Uncertainty Propagation in Transistor-level Statistical Circuit Analysis

Qian Ying Tang

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2011-66
http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-66.html

May 18, 2011

Uncertainty Propagation in Transistor-level Statistical Circuit Analysis

By

Qian Ying Tang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Engineering-Electrical Engineering and Computer Sciences

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Costas J. Spanos, Chair

Professor Kameshwar Poola

Professor David Aldous

Spring 2011

Abstract

Uncertainty Propagation in Transistor-level Statistical Circuit Analysis

By

Qian Ying Tang

Doctor of Philosophy in
Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Costas J. Spanos, Chair

In today's semiconductor technology, the size of a transistor is made smaller and smaller. One of the key challenges presently faced by the designers is the increasing impact of process variations to circuit performances. As a result, circuits designed using the traditional methods can deviate from the desired specifications after being manufactured. Therefore, new circuit design and characterization methodologies are required to handle these process variations.

The problem of estimating the circuit performance at a transistor-level due to parameter uncertainties is examined. Uncertainty in circuit process and device parameters arises as a result of manufacturing variability. Since electrical circuits are, in general, complex and nonlinear systems, estimating their performances efficiently and accurately is very challenging. Existing methods on propagating uncertainties in circuit parameters to circuit performance include worst case corner analysis, Monte-Carlo simulations, response surface modeling, sensitivity analysis and unscented transformation.

In this work, a novel interval based circuit simulation algorithm is proposed. An interval is a quantity consists of noise variables following Gaussian. The algorithm is developed for both Gaussian and non-Gaussian process variations.

When the uncertainty in the circuit process and device parameters can be captured by correlated Gaussian distributions, the process/device parameters are first represented by the appropriate interval representations. An interval-valued SPICE simulator, in which all real number operations are replaced by interval operations, is used to simulate the circuit. The simulation results are therefore interval-values that can be used to extract performance statistics. In this approach, only one circuit simulation is required to obtain the best Gaussian distribution approximation for any circuit performance.

The algorithm is tested on RC circuits and transistor circuits with excellent simulation accuracy (<2% error) as compared to Monte Carlo simulation results. It is

shown analytically that the runtime of the interval valued circuit simulation is on the order of $O(n+m)O(c^3)$ where n is the average number of noise variables per interval operation, m is the average number of noise variables shared between any two interval quantities and c is the number of nodes in the circuit.

In the case when the process/device parameters cannot be modeled with Gaussian distributions, A Mixture of Gaussian (MOG) distribution is used to approximate all non-Gaussian distributions and a novel extension to the interval representation is proposed. The algorithm is tested on circuit paths of 100 stages containing inverters, NAND gates and NOR gates. The simulation result of the proposed algorithm agrees very well with Monte-Carlo simulation. In addition, the runtime of the proposed algorithm shows a 54X speed up compared to Monte-Carlo simulation.

The proposed interval-value based simulation engine for both Gaussian and non-Gaussian process variations can be directly applied to fast and accurate standard cell library characterization, where the distribution of the cell delay and power are calculated. In addition, the distribution information provided by the proposed simulation method can be feed into statistical timing analysis engine for full-chip level timing closure. The proposed simulation engine can also be incorporated into statistical circuit optimizations.

_____

Professor Costas J. Spanos
Dissertation Committee chair

Acknowledgements

First and foremost, I would like to express my sincerest gratitude to my research advisors, Professor Costas J. Spanos, for his excellent guidance on my research. His knowledge and experience in many areas have inspired many new ideas and broadened my vision. I would also like to thank him for his understanding and continuous support during the most difficult time during my study. What I have learned from him will be beneficial throughout my life.

I would like to thank Professor Kameshawar Poolla for the insightful discussions on uncertainty propagations and generalized adjoint networks. These discussions have helped tremendously in shaping the dissertation structure and contents. I would also like to thank him for serving on my qualifying exam committee and dissertation committee.

I would like to thank Professor David Aldous for his commitment and time in serving on my qualifying exam committee and dissertation committee. I would also like to thank Professor Chenming Hu for serving as the chair of my qualifying exam committee and Professor Alon Elad for serving on my qualifying exam committee.

I would like to thank Dr. Victor Moroz and Dr. Xiwen Lin from Synopsys for consultations and discussions.

Thanks go out to all my past colleagues and members of BCAM group for their support and discussions: Kedar Patel, Paul Friedberg, Jing Xue, Kun Qian, Yu Ben, Dekong Zeng.

This work is sponsored by AMD, Applied Materials, ASM Lithography, Canon, Ebara, Global Foundaries, IBM, Intel, KLA Tencor, Magma, Marvell, Mentor Graphics, Novellus, Panoramic Tech, Sandisk, Synopsys, Tokyo Electron, Xilinx, with matching support by the UC discovery Program.

Table of Contents

List of Figures

List of Tables

Chapter 1 Introduction

In today's semiconductor technology, the size of a transistor is made smaller and smaller. The benefits of transistor scaling have been to reduce manufacturing cost and to enhance device performance. However, as the devices are scaled down into the nanometer regime, issues such as manufacturability, process variations, voltage scaling, power dissipation and device reliability have greatly undermined the benefits of scaling. In today's semiconductor design and manufacturing, sophisticated methodologies have been used to ensure chip performance and manufacturability.

