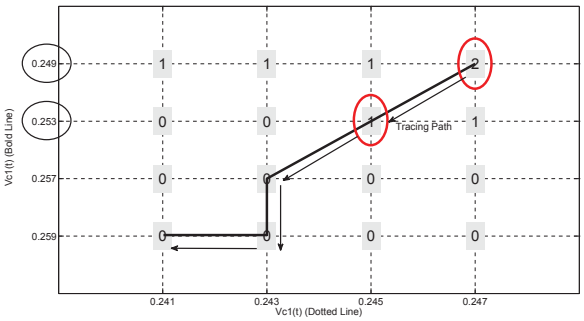


Figure 5.6: Tracing LCSS for the Chua Circuit

5.2.2 Dynamic Time Warping (DTW)

Dynamic time warping (DTW) is an algorithm developed by the speech recognition community to handle the matching of non-linearly expanded or contracted signals [118]. The algorithm finds the optimal path through a matrix of points representing possible time alignments between the signals. The optimal alignment can be efficiently calculated via dynamic programming.

Algorithm 5.2 Traceback Algorithm

Require: C

1: $lig \leftarrow n + 1$
2: $col \leftarrow m + 1$
3: $index \leftarrow [\,]$
4: $num \leftarrow C[lig,\ col]$
5: **if** $num \neq 0$ **then**
6: **while** $(lig\ >=\ 2)$ && $(col\ >=\ 2)$ && $num > 0$ **do**
7: **while** $(C[lig, col-1] >= num)$ && $C[lig, col-1] >= C[lig-1, col]$ **do**
8: $col \leftarrow col - 1$
9: **end while**
10: **while** $(C[lig-1, col] >= num)$ && $C[lig, col-1] <= C[lig-1, col]$ **do**
11: $lig \leftarrow lig - 1$
12: **end while**
13: $index \leftarrow [col - 1,\ index]$
14: $num \leftarrow num - 1$
15: $lig \leftarrow lig - 1$
16: $col \leftarrow col - 1$
17: **end while**
18: **else**
19: $print$ " $There\ is\ no\ matching$ "
20: **end if**
21: **return** $index$

Property 3 *Given any two sequences* $x = \{x_1, x_2, \ldots x_m\}$ *and* $y = \{y_1, y_2, \ldots y_n\}$*, then the distance of the best possible partial path is defined as,*

$$D(i,j) = d(x_i, y_j) + min \begin{cases} D(i, j-1) \\ D(i-1, j) \\ D(i-1, j-1) \end{cases} \tag{5.1}$$

where, $1 \leq i \leq m$; $1 \leq j \leq n$. $d(x, y)$ is a distance between the signals.

To better understand the algorithm, let us apply it to the following traces shown in the Figure 5.7.

The dotted waveform represents the jitter signal (Y) and the bold trace is the ideal signal (X). We have the following values for X and Y,

$X\ =\ [\ 00000111110000011111\ ]$

$Y\ =\ [\ 00000111100000111111\ ]$

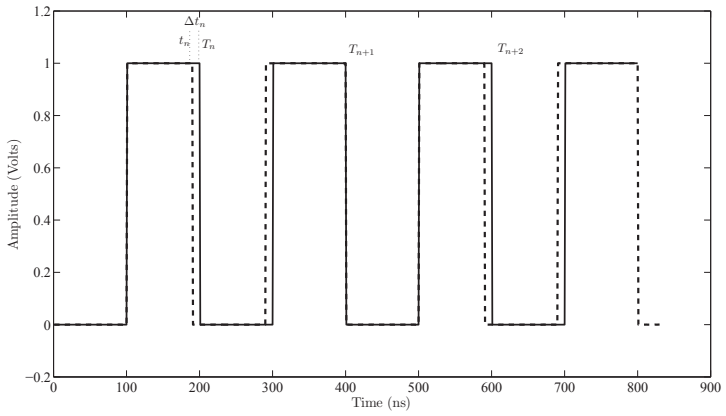


Figure 5.7: Dynamic Time Warping Example

The first step is to determine a *"m by n"* matrix table that represents the best possible distance between X and Y as shown in Table 5.1.

Table 5.1: Dynamic Time Warping Matrix

D[i, j]	1	2	3	4	5	6	7	8	· · ··	21
1	0	∞	∞	∞	∞	∞	∞	∞	· · ··	∞
2	∞	0	0	0	0	0	1	2	· · ··	10
3	∞	0	0	0	0	0	0	1	· · ··	10
4	∞	0	0	0	0	0	0	0	· · ··	10
5	∞	0	0	0	0	0	0	0	· · ··	10
6	∞	0	0	0	0	0	0	0	· · ··	10
7	∞	1	1	1	1	1	0	0	· · ··	4
· · ··	· · ··	· · ··	· · ··					· · ··	· · ··	6
· · ··	· · ··	· · ··	· · ··					· · ··	· · ··	6
16	· · ··	· · ··	· · ··				0	1	· · ··	6
· · ··	· · ··	· · ··	· · ··					· · ··	· · ··	6
21	∞	10	10	10	10	10	10	5	· · ··	0

The importance of this table is to understand the contraction and expansion of the two simulation traces. By contraction, we mean the jitter signal period is shifted to the left of the ideal signal and expansion represents the right shift of the jittery signal with respect to the ideal signal.

The implementation of the matrix table is described in Algorithm 5.3. The algorithm starts by reading the two sequences X and Y. The entries in the matrix table are generated as

follows: First, we define a matrix *D* that fills the first row and column with ∞ and $D(0,0)$ with 0 (lines 3-9). Then, it takes the minimum between $D(i-1,j-1), D(i,j-1), D(i-1,j)$, and adds to it the cost which is the distance between x_i and y_j (lines 10-15). Here *i* and *j* represent the indexes of the sequences X and Y, respectively. The algorithm continues until we reach the end of the sequence, and it returns the matrix table *D* for determining the optimal path and allows us to have the minimum cost alignment *D(n,m)*.

Algorithm 5.3 : DTW Algorithm

Require: x, y
1: $n \leftarrow length(x)$
2: $m \leftarrow length(y)$
3: **for** $i \leftarrow 1$ *to* m **do**
4: $D(0,i) \leftarrow inf$
5: **end for**
6: **for** $j \leftarrow 1$ *to* n **do**
7: $D(j,0) \leftarrow inf$
8: **end for**
9: $D(0,0) \leftarrow 0$
10: **for** $i \leftarrow 1$ *to* m **do**
11: **for** $j \leftarrow 1$ *to* n **do**
12: $cost \leftarrow d(x(i) - y(j))$
13: $D(i,j) \leftarrow cost + min(D(i-1,j-1), D(i,j-1), D(i-1,j))$
14: **end for**
15: **end for**
16: **return** D

Once, the matrix table is generated it is necessary to trace back the path to detect those values that represent the contraction and expansion. Algorithm 5.4 describes the way the trace-back is done. A visual representation of the DTW results with contraction and expansion is shown in Figure 5.8, with the jittery/ideal signal in X and Y-axis, respectively.

The algorithm detects the contraction (vertical line) or expansion (horizontal line) of the output signal. The result is skew from the diagonal which is the ideal output without any jitter noise. A good path is unlikely to wander very far from the diagonal. When there is no

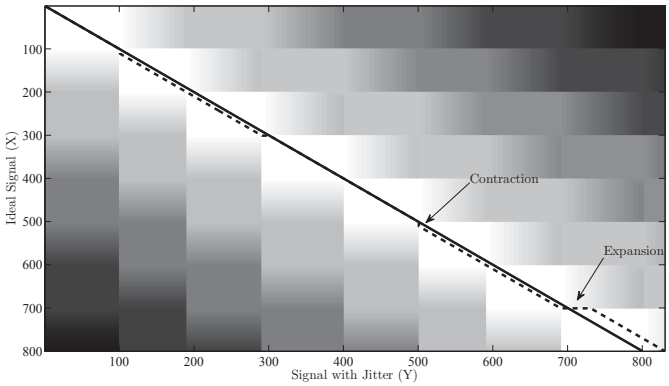


Figure 5.8: Dynamic Time Warping Results

timing difference between these patterns, the warping function coincides with the diagonal line $j = i$. It deviates further from the diagonal line as the timing difference grows. The closer the two lines are, the better the cost function. From a verification point of view, this means that the jitter does not have any effect on the simulation trace.

Algorithm 5.4 takes the matrix D as an input. It starts from the end *D(n,m)* (line 2-3) and calculates the points that contributed for the minimum cost function. For instance, if the minimum is at *D(i-1,j-1)* (line 7), then the algorithm will subtract diagonally -1 from both *i* and *j* (line 8-9), else, it will just subtract -1 from *i* or from *j* (line 10-13). The values associated in the i^{th} and j^{th} positions are then regrouped (stored) in *p* and *q* vectors, respectively (line 17-18). The algorithm returns *p*, *q* values that constitute the minimum path.

5.2.3 Hypothesis Testing

For a given analog circuit, every output simulation trace is considered to be a random variable *X*. As defined in Chapter 2, for a specified confidence level, a *two-tailed* test can be applied to decide on the acceptance/rejection of the circuit. The detailed procedure for bounded hypothesis testing is illustrated in Algorithm 5.5.

Algorithm 5.4 : Trace-Back Algorithm

Require: D

1: $[n, m] \leftarrow size(D)$
2: $i \leftarrow n$
3: $j \leftarrow m$
4: $p \leftarrow i$
5: $q \leftarrow j$
6: **while** $i > 1$ && $j > 1$ **do**
7: **if** $min(D(i-1, j-1), D(i, j-1), D(i-1, j)) = D(i-1, j-1)$ **then**
8: $i \leftarrow i - 1$
9: $j \leftarrow j - 1$
10: **else if** $min(D(i-1, j-1), D(i, j-1), D(i-1, j)) = D(i-1, j)$ **then**
11: $i \leftarrow i - 1$
12: **else if** $min(D(i-1, j-1), D(i, j-1), D(i-1, j)) = D(i, j-1)$ **then**
13: $j \leftarrow j - 1$
14: **else**
15: $"Exit"$
16: **end if**
17: $p = [i, p]$
18: $q = [j, q]$
19: **end while**
20: **return** p, q

The first step is to determine the kind of distribution associated with the output simulation trace. It is quite natural to assume a normal distribution for the outputs, however, the variation due to technology and mismatch may sometimes lead to other distributions. Lines (1-19) take into account different distributions and in turn deduce the cumulative distribution function ($CDF(x)$). This is followed by the search for "lower" and "upper" bounds of the critical value that satisfy Equation (2.45) (lines 20-27). Both the "lower" and "upper" bounds define the acceptance region (where H_0 is accepted) for every random variable X.

Algorithm 5.5 : Hypothesis Testing (Two-Tailed Test)

Require: $Distribution, Parameters$

if $(Distribution = LogNormal)$ **then**
  $\sigma \leftarrow Parameters(1)$
  $\mu \leftarrow Parameters(2)$
  $\gamma \leftarrow Parameters(3)$
  $CDF(x) \leftarrow \Phi\left(\frac{\ln(x-\gamma)-\mu}{\sigma}\right)$
else
  if $(Distribution = Normal)$ **then**
    $\sigma \leftarrow Parameters(1)$
    $\mu \leftarrow Parameters(2)$
    $CDF(x) \leftarrow \Phi\left(\frac{x-\mu}{\sigma}\right)$
  else
    if $(Distribution = Weibull)$ **then**
      $\alpha \leftarrow Parameters(1)$
      $\beta \leftarrow Parameters(2)$
      $\gamma \leftarrow Parameters(3)$
      $CDF(x) \leftarrow 1 - \exp\left(-\left(\frac{x-\gamma}{\beta}\right)^{\alpha}\right)$
    end if
  end if
end if
$lower \leftarrow Initial\ Value\ Low$
while $CDF(Lower) \leq 0.05$ **do**
  $lower \leftarrow lower\ +\ Step$
end while
$upper \leftarrow Initial\ Value\ Up$
while $CDF(Upper) \leq 0.95$ **do**
  $upper \leftarrow upper\ +\ Step$
end while
return $lower, upper$

5.3 Applications

The efficiency of the proposed methodology is illustrated on a Colpitts and a PLL based frequency synthesizer circuits. LCSS is used to evaluate the Colpitts oscillator circuit influenced by noise and process variation. The effect of jitter is analyzed through the DTW algorithm. We have used the MATLAB simulation environment (Windows Vista, AMD Dual-Core, 4GB RAM) for a *0.18μm* process [1] conditions. We have also applied the proposed methodology to a transistor level Rambus [67] Ring Oscillator circuit for the *90nm* technology. The results can be found in [104].

5.3.1 Colpitts Oscillator Circuit

The circuit diagram for a MOS transistor based Colpitts oscillator is shown in Figure 3.9. For the correct choice of component values the circuit will oscillate due to the bias current and negative resistance of the passive tank. The frequency of oscillation is determined by L, C_1 and C_2.

Finding the Longest Closest Subsequence

The first step in finding the LCSS, is to generate two sets of sequences (ideal and non-ideal) for different cases as shown in Figure 5.9. For such simulation results, the property of interest is: *"Whether or not for the given set of parameters, the inductor current is within a certain bound?"*

Figures 5.9 (a) to (d) show the simulation results for the Colpitt's oscillator under both ideal and non-ideal conditions. Figure 5.9 (a) shows the results in the absence of noise and process variation, which will be considered as an ideal output. From the Figures 5.9 (b) to (d), we note that, for the total simulation time of 1.0×10^{-6}, we come across some increase in amplitude for the circuit influenced by noise and process variation at certain

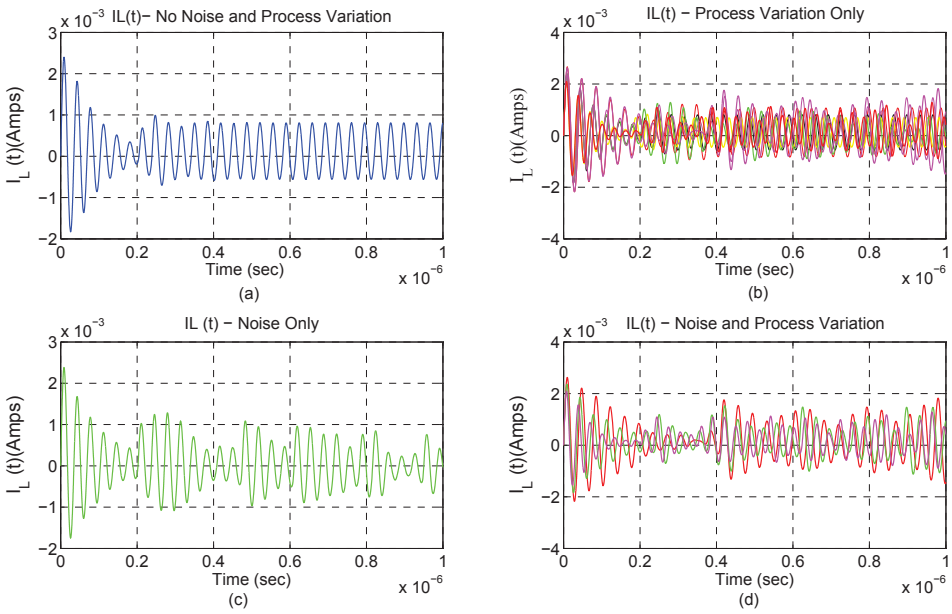


Figure 5.9: Colpitt's Simulation Results

simulation points. This is because, the variation in device parameter and additive noise in the current equation has caused an increase in the inductive current thereby creating a non-uniform output. The question now is: *"Do we have to reject those circuits entirely?"* To assist the designer in making a better decision, we can determine for different tolerance levels, the LCSS for different conditions using the technique described in Section 5.2.1. All together, we have one ideal and seven different Colpitts oscillator circuit implementations. For each of the seven different circuits, we compare the sequence with the ideal sequence in order to generate the LCSS as summarized in the top half of Table 5.2.

The experiments were conducted for different tolerance levels and for a sequence length of 1000 between the two outputs and for three different process conditions ($slow, nominal$ and $fast$). From Table 5.2, we see that when we consider the effect of noise only (column 2), based on the tolerance level, we find a smaller number of closely matched sequences. This is because, the additive Wiener process in the SDE model makes the inductor current to deviate from its specified value, thereby creating discrepancies between the ideal circuit

Table 5.2: Longest Closest Subsequence Computation Results.

MonteCarlo Trials M= 1000, P.V = Process Variation							
Tolerance	**With Noise Only**	**With P.V Only**			**With Noise & P.V**		
(%)	**No. of LCSS**	**No. of Slow LCSS**	**No. of Nominal LCSS**	**No. of Fast LCSS**	**No. of Slow LCSS**	**No. of Nominal LCSS**	**No. of Fast LCSS**
0.1	472	237	1000	2	235	440	2
0.5	585	341	1000	6	334	559	8
1.0	603	345	1000	12	338	573	16
2.0	640	349	1000	22	342	603	31
5.0	801	366	1000	53	358	705	71
8.0	948	373	1000	82	373	866	107
10.0	971	374	1000	100	385	976	129
12.0	973	375	1000	118	396	978	149
15.0	976	377	1000	144	397	980	182
No. of LCSS Computation for Different Monte Carlo Trials for Tolerance = 10%.							
Monte Carlo	**With Noise Only**	**With P.V Only**			**With Noise & P.V**		
Trials	**No. of LCSS**	**No. of Slow LCSS**	**No. of Nominal LCSS**	**No. of Fast LCSS**	**No. of Slow LCSS**	**No. of Nominal LCSS**	**No. of Fast LCSS**
10000	901	393	916	141	394	1000	121
25000	1000	411	845	147	399	979	139
50000	875	471	831	133	443	991	153
100000	893	479	971	119	471	963	155

and the noisy circuit. A tolerance level of 0.1% means that the output sequence of the non-ideal circuit is within $\pm 0.1\%$ range of the ideal circuit output. It is also evident from columns 3-5, that the analysis with parameter variation due to $0.18\mu m$ shows little effect for the nominal process and adverse effect for the $fast$ process corner. This is because $\pm 3\sigma$ parameter variation is large enough to create discrepancy on the inductor current. In contrast, in columns 6-8 of Table 5.2, it is evident that the effect of noise and process variation have led to minimum number of matches between the two sequences.