One of the key challenges presently faced by the designers is the increasing impact of process variations on circuit performances. For example, variations in the lithographic systems can cause variations in the printed line width, resulting in undesired shorts or opens in a chip layout, especially when the printed line width is narrow [1.1][1.2]. Other variability issues in nanoscale transistors include random dopant fluctuation [1.3], line edge roughness [1.4], well-proximity effects [1.5], layout dependent stress variation in strained silicon technologies [1.6], and rapid thermal anneal (RTA) temperature induced variation [1.7][1.8]. All of these variations cause circuits designed using the traditional design flow to deviate from the desired specifications after being manufactured.

In section 1.1, a brief overview on the types and modeling techniques of process variations observed in today's semiconductor manufacturing is given. Section 1.2 reviews the traditional worst-case corner based design flow. The dissertation organization is presented in section 1.3.

1.1 Process Variations in Semiconductor Manufacturing

In silicon manufacturing, many processing steps can cause a non-uniformity in the manufactured device properties, referred to as the process variations. Process variations have become a serious issue in nanoscale transistors. the following sections give an overview of the types of process variations, and current approaches to model such variations.

1.1.1 **Types of Variations**

Depending on their spatial scales, process variations can be categorized into inter-die (global) variations that affect all the devices on a die simultaneously, and intra-die (local) variations that affect each devices individually.

One example of inter-die variations are the loading effects in etching or deposition that impact the geometry of all the devices on a wafer [1.24]. Inter-die variations can be subdivided into lot-to-lot variations (i.e., variations between two lots of wafers), wafer-to-wafer variations (i.e., variations between two wafers within the same lot), and die-to-die variations (i.e., variation between two dies on a same wafer).

Examples of local variations include various types of layout dependent proximity effects [1.25]-[1.28]. The devices that are located closer to a well edge show higher threshold voltages than the devices located further away from the well edge, known as the well proximity effects [1.29]. In addition, the size and proximity of STI structures surrounding a device also impact device performance, due to the STI induced stresses in the device channel [1.27].

Depending on their nature, variations can also be categorized into *systematic* and *random*. Systematic refers to the variations that can be captured by a deterministic function, whereas random variations are stochastic in nature and can only by characterized by distributions. For example, during the manufacturing step known as rapid thermal annealing [1.32], the wafer is put into a furnace and the temperature is ramped up to the desired level. However, due to the non-uniform temperature distribution in the furnace, different locations of the wafer are processed at the different temperature, resulting into device property variations in the annealed devices. This variation is systematic since it can be captured by the known temperature distribution across the wafer. In another critical process step, when dopants are implanted into a material, the final concentrations injected into the material the variation is referred to as the random dopant fluctuation and it is random in nature. The following table summarizes the different types of variations.

Table 1.1: Categorization of different types of process variations.

	Systematic Variations	Random Variations
Inter-die Variations	across-wafer	Lot to Lot, Wafer to wafer Die to die
Intra-die Variations	across-die variations, Layout/context dependent Effects	Device to device LER/RDF

1.1.2 **Characterization and Modeling**

In order to characterize the amount of variations in a process, test structures consisting of ring oscillator and SRAM memory cells are often designed, manufactured and measured to obtain the statistics. Each device in the test structure are measured for

Ion and Ioff under various bias conditions to extract the I-V characteristic. Selected set of device parameter values are back extracted from this I-V characteristics. Device parameter extraction methods can be found in [1.9]-[1.11].

1.2 Variation-Aware Circuit Design

Worst case circuit analysis (WCCA) [1.19]-[1.23] has been used extensively in the industry for variation-aware circuit designs. In this approach, a set of worst case corners is generated by varying the physical and electrical parameters of representative devices (i.e., transistors, interconnect elements) [1.18] with the objective of achieving the worst case performance. Circuits are then simulated at these parameter corners to estimate the worst case performance, such as timing and power, in order to ensure that even under those worst case conditions the performance specifications are met.

The worst case value P^{wc} of a particular device performance P is defined as,

$$prob\,(P < P^{wc}) = \rho \tag{1.1}$$

where ρ is a pre-defined probability value typically close to 1 and approximates the required manufacturing yield. In theory, there can be a large number of different assignments of the device parameter values that give the same worst case performance. The particular set of worst case parameter values θ^{wc} assignment is chosen such that,

$$\begin{aligned} \theta^{wc} &= \max_{\theta} prob(\theta) \\ s.t. \quad & P(\theta) = P^{wc} \end{aligned} \tag{1.2}$$

or, in other words, the most probable values of device parameters that give rises to such worst case device performance. This is illustrated in Figure 1.1 for two device parameters, where the contours are from the joint distributions of the device parameters.

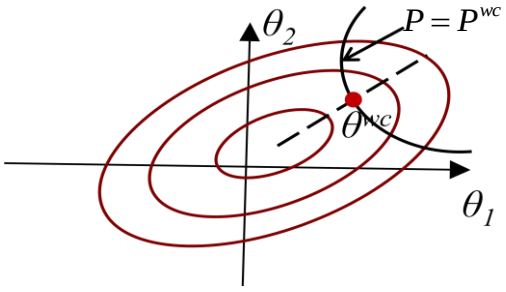

Figure 1.1: Illustration on determining the worst case corner for a device in the case of two process/device parameters.