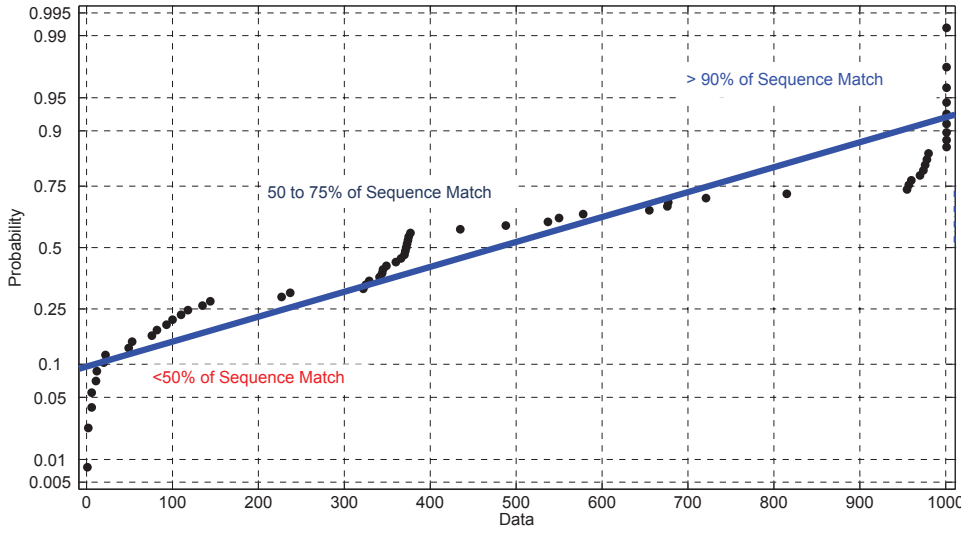


Figure 5.10: Cumulative Distributive Function for Table 5.2.

The best way to describe the results shown in Table 5.2 is through probability plot as shown in Figure 5.10. Depending upon the type of simulation results, a lower probability of sequence match means lower tolerance level and vice-versa. For instance, in the noise only case (column 2), we can achieve a high level of sequence matching for a high tolerance level. However, for process variation cases, if we consider an acceptable tolerance level (say 10%), then we move into higher probability range for the nominal corners, but will have minimum matching depending on the process corners.

We carried out the analysis for different MonteCarlo trials for a tolerance level of 10% and the results are summarized in the bottom half of Table 5.2. As seen, it is apparent that with a large number of MonteCarlo trials, we have the leverage to work on a larger group of samples and the variation on the length of the sequences tends to change considerably when compared to the previous results.

The results for the MonteCarlo trials are plotted as normal distribution curve as shown in Figure 5.11. The combined MonteCarlo and LCSS analysis can be different for tolerance confidence levels and the accuracy would be compromised if the tolerance level is too high or the number of trials being too low. Higher tolerance levels would increase the error margin and degrade the reliability.

5.3.2 PLL based Frequency Synthesizer

Lots of progress has been made in estimating the effect of jitter for an analog design in phase/voltage domains. For instance, for a PLL design, researchers achieve a good fit between the measured and the extracted values by advocating the use of gaussian distributed jitter models that are simulated at higher level of abstraction using the phase domain method [39]. Taking a step further, the author in [76] used the ideas of [39] to develop a methodology for commercial purposes by integrating Verilog-A based jitter model with

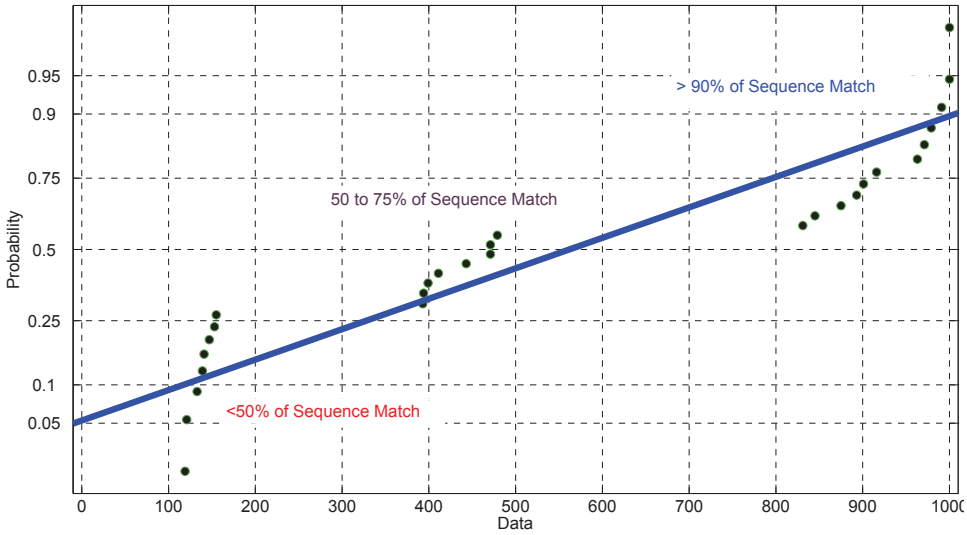


Figure 5.11: Probability Plot for Table 5.2.

Spectre [77]. As this involves behavioral simulation, it lays a strong foundation for a system level verification of analog/mixed-signal designs.

A similar phase-domain approach reported in [111] calculates the overall jitter noise power of a $\Delta\Sigma$ modulator based frequency synthesizer as a function of the bandwidth. However, with the need for accurate sampling associated with the modulator, the overall system dynamic response appears to be very slow. A different phase domain technique proposed in [99] makes use of non-linear equations for the phase error that are solved to detect random unsteadiness that characterize the timing jitter. Another methodology based on voltage domain models reported in [76] allows the designer to formulate the jitter noise in terms of voltages that are then added to the circuit. The use of a voltage-domain method in a Verilog-A environment has been campaigned by the authors in [94], wherein, the jitter properties of the synthesizer are extracted from transistor level through simulation. Unfortunately, simulation based verification approaches remain rigid to that particular analog design and taking an unified approach require colossal changes to the methodology and hence, is impractically expensive.

Figure 3.12 shows a PLL based frequency synthesizer that is commonly used in communication systems for clock generation and recovery. It is composed of two comparators, a phase/frequency detector, a charge pump, an analog filter, a voltage controlled oscillator (VCO) and a divider.

We incorporate in a MATLAB simulation environment the SRE based models [10] for the VCO with jitter and for the other blocks of the PLL design. We have applied the DTW algorithm in two ways: First, to study the effect of jitter on the "lock-time" property, and second, to estimate the optimal cost alignment by combining MonteCarlo simulation for "1000" trials with the bounded hypothesis testing.

Lock-Time Property Observation:

The critical property of a frequency synthesizer is the "lock-time", meaning, if the *Freq_Sel* is activated, the PLL will lock at the desired frequency within a certain time as identified in the specification. However, the jitter in the VCO circuit may cause a drift in its output that may lead to changes to the lock-time. The lock-time is an isolated property for all PLL based frequency synthesizers, i.e., once the PLL gets locked, the VCO will start oscillating until there is a change to the *Freq_Sel* signal. The conventional method [76] of verifying the "lock time" property is to check if the output of the low-pass filter has reached a new DC value within the lock time. Unlike such an approach that is dependent on the design under test, the proposed DTW method allows designers to work on the VCO simulation trace directly by finding the lock time and the minimum cost function associated with it.

Finding the Optimal Cost Function

The first step in finding the optimal cost, is to simulate the design and generate two sets of sequences as shown in Figure 5.12. The dotted/bold line represents the ideal/jitter signal, respectively. This is followed by generating the spectrogram of those two VCO signals as shown in Figure 5.13 because the total simulation trace of the VCO output has more than

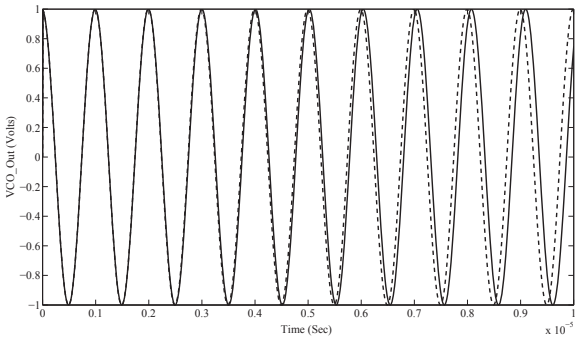


Figure 5.12: VCO Output

one million samples. This spectrogram of the VCO output with jitter will be compared with the spectrogram of the ideal output signal which has a constant frequency (horizontal line). We then apply the DTW algorithm to determine the minimum cost alignment between the two outputs. If the observed cost is very big with respect to the cost of the ideal output, it can be concluded that there is large deviation in the signal frequency compared with the ideal one.

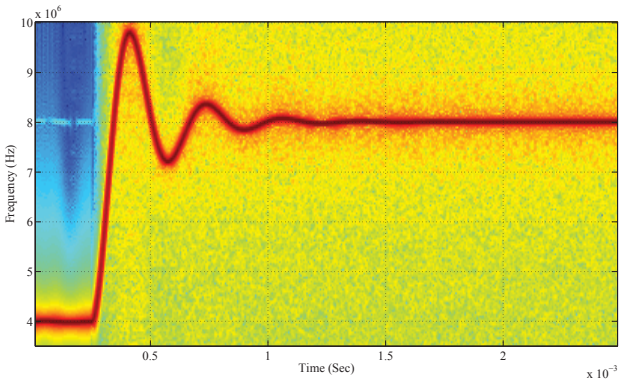


Figure 5.13: VCO Output Spectrogram

The lock time can now be determined by looking at the time when the minimum path calculated by the DTW algorithm crosses the diagonal as shown in Figure 5.14. This information is stored in the matrix *D(m,n)* and can be mapped directly to the corresponding time in the simulation trace. The novelty of such an approach lies in the fact that the DTW algorithm will not only classify outputs based on the frequency quality but can also

determine the value of the lock time. In this case, the lock time was determined to be *1.0944 ms*.