Despite the popularity of using worst case corners for variability-ware circuit designs, the method becomes increasingly inadequate as the increase in both magnitude and complexity of process variations, coupled with the aggressive scaling of MOS

devices. Since the true performance for a circuit is design specific, while the worst case corner are determined only based on the worst case of device, using worst case corners for circuit design often leads to pessimistic designs. In addition, the worst case corners design methodology gives no information about the actual circuit yield, nor the actual probabilistic distribution of the performance of interest. All these problems with worst case design methodology motivate the need for incorporating full statistics in performance evaluation for aggressive designs.

Many authors [1.12]-[1.17] have demonstrated design methods that take into account of the entire distribution information of process/device parameters. For example, chip-level statistical timing analysis [1.30]-[1.31] (SSTA) has been implemented as a way of estimating the chip-level critical path delay by assuming a Gaussian distribution of the delay of each individual gate. The method propagates the gate delay distribution through the entire circuit path by defining Gaussian approximations to the MAX and SUM of any two Gaussian distributions. Extensions of SSTA to non-Gaussian distributions have also been well studied in past literatures.

Many of the existing methods build upon the availability of an accurate estimation of the distribution of gate delay or power. However, obtaining such distributions often involves hundreds or thousands runs of costly SPICE circuit simulations. An efficient and accurate SPICE-level statistical circuit performance analysis and simulation is still an active area of research.

In this dissertation, we proposed a novel interval-value based algorithm for speeding up SPICE-level statistical performance simulation for both Gaussian distributions and non-Gaussian distributions.

1.3 Dissertation Organization

The dissertation is organized as follows: chapter two gives a background overview of exiting methods of transistor/spice level statistical circuit simulation. Chapter three discusses our proposed interval-value based approach to efficient transistor level statistical circuit simulation for Gaussian uncertainties in parameter values. Chapter four extends this method to incorporating non-Gaussian parameter uncertainties. Chapter five concludes this dissertation.

References for Chapter 1

[1.1]. L.W. Liebmann *et. al.*, "TCAD Development for Lithography Resolution Enhancement", *IBM J. of Res. and Dev.*, vol. 45, pp. 651-665, Sep 2001.

[1.2]. L.W. Liebmann, "Resolution Enhancement Techniques in Optical Lithography, It's not Just a Mask Problem", *Proceedings of SPIE*, vol. 4409, pp. 23-32, 2001.

[1.3]. R.W. Keyes, "The Impact of Randomness in the Distribution of Impurity Atoms on FET Threshold", *Journal of Applied Physics*, 1975, pp. 251-259.

[1.4]. A. Asenov, *et. al.*, "Intrinsic Parameter Fluctuations in Decananometer MOSFETs Introduced by Gate Line Edge Roughness", *IEEE Transaction on Electron Devices.*, vol. 50, no. 5, pp. 1254-1260. May, 2003.

[1.5]. T.B. Hook *et al.*, 'Lateral Ion Implant Straggle and Mask Proximity Effect", *IEEE Trans on Elec. Devices*, vol. 50, pp. 1946-1951, Sep 2003.

[1.6]. V. Moroz, L. Smith, X.-W. Lin, D. Pramanik and G. Rollins, "Stress-Aware Design Methodology", *Intl. Symp. Quality Elec. Design*, 2006.

[1.7]. P.J. Timans *et al.*, "Challenges for Ultra-Shallow Junction Formation Technologies beyond the 90nm Node", *Intl. Conf. on Adv. Thermal Processing of Semiconductors*, pp. 17-33, Sep 2003.

[1.8]. I. Ahsan *et al.*, ''RTA-Driven Intra-Die Variations in Stage Delay, and Parametric Sensitivities for 65nm Technology,'' *Symp. on VLSI Technology*, pp. 170-171, 2006.

[1.9]. S. Natarajan, et. al., "Process variations and their impact on circuit operation", IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, pp 73-81., Nov. 1998.

[1.10]. K. Agarwal, S. Nassif, "Characterizing Process Variation in Nanometer CMOS", DAC 2007: 396-399.

[1.11]. H. Masuda, S. Ohkawa, A. Kurokawa, and M. Aoki. Challenge: Variability characterization and modeling for 65- to 90-nm processes. In *IEEE CICC*, pages 593–600, Sept. 2005.

[1.12]. S. Burns, et. al., "Comparative Analysis of Conventional and Statistical Design Techniques", Proceedings of the 44th annual Design Automation Conference (DAC), 2007.

[1.13]. A. Agarwal, K. Chopra, D. Blaauw, and V. Zolotov. Circuit optimization using statistical static timing analysis. In *DAC*, pages 321–324, 2005.

[1.14]. S. Borkar, T. Karnik, S. Narendra, J. Tschanz, A. Keshavarzi, and V. De. Parameter variations and impact on circuits and microarchitecture. In *DAC*, pages 338–342, June 2003.

[1.15]. K. A. Bowman, S. G. Duvall, and J. D. Meindl. Impact of die-to-die and within-die parameter fluctuations on the maximum clock frequency distribution for gigascale integration. *IEEE J. Solid-State Circuits*, pages 183–190, Nov. 2002.

[1.16]. S. G. Duvall. Statistical circuit modeling and optimization. In *5th Intl. Workshop on Statistical Metrology*, pages 56–63, June 2000.

[1.17]. F. Najm and N. Menezes. Statistical timing analysis based on a timing yield model. In *DAC*, pages 460–465, June 2004.

[1.18]. S. Nassif, A. Strojwas, and S. Director. A methodology for worst-case analysis of integrated circuits. *IEEE Trans. Computer-Aided Design*, pages 104–113, Jan.1986.

[1.19]. H. To, et. al., "Worst case analysis of low-voltage analog MOS integrated circuits", Proceedings of the 38th Midwest Symposium on Circuits and Systems,vol.1, pp. 278 - 281, Aug 1995.

[1.20]. S. Lee, et. al., "A realistic methodology for the worst case analysis of VLSI circuit performances", International Conference on Simulation of Semiconductor Processes and Devices. Pp. 155, Sept. 1996.

[1.21]. A. Dharchoudhury, S. Kang, "Worst-case analysis and optimization of VLSI circuit performances". IEEE Trans. on CAD of Integrated Circuits and Systems 14(4): 481-492 (1995)

[1.22]. A. Nardi, et. al., "Impact of Unrealistic Worst Case Modeling on the Performance of VLSI Circuits in Deep Sub-Micron CMOS Technologies", IEEE Transaction on Semiconductor Manufacturing, vol. 12, No. 4: 396-402. 1999.

[1.23]. A. N. Lokanathan, J. B. Brockman, "Efficient worst case analysis of integrated circuits", Proceedings of the IEEE Custom Integrated Circuits Conference, Pp. 237 - 240, 1995.

[1.24]. Y. Cao, L.T. Clark, "Mapping Statistical Process Variations Toward Circuit Performance Variability: An Analytical Modeling Approach", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol 26, No.10, pp 1866 - 1873. Oct. 2007.

[1.25]. Xi-Wei Lin, Victor Moroz, "Layout Proximity Effects and Modeling Alternatives for IC Designs," IEEE Design and Test of Computers, vol. 27, no. 2, pp. 18-25, Mar./Apr. 2010.

[1.26]. H. Tsuno et al., "Advanced Analysis and Modeling of MOSFET Characteristic Fluctuation Caused by Layout Variation," *Proc. Symp. VLSI Technology*, IEEE Press, 2007, pp. 204-205.

[1.27]. V. Moroz et al., "Stress-Aware Design Methodology," *Proc. 7th Int'l Symp. Quality Electronic Design* (ISQED 06), IEEE CS Press, 2006, pp. 807-812.

[1.28]. E. Morifuji et al., "Layout Dependence Modeling for 45-nm CMOS with Stress-Enhanced Technique," *IEEE Trans. Electron Devices*, vol. 56, no. 9, 2009, pp. 1991-1998.

[1.29]. J. Watts et al., "Netlisting and Modeling Well-Proximity Effects," *IEEE Trans. Electron Devices*, vol. 53, no. 9, 2006, pp. 2179-2186.

[1.30]. Orshansky, M.; Keutzer, K., 2002, A general probabilistic framework for worst case timing analysis, Design Automation Conference, 2002. Proceedings. 39th, Vol., Iss., 2002, Pages: 556–561.

[1.31]. Visweswariah, C.; Ravindran, K.; Kalafala, K.; Walker, S.G.; Narayan, S.; Beece, D.K.; Piaget, J.; Venkateswaran, N.; Hemmett, J.G., 2006, First-Order Incremental Block-Based Statistical Timing Analysis, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, , Vol.25, Iss.10, Oct. 2006, Pages: 2170–2180.

[1.32]. A. Kamgar, "Rapid Thermal Processing of Silicon," in *Submicron Integrated Circuits*, edited by R. K. Watts (Wiley Interscience, New York, 1989).

Chapter 2 Uncertainty Propagation in Circuit Analysis

2.1 Introduction

In this chapter, existing uncertainty characterization and propagation method are reviewed and their applications to transistor-level statistical circuit analysis are discussed.

2.2 Sampling Based Method

The most straightforward method for evaluating statistical circuit performance is by means of sampling. In such approach, samples are drawn from the distributions of the circuit parameters, and the circuit is simulated in a transistor level simulator, e.g., HSPICE, using these sample values. Statistics on performance distributions can then be found by looking at these simulation results.

If samples are drawn randomly in the parameter space, the method is referred to as Monte-Carlo sampling method. The cost of the Monte-Carlo sampling method for statistical circuit performance evaluation is the product of the number of samples and the runtime required for a single circuit simulation, plus any additional cost required to estimate the distributions of the performance from samples. In transistor-level circuit simulators such as HSPICE, the cost of one simulation is on the order of $O(n^3)$, where n is the size of the circuit [2.1]. Therefore, the cost can be very large when the number of samples required is large.

The benefits of the Monte-Carlo approach is that the number of samples required to obtain the distributions for a particular circuit performances is independent of the number of parameters. However, the estimation accuracy depends highly (and generally the square) on the number of samples used. In the discussion by [2.2], the authors showed that 10x reduction in estimation error requires a 100x increase in the number of samples when estimating a Gaussian distribution. This problem becomes very acute when estimating low probability events, such as yield loss..

The Latin Hypercube sampling method [2.3][2.4] tries to reduce the number of samples required in a Monte-Carlo estimation by placing more control on how the

samples are drawn from the parameter space. The goal is to make sure that the samples are drawn uniformly from all regions of the parameter space, and to avoid repeated samples from the regions where the parameter probability density function have a higher value (such situation occurs in Monte-Carlo sampling method).