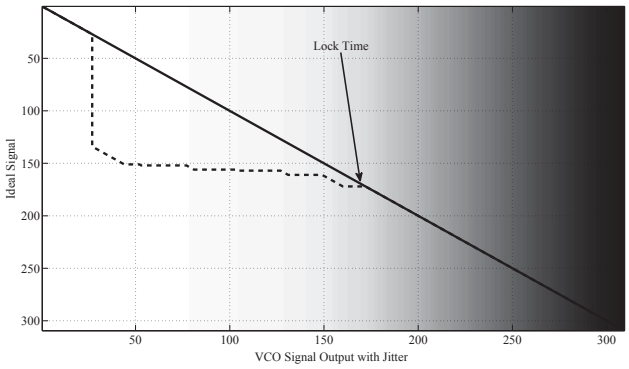


Figure 5.14: VCO Output Warped using DTW

Decision Based on Hypothesis Testing

Since the jitter is considered to be a random noise that has Gaussian distribution, we have performed MonteCarlo simulation for "1000" trials to evaluate the cost and then used hypothesis testing to reason about the results. For this kind of verification, one would be interested to know *"Whether for the given confidence level α, and M MonteCarlo trials, what is the region of acceptance and rejection of the circuit?"*

Table 5.3: DTW and Hypothesis Tesing Results for the PLL

J	Jitter Effect			Minimum Alignment Cost			Lock Time
	Mean	Variance	Acceptance Region	Mean	Variance	Acceptance Region	(ms)
1e-13	-2.73e-7	8.12e-5	[-1.338e-4 - 1.333e-4]	153.190	0.00544	[153.181 - 153.199]	1.0944
1e-12	-1.0032e-5	8.0281e-4	[0.00131 - -0.00133]	153.197	0.00748	[153.185 - 153.209]	1.0944
1e-11	-3.1265e-5	0.0081	[-0.01332 - 0.01326]	153.194	0.01492	[153.169 - 153.218]	1.0944
1e-10	-6.6129e-4	0.0810	[-0.13396 - 0.13264]	153.122	0.14995	[152.875 - 153.369]	1.0944
5e-10	0.0015	0.4081	[-0.6697 - 0.6728]	152.930	0.65085	[151.860 - 154.001]	1.0944
7e-10	-0.0044	0.5694	[-0.9410 - 0.9322]	153.059	0.99108	[151.429 - 154.689]	1.0944
9e-10	0.0068	0.7304	[-1.1946 - 1.2082]	153.443	1.16306	[151.529 - 155.355]	1.0944
1e-9	-0.0015	0.8215	[-1.3528 - 1.3497]	153.691	1.33621	[151.492 - 155.888]	1.2096
2e-9	-0.0035	1.6325	[-2.6887 - 2.6817]	158.179	2.36523	[154.288 - 162.069]	1.7472
3e-9	-0.01598	2.4520	[-4.0493 - 4.0174]	166.015	3.26501	[160.644 - 171.385]	—
5e-9	-0.01079	4.1609	[-6.8549 - 6.8333]	185.937	3.69045	[179.867 - 192.007]	—

As a part of the specification, designers have to specify the confidence interval and

the cost. This cost will be checked if it is in the acceptance region of tolerance of the jittery signal with respect to the ideal signal. Table 5.3 summarizes the results for different jitter "J" deviations. The table is derived by taking into account the jitter factor that represents the effect of jitter on the phase of the VCO output signal and are plotted as a normal probability function as shown in Figure 5.15.

The results show that the minimum cost is also following a normal distribution like the jitter noise in the VCO with different means and different deviations. When comparing Table 5.3 with the plot, we see that unlike the jitter "J" that has a zero mean, the mean of the jitter factor is small with an increasing variance. Also, the minimum cost alignment increases linearly when the jitter noise increases in terms of mean and variance.

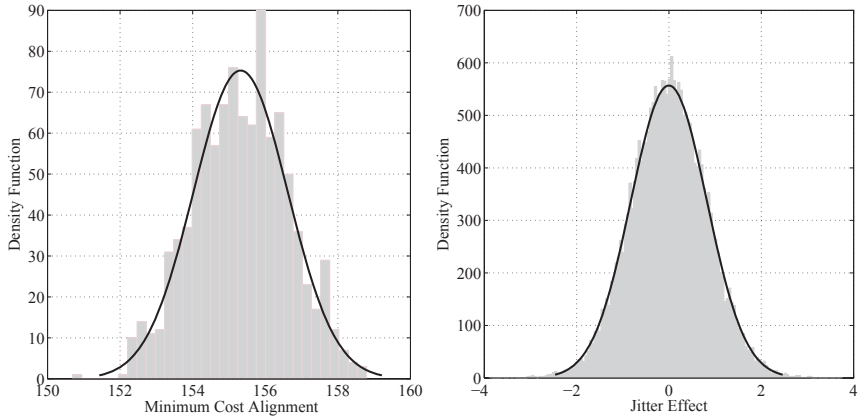


Figure 5.15: Influence of the Jitter on the Cost

We also see that when "J" is large, the PLL fails to lock as represented by the "dashed entry" in Table 5.3. The spectrogram in such a case will show the minimum path failing to cross the diagonal line. We use hypothesis testing to find the acceptance region for each "J" as shown in Table 5.3.

5.4 Summary

This chapter presented a pattern matching to account for thermal, shot, jitter noise, and process variation in an analog circuit. For noise, the idea is to apply SDE to model design and integrate device variation in a MATLAB simulation environment. The effect of noise and manufacturing constraints are analyzed by performing a statistical method by combining MonteCarlo simulation with two pattern matching algorithms (LCSS and DTW) and hypothesis testing to determine the probability of acceptance/rejection of the simulation traces. The efficiency of LCSS algorithm is illustrated on a Colpitts oscillator circuit to study the effect of noise and process conditions. The influence of jitter noise on the "lock-time" property of a PLL based frequency synthesizer is analyzed through DTW algorithm.

The conventional verification method may require major changes to the test-bench structure during scaling of analog designs, and still cannot answer the question: *"How do we choose the test set?"* or *"Can we retain the same test points?"* This is because, the test points are chosen in such a way that it represents the limit of operation of the design which of course may or may not change when the designs are scaled. However, DTW based techniques work on the simulation trace in polynomial time and hence it will be well suited for verifying "black-box" analog/mixed-signal (AMS) designs. Also, the use of spectrogram has provided an alternative solution to the memory usage problem faced by AMS design verification. The statistical environment provides designers with additional information about the acceptance region, thereby allowing them to make better decisions.

Chapter 6

Conclusions and Future Work

6.1 Conclusions

In this thesis, we have proposed modeling and verification approach of analog and mixed signal circuits with noise and process variation. The approach allows us to study some of the effects in a traditional analog design flow at the system level. The main idea is to use stochastic differential equation (SDE) to model thermal and shot noise and then integrate process variation in a MATLAB simulation environment. As the industrial verification environment relies on simulation, we believe that the methodology presented in this thesis can be quite useful for the performance evaluation of analog circuit for architecture exploration.

Towards the development of a successful SDE based noise analysis, the thesis mainly contributes in three directions.

1. Firstly, it presents a framework that allow us to model and verify the deterministic property in the presence of shot noise, thermal noise, and process variations. The idea is to use stochastic differential equations (SDE) to model noise in additive and multiplicative form and then combine process variation in a runtime verification environment. The practical effectiveness of the proposed framework is compared for

Colpitts oscillator, Tunnel Diode oscillator and a Phase Locked Loop (PLL) based frequency synthesizer circuit. We have shown that the properties that are satisfied without noise have failed in the presence of noise and process variation, thereby making the method efficient in finding bugs.

2. Secondly, a framework is presented to model and verify the statistical property of an analog designs in the presence of shot noise, thermal noise, and process variations. The idea is to use SDEs to model noise in additive and multiplicative form and then combine process variation in a statistical runtime verification environment. Statistical run-time verification combines hypothesis testing and MonteCarlo/Bootstrap simulation for monitoring the statistical behavior in an analog circuit. To illustrate the practical effectiveness of the proposed framework, the efficiency of MonteCarlo and Bootstrap statistical techniques are compared for Colpitts oscillator, Band-Gap reference generator and a PLL based frequency synthesizer circuit.

3. Thirdly, the thesis presents a methodology that relies on two different pattern matching algorithms for the qualitative estimation of analog circuits with noise and process variation. The first algorithm, is the *longest closest subsequence* (LCSS), a variant of the longest common subsequence (LCS) that is effective in estimating the percentage of matching between an ideal and non-ideal circuit simulation traces influenced by noise and process variations. The second algorithm is a modification of *dynamic time warping* (DTW) that is used to handle instabilities in the simulation traces due to jitter conditions. The underlying idea of both these algorithm is to find the subsequence simulation trace between a set of analog/mixed-signal traces and then use the combination of MonteCarlo and hypothesis testing to determine the probability of acceptance/rejection of those traces.