In Latin hypercube sampling, the cumulative distribution functions of the parameters are stratified into equal segments, and one or more samples per segment are used for performance evaluation. Figure 2.1 illustrates how four samples are generated using Latin Hypercube sampling schemes in a 2-dimensional case. By ensuring that there is only one single grid filled with one sample for each row or column, the samples can be more uniformly located within the sample space. The sample inside each grid is drawn randomly.

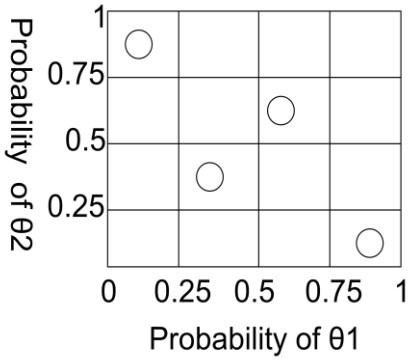

Figure 2.1: Illustration of the Latin Hyper cube sampling method on two independent random parameters with four samples.

Authors in [2.3] show that for the same number of samples, the variance of the Latin hypercube estimator is smaller than that of the Monte-Carlo estimator, when "certain monotonicity condition holds".

Another commonly used sampling method is importance sampling [2.5][2.6]. This approach is particularly advantageous in estimating the tail of a distribution. In importance sampling, the probability distribution function of the parameters are first distorted such that when the samples are drawn from the distorted, the variance of the estimation is reduced. For example, in the case of SRAM failure probability estimation, importance sample will shift the original distribution such that the events in the tail of the distribution (i.e., the events that an SRAM cell fails) now have a much higher probability to occur. In doing so, with the same number of samples, a more accurate estimation can be obtained compared to sampling from the original distribution. However, the disadvantage in using importance sampling is that there is not always an easy method to find the distorted distribution while providing the required estimator variance reduction [2.2].

Despite the possibility of the reducing the number of samples, the runtime of sampling based methods are generally substantial compared to other methods that will be discussed in the following sections. The cost of sampling based methods for statistical circuit simulation is equal to the number of samples multiplied by the cost of simulating a

circuit. The cost of circuit simulation itself can be substantial, thus limits the use of sampling based methods to only very small sized circuits.

2.3 Response Surface Modeling

Response surface modeling [2.7][2.10][2.11] tries to reduce the cost of statistical circuit simulation by replacing the costly, SPICE circuit simulations by an simple empirical model that can be evaluated at a much lower cost.

In this approach, a design of experiment (DOE) is first performed on the parameters to generate representative data points from the parameter space. General background on DOE can be found in [2.9]. One commonly used DOE scheme is the central composite design as illustrated in Figure 2.2 for a 2-dimensional case. A center point and four factorial points located ±1 unit away from the center (as shown by the blue dots in Figure 2.2). In addition, four points located at $\pm\alpha$ with $|\alpha| > 1$ away from 0 and on the parameter axis are chosen, as shown by the diamond shaped points in Figure 2.2. The precise value of α depends on the desired properties for the DOE and on the number of factors involved [2.12].

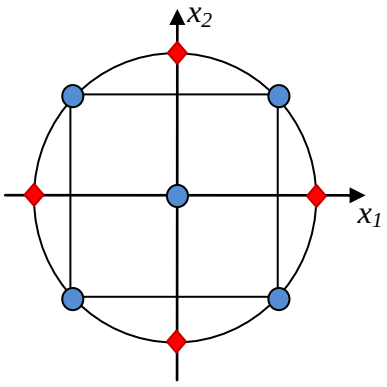

Figure 2.2: Illustration of the central composite design for two variables. The diamond and circle shape marks the samples points resulted from this DOE.

After data points are selected using a DOE, the circuit is simulated under these data points to obtain the performance values. A response surface model is fitted to relate the circuit performance to parameters. Commonly used response surface models are either first order or second order. The first order model, or the linear response surface model is given by,

$$f(\tilde{\boldsymbol{x}}) = \tilde{b}^T\tilde{\boldsymbol{x}} + c \tag{2.1}$$

and the second order model is given by,

$$f(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}}^T A\tilde{\boldsymbol{x}} + b^T\tilde{\boldsymbol{x}} + c$$

(2.2)

where $f(\tilde{\boldsymbol{x}})$ denotes the value of the performance, or the response of the model, $\tilde{\boldsymbol{x}}$ is the set of all device/process parameters, $\tilde{b}$ is a *n*-dimensional vector of coefficients, A is a n by n matrix, and c is a constant bias value. The linear model is suitable when the variation in $\tilde{\boldsymbol{x}}$ is small, while the second order model is used for the case when the variations is large. The value of the model parameters $\tilde{b}$, A and c is obtained by solving the set of over-determined equations,

$$f_i = \tilde{b}^T \tilde{x}_i + c \qquad\qquad i = 1, \dots, s \tag{2.3}$$

for the linear model, and

$$f_i = \tilde{x}_i{}^T A \tilde{x}_i + b^T \tilde{x}_i + c \quad i = 1, \dots, s \tag{2.4}$$

for the second order model. The terms $\tilde{x}_i$ and f_i are the DOE samples from the parameter space and the corresponding circuit responses, for a total of s number of samples.