The successful handling of these diverse simulation methodologies clearly demonstrates its feasibility for real-world industrial designs. We believe that the foundation set in the thesis is the first step towards analyzing noise and process variation at a higher level of abstraction. The main limitation of the proposed approach is that the models used for analog circuits are primitive with a trade-off in accuracy. To overcome this, we may need to use complex models for circuit elements. In addition, the simulation based methods are deemed inaccurate for safety critical applications. Because of this, the proposed approach should be complemented with formal techniques such as theorem proving and model checking, which can prove to be very useful when precision of the results is of prime importance.

6.2 Future Work

The results presented in this thesis open new avenues in using SDE based methodology for the verification of AMS designs. Building on our results, more features can be added to strengthen the capabilities of the methodology to handle complex designs. Some of the future extensions are outlined below.

1. The simulation at the circuit level using SPICE incorporates complex models that are considered highly accurate. These standard compact models (*BSIM3v3, Philips LEVEL9 or EKV*) allow many nanotechnology process variation on device parameters. Some of the parameters such as, the variation in oxide thickness (t_{ox}), threshold voltage (V_t), aging may considerably affect the performance of AMS designs. In order to have a robust design, it is necessary that the verification environment is mature enough to handle such complex issues. Hence, it is necessary to look into developing models that could be integrated with the verification methodology presented in this thesis.

2. Extraction of the SDE equations from the spice netlist descriptions is an area that needs to be explored. We have tools that can extract ODEs from the spice netlist, and we need to investigate how that could be extended for SDE extraction automatically. In addition, developing higher order numerical approximation in the form of Taylor series has to be explored.

3. The formalization and verification of AMS design is an interesting direction. The theories and infrastructure developed in the context of higher-order logic (HOL) [75] have used random variables to verify the statistical properties of probabilistic systems [108]. Due to the stochastic nature of noise, it would be intriguing to use HOL to develop an infrastructure for noise. Formal approaches such as model checking have theories associated with discrete and continuous Markov Chains [62]. As SDEs are considered to be a Markov process, a research direction to use reachability analysis has to be explored.

4. Both the quantitative and qualitative approaches can be extended to accommodate evidential reasoning methods such as Multi-Value Attribute Theory [19] or Hierarchical Analysis [51]. These methods would help us to extract a rank from a pool of simulation results, on the basis of the qualitative/quantitative impact of the results.

Appendix A

AnalogSDE: A Verification Tool for Analog Circuits

As a part of the thesis, we have developed a tool for automatic verification of analog design with noise and process variation. The following are the features of the proposed tool:

1. Support automatic generation of thermal, shot noise in "additive" and "multiplicative" forms.
2. Probability Distribution for Circuit Parameters to study the effect of process variations
3. Run-Time Verification for Online Monitoring Noise
    (a) Assertion Based Verification
    (b) Statistical Based Verification
        i. MonteCarlo Based Hypothesis Testing
        ii. Bootstrap Based Hypothesis Testing
4. Formal Verification Using MetiTarski

Figure A.1 shows the tool framework that integrates formal and semi-formal verification in the MATLAB environment.

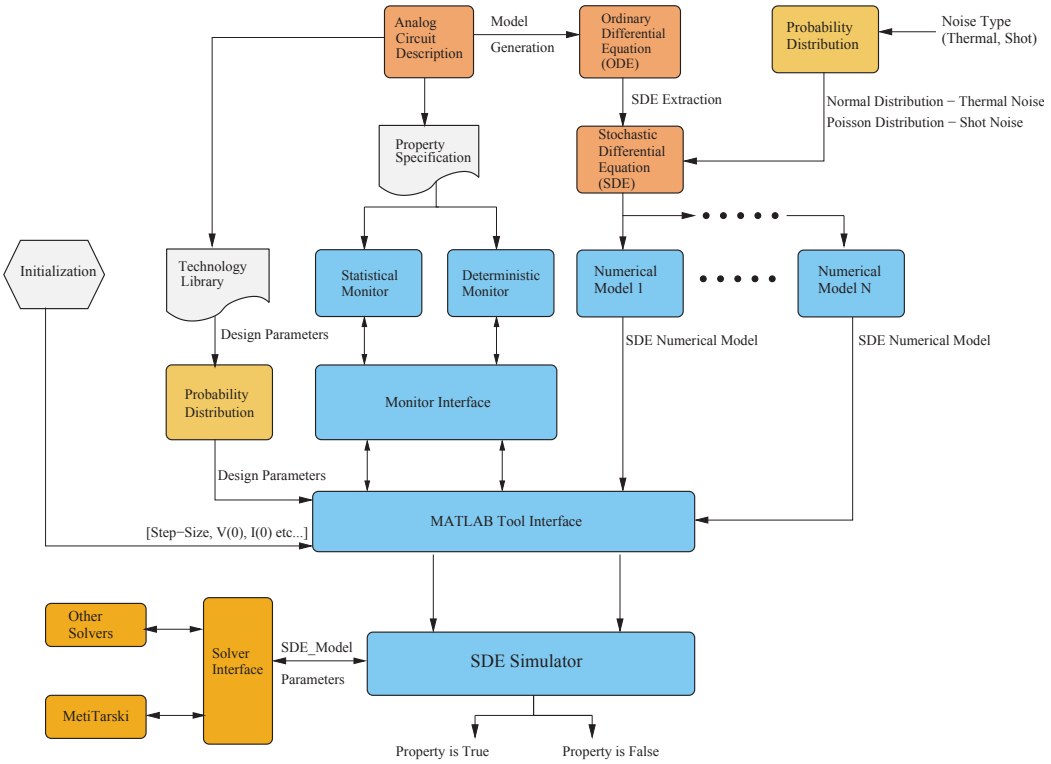


Figure A.1: Overview of the Noise Analysis and Verification Framework

Thereafter, given an analog design described as a system of *ODEs*, the idea is to include a stochastic process that describes the noise behavior. Since there are no functions that can automatically incorporate stochastic processes, we manually generate the *SDEs*. Depending on the type of process and technology library, various design parameters in the circuit are calculated using different probability distributions. We have used $0.18\mu m$ CMOS technology to evaluate the tool for different benchmark circuits.

The SDE simulator is a MATLAB decision procedure for simulating the design. The input to the SDE simulator is the SDE numerical model, design parameters and the property to be monitored. The design parameters may include the amplitude of the noise, initial conditions of the circuit current and voltages, step size, and simulation cycle. The property

of interest could be monitored using *deterministic* or *statistical* monitors, based on user selection at the SDE simulator level. All communications to the monitors occur through the *monitor interface*, which is a decision procedure that controls the monitor selection and data paths.

The *deterministic* monitors are based on finite-state machine and is implemented as a simple assertions. The details of which can be found in Chapter 3. In contrast, the *statistical* monitors combines hypothesis testing procedures and different statistical technique (MonteCarlo and Bootstrap) to verify the statistical property of the design. Please refer to Chapter 4 for more details. In general, for any given simulation run, the user can generate various simulation and histogram plots to monitor the property of interest.

On the formal verification side, we integrate external solvers such as automated theorem prover (MetiTarski) into the simulation environment. The idea is to formally verify the numerical model for every simulation step size by calling the external solver through the *solver interface*. The SDE simulator engine passes the required parameters (simulation time, step-size, design parameters, etc.) to MetiTarski. For MetiTarski, the properties of interest is described as inequalities over special functions. If MetiTarski is successful, it delivers a proof and we are done. If unsuccessful, it will run until terminated by the timer in the SDE simulator. At this point, the SDE simulator has to decide if the property could be verified using other solvers or to report a bug.

We have applied the tool framework on several benchmark circuits such as Schmitt trigger, Colpitts and Tunnel Diode oscillators and the results are presented in Chapters 3 and 4.

Figure A.2 shows the corresponding class diagram of the AnalogSDE tool framework.

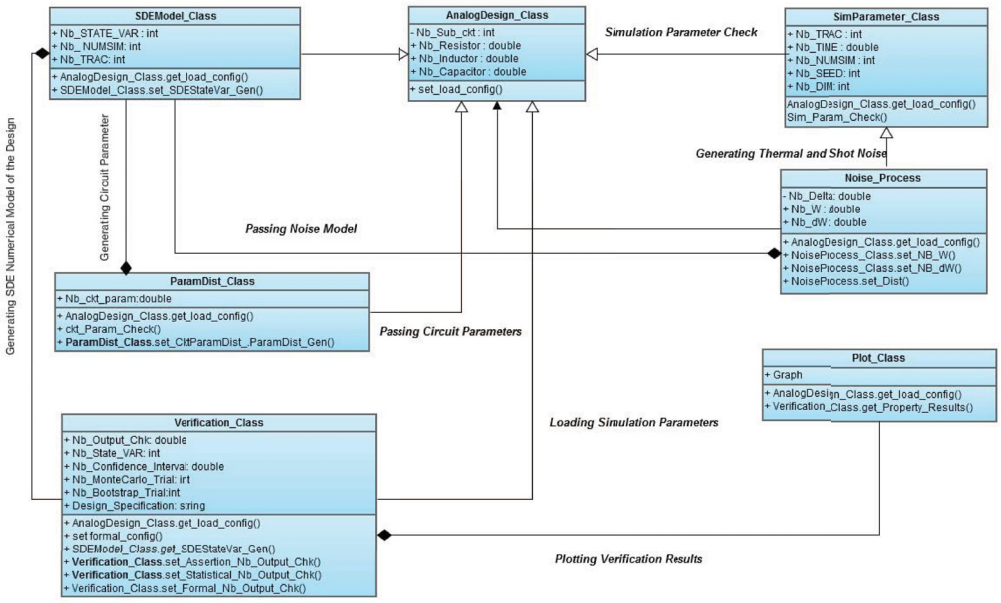


Figure A.2: Class Diagram for AnalogSDE Tool Framework.

The classes are developed using MATLAB based object-oriented programming [123]. The class diagram can be described as follows:

1. **AnalogDesign_Class:** This represents the overall design of the analog circuit that includes defining circuit and simulation parameters. $Load\_Config$ file is used to get all the parameters associated with the design. This may include the amplitude of the noise, initial conditions of the circuit current and voltages, step size, and simulation cycle. Also, the configuration file also includes the number of state variables in the design.

2. **SimParameter_Class:** This class has an built-in checker to verify if the loaded simulation parameters meet the simulation standards. For instance, to check if the dimension of the noise DIM is a positive integer, the checker does the following

```
if(Param_obj.DIM <=0)
  error('The Number of Noise Source must
```

```
            be a positive integer');
    end
```

3. **Noise_Process:** This class generates the thermal and shot noise based on gaussian and poisson white noise distribution. The input to this class is the number of noise sources (DIM) and the simulation step-size ($Delta$). Based on this, it generates multi-dimensional noise.

4. **ParameterDist_Class:** The input to this class is the design parameter along with the $Load_Config$ file. Based on the technology, this class uses probability distribution to generate circuit parameters. The technology information is provided by the user in the $Load_Config$ file.

5. **SDEModel_Class:** This is the class where the input from the noise modeling and simulation of the design is carried out. The model is based on the number of state variables $STATE_VAR$ in the design. The model is simulated along with process variation from the $ParameterDist\_Class$ and parameters from the $SimParameter\_Class$. The results are then sent to the verification environment.

6. **Verification_Class:** Both assertion and statistical based verification are carried out within this class. The user has to specify the type of verification "ASSERT" for ABV or "STAT_Mont" for MonteCarlo based statistical verification or "STAT_BS" for MonteCarlo based statistical verification. In addition, the class requires the property (deterministic or statistical) to be specified as a function as given by $"Verification\_Class.set_Assertion\_Nb\_Output\_Chk()"$ for ABV or $"Verification\_Class.set\_Statistical\_Nb_Output\_Chk()"$ for statistical verification. Formal approach can also be done using this verification class. This require the user

to specify "Formal" in the $Load\_Config$ and the user has to work in an Unix environment.

7. **Plot_Class:** Using this class, the user can plot histograms and other 2-D plots.

Bibliography

[1] *0.18μm* CMOS Fabrication Process. `http://www.tsmc.com`, 2008.

[2] K. S. Abe, and W. T. Shaw. Measure Order of Convergence Without an Exact Solution, Euler Vs Milstein Schme. International Journal of Pure and Applied Mathematics. (24)3: pp. 365-381, 2005.

[3] D. Abercrombie, B. Koenemann. Process/Design Learning from Electrical Test. IEEE/ACM International Conference on Computer-Aided Design, pp. 733-738, 2004.

[4] A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu. Bounded Model Checking. Advances in Computers. Academic Press, 58:118-149, 2003.

[5] Accellera, Verilog-AMS Language Reference Manual Analog & Mixed-Signal Extensions to Verilog-HDL. `http://www.designers-guide.org/`, 2003.

[6] Accellera Property Specification Language. `http://www.accellera.org/`, 2004.

[7] B. Akbarpour and L. C. Paulson. MetiTarski: An Automatic Prover for the Elementary Functions, In Intelligent Computer Mathematics, LNCS 5144, pp. 217231, Springer, 2008.

[8] B. Akbarpour and L. C. Paulson. Applications of MetiTarski in the Verification of Control and Hybrid Systems, In Hybrid Systems: Computation and Control, LNCS 5469, pp. 1-15, Springer, 2009.

[9] H. Al-Junaid, T.Kazmierski. HDL Models of Ferromagnetic Core Hysteresis Using Timeless Discretisation of the Magnetic Slope. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 25(12): pp. 2757-2764, 2006.

[10] G. Al-Sammane. Simulation Symbolique des Circuits Decrits au Niveau Algorithmique. PhD thesis, Universitè Joseph Fourier, 2005.

[11] G. Al Sammane, M. Zaki, and S. Tahar. A Symbolic Methodology for the Verification of Analog and Mixed Signal Designs. IEEE/ACM Design Automation and Test in Europe, pp. 249-254, April 2007.

[12] G. Al Sammane, M.H. Zaki, Z.J. Dong and S. Tahar. Towards Assertion Based Verification of Analog and Mixed Signal Designs Using PSL. Forum on Specification & Design Languages, pp. 293-298, 2007.

[13] Analoginsydes. The Intelligent Symbolic Design System for Analog Circuits. `http://www.analog-insydes.de/`, 2010.

[14] B. Ankele,W. Hölzl, and P. O'Leary. Enhanced MOS parameter extraction and SPICE modeling for mixed signal analogue and Digital Circuit Simulation. IEEE International Conference on Microelectronic Test Structures, pp. 133137, 1989.

[15] P. Antognetti and G. Massobrio. Semiconductor Device Modeling with SPICE, McGraw-Hill, New York, 1988.

[16] A. Aziz, K. Sanwal, V. Singhal, and R.K. Brayton. Verifying Continuous Time Markov Chains. In Computer Aided Verification, LNCS 1102, pp.269-276, Springer, 1996.

[17] Z. Bai. Krylov Subspace Techniques for Reduced-Order Modeling of Large-Scale Dynamical Systems. Journal of Applied Numerical Mathematics. 43(1): pp. 9-44, 2002.

[18] H. B. Bakoglu. Circuits Interconnects, and Packaging for VLSI. Addison Wesley, 1990.

[19] V. Belton, and T. J. Stewart. Multiple Criteria Decision Analysis: An Integrated Approach. Kluwer Academic Publishers, 2002.

[20] Berkeley Design Automation, Inc. Efficient Noise Analysis for Complex Non-Periodic Analog/RF Blocks. http://www.berkeley-da.com/prod/datasheets/Berkeley_DA_Noise_Analisys_Analog_RF_WP.pdf, 2009.

[21] R. Best. Phase Locked Loops: Design, Simulation, and Applications, McGraw-Hill, 2007

[22] P. Bolcato and R. Poujois. A New Approach for Noise Simulation in Transient Analysis. IEEE International Symposium on Circuits and Systems, pp. 887-890, 1992.

[23] T.E. Bonnerud, B. Hernes, T. Ytterdal. A Mixed-signal Functional Level Simulation Framework based on SystemC for System-on-a-Chip Applications. IEEE Custom Integrated Circuits, pp. 541-544, 2001.

[24] F. Bouchhima, M. Brirel, G. Nicolescu1, M. Abid, E. M. Aboulhamid. A SystemC/Simulink Co-Simulation Framework for Continuous/Discrete-Events Simulation, IEEE Behavioral Modeling and Simulation, pp. 1-6, 2006.

[25] Cadence Design Systems. Using a SoC Functional Verification Kit to Improve Productivity, Reduce Risk, and Increase Quality. White Paper, http://w2.cadence.com/whitepapers/SoC_fv_Kit_wp.pdf, 2007.

[26] Cadence Design Systems. The Role of Assertions in Verification Methodologies-Using Assertions in a Simulation Environment. White Paper, http://www.cadence.com/rl/Resources/application_notes/CDN_Assertions_in_Verification_Methodologies.pdf, 2003.

[27] F. E. Cellier and A. Nebot, The Modelica Bond Graph Library. Swiss Federal Institute of Technology, Technical Report, 2007.

[28] Celoxia Inc. http://www.celoxica.com/, 2008.

[29] W. K. Chen, The Circuits and Filters Handbook. CRC Press LLC, New York, 2006.

[30] Y. Cheng. The Influence and Modeling of Process Variation and Device Mismatch on Analog/RF Circuit Design. IEEE International Caracas Conference on Devices, Circuits and Systems, pp. 1-8, 2002.

[31] M. R. Chernick. Bootstrap Methods, A Practitioner's Guide. Wiley Series, 1999.

[32] L. O. Chua, Chuas Circuit : An Overview Ten Years Later, Journal of Circuits, Systems and Computers, Vol. 4, pp. 117159, 1994.

[33] A. Chutinan, B. H. Krogh. Computational Techniques for Hybrid System Verification. IEEE Transaction on Automotive and Control 48(1): pp. 6475, 2003.

[34] E. Clarke, A. Donze and A. Legay. Statistical Model Checking of Mixed-Analog Circuits With an Application to a Third-Order Delta-Sigma Modulator. In Hardware and Software: Verification and Testing, LNCS. 5394, pp. 149-163, Springer, 2008.

[35] E.M. Clarke, O. Grumberg, and D.A. Peled. Model Checking. MIT press, December 1999.

[36] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. Introduction to Algorithms, The MIT Press, 2001.

[37] Dassault Systemes, The Dymola Modelling Laboratory. `http://www.dymola.com/index.htm`, 2011.

[38] W. B. Davenport and W. L. Root. An Introduction to the Theory of Random Signals and Noise. IEEE Press, 1987.

[39] A. Demir, E. Liu, A. Vincentelli, and I. Vassiliou. Behavioral Simulation Techniques for Phase/Delay Locked Systems. IEEE Custom Integrated Circuits Conference, pp. 453-456, 1994.

[40] A. Demir. Analysis and Simulation of Noise in Nonlinear Electronic Circuits and Systems. Ph.D. dissertation, University of California, Berkeley, 1997.