One of the commonly used methods for solving systems of over determined equations is the least-square approximation [2.8]. For example, in the case of linear response surface models, to estimate the value of parameters, a matrix that collects all the DOE samples is first formed as

$$X = \begin{bmatrix} 1 & \tilde{x}_1^T \\ 1 & \tilde{x}_2^T \\ \vdots & \vdots \\ 1 & \tilde{x}_n^T \end{bmatrix} \tag{2.5}$$

where a 1.0 is appended to all DOE samples $\tilde{x}_1, \dots, \tilde{x}_n$ in oprder to accommodate the costant "c" shown in equations 2.3 and 2.4. Let the vector $\tilde{y} = [f_1, \dots, f_n]^T$ collect all the circuit responses, then the least square approximation of the model parameters for the linear model is given by

$$\begin{bmatrix} \boldsymbol{c} \\ \tilde{\boldsymbol{b}} \end{bmatrix} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T} \tilde{\boldsymbol{y}} \tag{2.6}$$

for c and $\tilde{b}$ in equation (2.3).

Once a response surface model has been built, the distribution of the circuit performance can be obtained by repeatedly evaluate the model using a Monte-Carlo method. Since the cost of model evaluation is very small, a much larger number of

Monte-Carlo samples can now be used for obtaining an estimate of the performance distribution.

The total cost for using response surface modeling for statistical circuit performance evaluation consists of three parts, a) the cost for generating DOE samples and simulations of the circuit at those DOE points; b) the cost for building a response surface model, and c) the cost for generating performance distributions by evaluating the response surface models. In general the cost of part (a) and part (b) dominates over the cost for part (c).

The major benefit of response surface modeling over direct sampling approaches is the reduction in runtime when the dimension of the parameter space is small. The number of DOE experiments required to arrive at response surface model is generally smaller than the number of samples required in a sampling approach, as long as the dimension is confined within a value. However, the number DOE experiments grows exponentially with the number of dimensions, and thus can be much more costly than a sampling based method if the parameter dimension is large.

2.4 Unscented Transformation

The unscented transformation (UT) [2.14]-[2.18] is an alternative method for estimating a probability distribution function. Given a vector-valued random variable $\tilde{\boldsymbol{x}}$, and an differentiable function *f*, such that

$$y = f(\tilde{\boldsymbol{x}}) \tag{2.7}$$

The unscented transformation tries to estimate the statistics, i.e., mean and variance, of *y*, from the function *f* and the distribution of $\tilde{\boldsymbol{x}}$. In the context of statistical circuit simulation, $\tilde{\boldsymbol{x}}$ is the vector of device/process parameters subject to variations, *y* is the circuit performance of interest, and the function *f* is implicitly available through SPICE circuit simulations.

The basic idea behind UT is to obtain a set of *s* "sigma" points, i.e., points that are a certain number of standard deviations away from the mean, from the distribution of $\tilde{\boldsymbol{x}}$. These "sigma" points are denoted as x_i's. The function *f* is then evaluated at those sigma points to obtain a corresponding set of values $y_i = f(x_i), i = 1, \ldots, n$. The mean and variance of *y* is then estimated from those points through weighted sums, i.e.

$$\bar{y}^* = \sum_{i=1}^{s} w_i y_i \tag{2.8}$$

and

$$p^* = \sum_{i=1}^{s} w_i (y_i - \bar{y}^*)(y_i - \bar{y}^*)^T \tag{2.9}$$

where $\bar{y}^*$ and p^* is the estimation of the mean and variance of y using the unscented transformation; s is the number of sigma points; $\bar{x}$ is the mean vector of $\tilde{\boldsymbol{x}}$; and w_i are the weights, whose calculation is discussed next.

The concept of unscented transformation is illustrated in Figure 2.3 and in comparison with the Monte-Carlo method,