[41] A. Demir, A. Mehrotra and J. Roychowdhury. Phase Noise in Oscillators: A Unifying Theory and Numerical Methods for Characterization. IEEE Transactions on Circuits and Systems-II, Vol.47, pp. 655674, 2000.

[42] W. Denman, Towards the Automated Modelling and Formal Verification of Analog Designs. M.A.Sc Thesis, Dept of ECE, Concordia Univesity, Montreal, Canada, April 2009.

[43] A. Dunlop, A. Demir, P. Feldmann, S. Kapur, D. Long, R. Melville and J. Roychowdhury. Tools and Methodology for RF IC Design, IEEE International Conference on Computer-Aided Design, pp. 414-420, 1998.

[44] L. Feng. Review of Model Order Reduction Methods for numerical Simulation of Nonlinear Circuits. Applied Mathematics and Computation. 167(1): pp. 576-591, 2005.

[45] I. M. Filanovsky, C. J. M. Verhoeven, and M. Reja. Remarks on Analysis, Design and Amplitude Stability of MOS Colpitts Oscillator. IEEE Transactions on Circuits and Systems-II, (54)9: pp. 800-804, 2007.

[46] H. D. Foster, A. C. Krolink. Creating Assertion-Based IP (Integrated Circuits and Systems). Springer, 2010.

[47] $f$reeda^{TM}: `http://www.freeda.org/`, 2011.

[48] G. Frehse. PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech. In Hybrid Systems: Computation and Control, LNCS.3414, pp. 258273, Springer, 2005.

[49] J. E. Freund. Modern Elementary Statistics. Prentice hall, 1984.

[50] A. Ghosh, R. Vemuri, and D. R. Vemuri. Formal Verification of Synthesized Analog Designs. IEEE International Conference on Computer Design. pp. 4045, 1999.

[51] J. Gordon and E. H. Shortllife. A Method for Managing Evidential Reasoning in a Hierarchical Hypothesis Space. Technical Report, Stanford University, `ftp://reports.stanford.edu/pub/cstr/reports/cs/tr/84/1023/CS-TR-84-1023.pdf`, 1984.

[52] P. A. Gray, P. J. Hurst, S. H. Lewis and R. G. Meyer. Analysis and Design of Analog Integrator Circuits. Wiley, 2009.

[53] M. Greenstreet. Verifying Safety Properties of Differential Equations. In Computer Aided Verification, LNCS.1102, pp. 277287, Springer, 1996.

[54] M. Greenstreet, I. Mitchell. Integrating Projections. In HybridSystems: Computation and Control. LNCS. 1386, pp. 159174, Springer, 1998.

[55] M. Greenstreet, and S. Yang. Verifying Start-Up Conditions for a Ring Oscillator, ACM Great Lakes Symposium on VLSI, pp. 201-206, 2008.

[56] C.M. Grinstead and J.L. Snell. Introduction to Probability. American Mathematical Society, 1997.

[57] S. Gupta, B. H. Krogh, and R. A. Rutenbar. Towards Formal Verification of Analog Designs, IEEE/ACM International Conference on Computer Aided Design. pp. 210-217, 2004.

[58] W. Haas, U. Heinkel, H. Braisz, T. Gentner, M. Padeffke, T. Buerner, G. Alexander and F. Alexander. The VHDL Reference: A Practical Guide to Computer-Aided Integrated Circuit Design Including VHDL-AMS, Wiley, 2000.

[59] A. Hajimiri, S. Limotyrakis and T. H. Lee. Jitter and Phase Noise in Ring Oscillators. IEEE Journal of Solid-State Circuits, 34(6): pp. 790-804, 1999.

[60] R. J. Haller. The Nuts and Bolts of Signal-Integrity Analysis. Electronics Design, Strategies, News (EDN), 2000

[61] F. Herzel and B. Razavi. A Study of Oscillator Jitter Due to Supply and Substrate Noise. IEEE Transactions on Circuits and Systems-II, 46(1):56-62, 1999.

[62] A. Hinton, M. Kwiatkowska, G. Norman and D. Parker. PRISM: A Tool for Automatic Verification of Probabilistic Systems. International Conference on Tools and Algorithms for the Construction and Analysis of Systems, LNCS.3920, pp. 441-444, Springer, 2006.

[63] M. Horowitz. Digital Analog Design. Workshop on Frontiers in Analog Circuit Synthesis and Verification. 2011.

[64] R. Hum, Where are the dragons? Workshop on Frontiers in Analog Circuit Synthesis and Verification, 2011.

[65] IO Methodology Inc. SimDE^{TM} Waveform. `http://www.iometh.com/Product/SignalMeth/index.html`, 2012.

[66] D. A. Johns and K. Martin. Analog Integrated Circuit Design. Wiley, 1997.

[67] K. D. Jones, J. Kim, and V. Konrad. Some "Real World" Problems in the Analog and Mixed Signal Domains. International Workshop on Designing Correct Circuits, pp. 51-59, 2008.

[68] W. G. Kelley, and A. C. Peterson. Difference Equations: An Introduction with Applications. Academic Press, 2001.

[69] M. Kennedy. Chaos in the Colpitts Oscillator. IEEE Transactions on Circuits and Systems, 41(11): pp. 771774, 1994.

[70] C. Kim, E. K. Lee, P. H$\ddot{a}$nggi, and P. Talkner. Numerical Method for Solving Stochastic Differential Equations with Poissonian White Shot Noise. Physical Review E. (76)1:1-10, 2007.

[71] P. E. Kloden and E. Platen. Numerical Solution of Stochastic Differential Equations. Springer, 1995.

[72] Z. Kohavi. Switching and Finite Automata Theory. McGraw-Hill, 1978.

[73] E. Kolarova. Modelling RL Electrical Circuits by Stochastic Diferential Equations. International Conference on Computer as a Tool, pp. 1236-1238, 2005.

[74] N. M. Kriplani, A. Victor and M. B. Steer. Time-Domain Modelling of Phase Noise in an Oscillator. European Microwave Conference, pp. 514-517, 2006.

[75] T. Kropf. Introduction to Formal Hardware Verification, Springer, 2000.

[76] K. Kundert. Predicting the Phase Noise and Jitter of PLL-Based Frequency Synthesizers. https://www.designers-guide.com, 2003.

[77] K. Kundert. The Designers Guide to SPICE and Spectre. Kluwer Academic Publishers, 1995.

[78] K. Kundert, H. Chang, D. Jefferies, G. Lamant, E. Malavasi, F. Sendig. Design of Mixed-signal Systems-on-a-chip, IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, 19(12):1561-1571, 2000.

[79] K. Kundert, H. Chang. Top-Down Design and Verification of Mixed-Signal Circuits. Designers Guide Consulting, http://www.designers-guide.com/docs/tddv.pdf, 2009.

[80] K. Kundert and H. Chang. Verification of Complex Analog Integrated Circuits. IEEE Custom Integrated Circuits Conference, pp.177-184, 2006.

[81] K. Kundert, J. K. White and A. Sangiovanni-Vincentelli. Steady-State Methods for Simulating Analog and Microwave Circuits, Kluwer Academic Press, 1990.

[82] R. P. Kurshan, K. L. McMillan. Analysis of Digital Circuits Through Symbolic Reduction. IEEE Trans. Computer-Aided Design. 10(11): pp. 13501371, 1991.

[83] M. Kwiatkowska, G. Norman and D. Parker. Stochastic Model Checking. Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation, Vol. 4486: pp 220-270, Springer, 2007.

[84] O. Lahiouel, H. Aridhi, M. H. Zaki, and S. Tahar. Tool for Modeling and Analysis of Electronic Circuits and Systems. Technical Report, Dept. of ECE, Concordia University, Montreal, http://hvg.ece.concordia.ca/Publications/TECH_REP/TMAES.pdf, 2011.

[85] E. L. Lehmann, J. P. Romano. Testing Statistical Hypotheses. Springer. 2005.

[86] L. L. Lewyn, T. Ytterdal, C. Wulff, and K. Martin. Analog Circuit Design in Nanoscale CMOS Technologies. Proceedings of the IEEE, 95(10):1687-1714, 2009.

[87] M.P. Li. Jitter, Noise, and Signal Integrity at High-Speed. Prentice Hall, 2007.

[88] L. Ling and W. Burleson. Analysis and Mitigation of Process Variation Impacts on Power-Attack Tolerance. IEEE/ACM Design, Automation Conference, pp.238-243, 2009.

[89] S. Little, D. Walter, K. Jones, and C. Myers. Analog/Mixed-Signal Circuit Verification Using Models Generated from Simulation Traces. In Automated Technology for Verification and Analysis, LNCS 4762, pp.114128, Springer, 2007.

[90] R. Ludwig and P. Bretchko. RF Circuit Design, Theory and Applications. Pearson Education, 2004.

[91] O. Maler and D. Nickovic. Monitoring Temporal Properties of Continuous Signals. In Formal Modelling and Analysis of Timed Systems, LNCS 3253, 152-166, Springer, 2004.

[92] S. A. Maas, Nonlinear Microwave and RF Circuits. Artech House, 2003.

[93] S. K. Magierowski and S. Zukotynski. CMOS LC-Oscillator Phase-Noise Analysis Using Nonlinear Models. IEEE Transaction on Circuits and Systems, 51(4): 664-677, 2004.