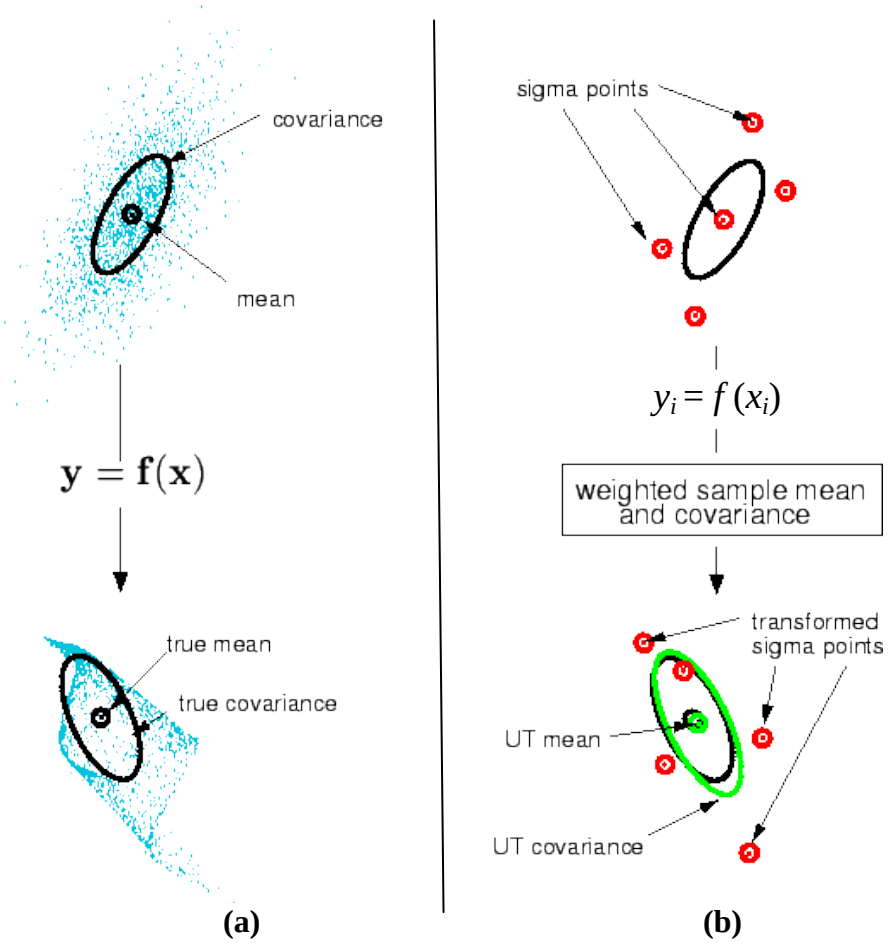

Figure 2.3: Example of the unscented transformation for mean and covariance propagation. a) Monte-Carlo b) Unscented Transformation (UT). (Reproduced from [2.12]).

A Taylor series expansion of the function f around the mean vector $\bar{x}$, is performed in order to derive the values for the weights and the sigma points [2.14]

$$y = f(\tilde{\boldsymbol{x}}) = f(\bar{x}) + \nabla_f^T(\bar{x})(\tilde{\boldsymbol{x}} - \bar{x}) + \frac{1}{2}(\tilde{\boldsymbol{x}} - \bar{x})^T H_f(\bar{x})(\tilde{\boldsymbol{x}} - \bar{x}) + o(|\tilde{\boldsymbol{x}} - \bar{x}|^2) \tag{2.10}$$

where $\nabla_f^T(\bar{x})$ and $H_f(\bar{x})$ are the gradient and the Hessian matrix of the function f evaluated at $\bar{x}$. Taking the mean of equation (2.10),

$$\bar{y} = f(\bar{x}) + \frac{1}{2}\mathrm{E}\left((\tilde{\boldsymbol{x}} - \bar{x})^T H_f(\bar{x})(\tilde{\boldsymbol{x}} - \bar{x})\right) \tag{2.11}$$

Equation (2.8) can be expanded by substituting the second order Taylor series approximation for each y_i to become

$$\bar{y}^* = \sum_{i=1}^{s} w_i y_i = \sum_{i=1}^{s} w_i \left\{ f(\bar{x}) + \nabla_f^T(\bar{x})(x_i - \bar{x}) + \frac{1}{2}(x_i - \bar{x})^T H_f(\bar{x})(x_i - \bar{x}) + o(|x_i - \bar{x}|^2) \right\} \tag{2.12}$$

where s is the total number of sigma points. Comparing equation (2.12) with equation (2.11), the values of x_i and the weights w_i and w_0 have to satisfy the following conditions,

$$\sum_i w_i = 1 \tag{2.13}$$

$$\sum_i w_i(x_i - \bar{x}) = 0 \tag{2.14}$$

$$\mathrm{E}\left((\tilde{\boldsymbol{x}} - \bar{x})^T(\tilde{\boldsymbol{x}} - \bar{x})\right) = \sum_i w_i(x_i - \bar{x})^T(x_i - \bar{x}) \tag{2.15}$$

In addition, equation (2.13) - (2.15) are the only conditions required for the unscented transformation estimation for the mean estimation to be unbiased up to the second order. These equations are referred to as the unscented conditions for mean. Assuming the covariance matrix of $\tilde{\boldsymbol{x}}$ can be decomposed into $\sigma\sigma^T$ with $\sigma \in R^{n\times n}$, n is the dimension of $\tilde{\boldsymbol{x}}$, and with the introduction of any scaling parameter, $\alpha \in R$, the value of the weights w_i and the sigma points x_i are

$$\begin{aligned} w_{\pm i} &= \frac{1}{2\alpha^2}, \quad i = 1, \dots, n \\ w_0 &= 1 - \frac{n}{\alpha^2} \\ x_{\pm i} &= \bar{x} \pm \alpha\sigma_i, i = 1, \dots, n \\ x_0 &= \bar{x} \end{aligned} \tag{2.16}$$

where σ_i is the i^{th} column of the matrix.

A similar derivation is possible for the unscented transformation estimation of the variance. It can be shown [2.14] that the value in equation (2.16) also gives a second order estimation for the variance.