[94] X. Mao, H. Yang, and H. Wang. Behavioral Modeling and Simulation of Jitter and Phase Noise in Fractional-N PLL Frequency Synthesizer. IEEE Behavioral Modeling and Simulation Conference, pp. 25-30, 2004.

[95] D. E. Martin, P. A. Wilsey, R. J. Hoekstra, E. R. Keiter, S. A. Hutchinson, T. V. Russo, and L. J. Waters. Integrating Multiple Parallel Simulation Engines for Mixed-technology Parallel Simulation, IEEE Simulation Symposium. pp. 45-52, 2002.

[96] W. L. Martinez and A. R. Martinez. Computational Statistics Handbook with MATLAB. Chapman & Hall/CRC, 2002.

[97] W. Mathis, and T. Thiessen. On Noise Analysis of Oscillators Based on Statistical Mechanics. International Conference on Mixed Design of Integrated Circuits & Systems, pp. 472-477, 2009.

[98] K. Mayaram, D. C. Lee, S. Moinian, D. A. Rich, and J. Roychowdhury. Computer-Aided Circuit Analysis Tools for RFIC Simulation: Algorithms, Features, and Limitations. IEEE Transactions on Circuits and Systems- II: Analog and Digital Signal Processing, 47(4), April 2000.

[99] J. A. McNeill. Jitter in Ring Oscillators. IEEE Journal of Solid-State Circuits, 32(6): pp. 870-879, 1997.

[100] L. Mendonc-a de Moura, B. Dutertre, N. Shankar. A Tutorial on Satisfiability Modulo Theories. In Computer Aided Verification. LNCS.4590, pp.2036, Springer, 2007.

[101] R. Meolic, T. Kapus and Z. Brezocnik. CTL and ACTL patterns. International Conference on Trends in Communications. Vol. 2, pp. 540-543, 2001.

[102] R. G. Miller. The Jackknife– A Review. Biometrika, (61)1: pp. 1-15, 1974.

[103] K. Morin-Allory, L. Fesquet, B. Roustan, and D. Borrione. Asynchronous Online-Monitoring of Logical and Temporal Assertions. Embedded Systems Specification and Design Languages, LNEE. 10, pp. 243-253 Springer, 2008.

[104] R. Narayanan, A. Daghar, M. Zaki, and S. Tahar: Using LCSS Algorithm for Circuit Level Verification of Analog Designs. Technical Report, Department of Electrical and Computer Engineering, Concordia University, February 2012.

[105] E. Naviasky, and M. Nizic. Cadence Design Services. Mixed-Signal Design Challenges and Requirements. `http://www.cadence.com/rl/Resources/white_papers/mixed_signal_challenges_wp.pdf`, 2009.

[106] H. Nyquist. Thermal Agitation of Electric Charge in Conductors, Physical Review Letter, (32)1: pp. 110-113, 1928.

[107] B. Oksendal. Stochastic Differential Equations: An Introduction with Applications. Springer, 2000.

[108] Osman Hassan. Formal Probabilistic Analysis using Theorem Proving, PhD Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada, April 2008.

[109] P. Paper, M. Jamal Deen and O. Marinov. Noise in Advanced Electronic Devices and Circuits. AIP International Conference on Noise in Physical Systems and 1/f Fluctuations, Vol.780, pp. 3-12, 2005.

[110] F. Pcheux, C. Lallement, A. Vachoux. VHDL-AMS and Verilog-AMS as Alternative Hardware Description Languages for Efficient Modeling of Multidiscipline Systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 24(2): pp.204-225, 2005.

[111] M. H. Perrott, M. D. Trott, and C. G. Sodini. A Modeling Approach for $\Delta\Sigma$ Fractional-N Frequency Synthesizers Allowing Straightforward Noise Analysis. IEEE Journal of Solid State Circuits, 37(8): pp. 1028-1038, 2002.

[112] A.S. Priya, and P. Vaya. Modeling of Sigma-Delta Modulator Non-Idealities in MATLAB/SIMULINK. IEEE International Conference on Communication Systems and Network Technologies, pp. 310-315, 2011.

[113] PVS Specification and Verification System. `http://pvs.csl.sri.com/`, 2009.

[114] QEPCAD - Quantifier Elimination by Partial Cylindrical Algebraic Decomposition, `http://www.usna.edu/Users/cs/qepcad/B/QEPCAD.html`, 2009.

[115] L. R. Rabiner and B. Juang. Fundamentals of Speech Recognition. Prentice-Hall, Inc., 1993.

[116] T. K. Rawat, A. Lahiri, and A. Gupta. Noise Analysis of Single-Ended Input Differential Amplifier using Stochastic Differential Equation. International Journal of Computer, Information, and Systems Science, and Engineering, 2(3):192-196, 2008.

[117] R. Rohrer, L. Nagel, R. G. Meyer and L.Weber, Computationally Efficient Electronic Circuit Noise Calculation. IEEE Journal of Solid-State Circuits Society, (6)4: pp. 204-213, 1971.

[118] H. Sakoe, and S. Chiba. Dynamic Programming Algorithm Optimization for Spoken Word Recognition, IEEE Transactions on Acoustics, Speech and Signal Processing, 26(1), pp. 43- 49, 1978.

[119] D. Stirzaker. Elementary Probability. Cambridge Press, 2003.

[120] Synopsys HSPICE User Guide: RF Analysis. `http://www.synopsys.com`, 2009.

[121] SystemC-AMS User Community. `http://www.systemc-ams.org`, 2008

[122] W. E. Thain Jr *et. al*. Simulating Phase Noise in Phase Locked Loops with a Circuit Simulator. IEEE International Symposium on Circuits and Systems, Vol. 3, pp. 1760-1763, 1995.

[123] The Mathworks Inc. MATLAB User Guide. `http://www.mathworks.com/`, 2011.

[124] M. Bühler, J. Koehl, J. Bickford, J. Hibbeler, R. Sommer, M. Pronath, and A. Ripp. DFM/DFY Design for Manufacturability and Yield - Influence of Process Variations in Digital, Analog and Mixed-Signal Circuit Design. IEEE/ACM Design, Automation and Test in Europe, pp. 387 - 392, 2006.

[125] M. Vasilevski, F. Pecheux, H. Aboushady, and L. de Lamarre. Modeling Heterogeneous Systems Using SystemC-AMS Case Study: A Wireless Sensor Network Node. IEEE Behavioral Modeling and Simulation Workshop, pp.1-6, 2007.

[126] V. Vasudevan. A Time-Domain Technique for Computation of Noise-Spectral Density in Linear and Nonlinear Time-Varying Circuits. IEEE Transaction on Circuits and SystemsI: Vol. 51(2), 2004.

[127] S. Vijayaraghavan and M. Ramanathan. A Practical Guide for SystemVerilog Assertions. Springer, 2005.

[128] C. Yan, M. Greenstreet. Circuit-Level Verification of a High-Speed Toggle. IEEE International Conference on Formal Methods in Computer-Aided Design, pp.199206, 2007.

[129] M. Yoshikawa and H. Terai. Constraint-Driven Floorplanning based on Genetic Algorithm. ACM International Conference on Computer Engineering and Applications, pp.147-151, 2007

[130] H.L.S. Younes and R.G. Simmons. Probabilistic Verification of Discrete Event Systems Using Acceptance Sampling. In Computer Aided Verification, LNCS 2404, pp. 23-39, Springer, 2002.

[131] W. Yu, and B. H. Leung. Noise Analysis for Sampling Mixers Using Stochastic Differential Equations. IEEE Transactions on Circuits and Systems- II: Analog and Digital Signal Processing, Vol. 46(6), 1999.

[132] J. Yuan, C. Pixley, and A. Aziz Constraint-Based Verification, Springer, 2006.

[133] D. Walter, S. Little, N. Seegmiller, C. Myers and T. Yoneda. Symbolic Model Checking of Analog/Mixed-Signal Circuits. IEEE Asia and South Pacific Design Automation Conference, pp.316323, 2007.

[134] P. Wambacq, P. Dobrovolny, S. Donnay, M. Engels and I. Bolsens. Compact Modeling of Nonlinear Distortion in Analog Communication Circuits. IEEE/ACM Design, Automation and Test in Europe, pp.350-354, 2000.

[135] Z. Wang, N. Abbasi, R. Narayanan, M. Zaki, G. Sammane and S. Tahar. Verification of Analog and Mixed Signal Designs using On-line Monitoring, IEEE Mixed-Signals, Sensors, System Test Workshop, pp. 1-6, 2009.

[136] Z. Wang, M. H. Zaki, and S. Tahar. Statistical Runtime Verification of Analog and Mixed Signal Designs. IEEE International Conference on Signals, Circuits and Systems. pp. 1-6, 2009.

[137] J. F. Witte, K .A. A. Makinwa and J. H. Huijsing: Dynamic Offset Compensated CMOS Amplifiers. Analog Circuits and Signal Processing, Springer, 2009.

[138] C. F. J. Wu. Jackknife, Bootstrap and Other Resampling Methods in Regression Analysis. Annals of Statistics. (14)4: pp. 1261-1295, 1986.

[139] M. Zaki. Techniques for the Formal Verification of Analog and Mixed- Signal Designs, PhD Thesis, Department of Electrical and Computer Engineering, Concordia University, 2008.

[140] M. Zaki, S. Tahar and G. Bois. Formal Verification of Analog and Mixed Signal Designs: A Survey. *Microelectronics Journal*, Elsevier, 39(12): 1395-1404, 2008.