In the context of the statistical circuit simulation, the sigma points x_i are extracted from the distribution of the process parameter by using equation (2.16). y_i is obtained by simulating the circuit at each of the sigma points. The mean and variance of the circuit performance *y* are then obtained using equations (2.8) and (2.9). The cost of using unscented transformation for statistical circuit performance evaluation equals to the number of the sigma points multiplied by the cost of one circuit simulation, plus the cost of obtaining the matrix *S* from the covariance matrix of $\tilde{\boldsymbol{x}}$. The number of sigma points, if using the scheme in equation (2.16), is 2*n*+1, where *n* is the number of random process parameters. The cost of obtaining matrix *S* is given by the runtime required for an eigen-decomposition on the covariance matrix of $\tilde{\boldsymbol{x}}$. The runtime of such decomposition is on the order of n^3, where *n* is the number of process parameters

2.5 Sensitivity Analysis

In circuit sensitivity analysis, one tries to find out the changes in performance value Δp due to a small perturbation in the device/process parameters $\Delta\theta$. One example of sensitivity analysis is the study of the changes in the delay of logic gate due to a small change in the device threshold voltage. Sensitivities of a performance *p* with respect to a single parameter can be obtained by simulating the circuit at parameter values that are $\pm\Delta\theta$ away from the nominal condition θ_0, and by using the finite differences formula to calculate the sensitivity s_θ as

$$s_\theta = \frac{p(\theta + \Delta\theta) - p(\theta - \Delta\theta)}{2\Delta\theta} \tag{2.17}$$

If the sensitivities of the performance with respect to all parameters are given, then the variances of circuit performance $\sigma^2(p)$ due to random process/device parameters can be estimated from the sensitivities, i.e.,

$$\sigma^2(p) = s_{\theta_1}\sigma^2(\theta_1) + \ldots + s_{\theta_n}\sigma^2(\theta_n) \tag{2.18}$$

where the performance p is a function of the parameters, $\theta_1, \ldots, \theta_n$. The terms $s_{\theta_1}, \ldots, s_{\theta_n}$ are the sensitivities of the performance p with respect to the parameters, and $\sigma^2(\theta_1), \ldots, \sigma^2(\theta_n)$ is the variance in each individual parameters.

The variance estimation as represented in equation (2.18) is only accurate when the performance is a linear function of the parameters, or when the performance function can be approximated as a linear function of the parameters in the range of parameter

variations. In the context of circuit simulation, the sensitivity analysis is used only when the process variations are small.

Direct sensitivity analysis by means of perturbing the parameters requires a large number of circuit simulations, especially when the parameter space is large. The Generalized Adjoint Network approach [2.19]-[2.21] has been investigated as a method of efficiently calculating the sensitivities, and the runtime is relatively independent of the dimension of the parameter space.

In the generalized adjoint network approach, each circuit element is replaced with its corresponding adjoint counterpart to produce a new circuit with the same circuit topology. The new adjoint circuit is then simulated, but with a reversal in time as compared to the original circuit, in order to obtain a set of branch currents and nodal voltages. The voltages and currents from the original circuit simulation, and those obtained from the adjoint circuit simulations are used together to calculate the sensitivities of a particular circuit response to the parameters of the circuit elements which has been replaced by their adjoints. Therefore, in the adjoint network approach, only two SPICE circuit simulations are required per circuit response of interest, regardless of the number of parameters.

The adjoint network for sensitivity calculations make use of the Tellegen's Theorem [2.23], which states that for any network with arbitrary multi-terminal or two-terminal elements, as long as each element possesses a parametric representation, then there exists an adjoint network with the identical topology, that satisfies the following relationships,

$$\sum_B v_B(t)\phi_B(\tau) = 0$$

$$\sum_B i_B(t)\psi_B(\tau) = 0 \tag{2.19}$$

where $v_B(t)$ and $i_B(t)$ are the branch voltages and currents of the original network; $\psi_B(\tau)$ and $\phi_B(\tau)$ are the branch currents and voltages of the adjoint network; and the summation is taken over all branches in the network.

In the presence of perturbations of the device parameters, the branch currents and voltages deviate from the nominal values, and the deviation is denoted as $\Delta i_B(t)$ and $\Delta v_B(t)$. Applying Tellegen's theorem to the perturbed network gives,

$$\sum_B \big(v_B(t) + \Delta v_B(t)\big)\phi_B(\tau) = 0$$

$$\sum_B \big(i_B(t) + \Delta i_B(t)\big)\psi_B(\tau) = 0 \tag{2.20}$$

By comparing equation (2.19) to equation (2.20), the following is obtained,

$$\sum_B \Delta v_B(t)\phi_B(\tau) = 0$$
$$\sum_B \Delta i_B(t)\psi_B(\tau) = 0$$

(2.21)

and therefore,

$$\sum_B [\Delta v_B(t)\phi_B(\tau) - \Delta i_B(t)\psi_B(\tau)] = 0$$

(2.22)

Equation (2.22) is the key governing equation that is used to decide on the adjoint counterpart of each circuit element. For a resistive branch described by the set of branch voltage and current equations

$$v_R(t) = f_R(x_R(t), p_R, t)$$
$$i_R(t) = g_R(x_R(t), p_R, t)$$

(2.23)

where $x_R(t)$ is a representative performance value, and p_R is the set of device/process parameters subjected to variations. For example, in the case of a time-invariant linear resistive branch, $x_R(t) = i_R(t)$, $f_R = R(p_R)x_R(t)$. Applying a first order perturbation on this branch gives,

$$\Delta v_R(t) = \frac{\partial f}{\partial x_R}\Delta x_R(t) + \frac{\partial f}{\partial p_R}\Delta p_R$$
$$\Delta i_R(t) = \frac{\partial g}{\partial x_R}\Delta x_R(t) + \frac{\partial g}{\partial p_R}\Delta p_R$$

(2.24)

Substitute equation (2.24) into equation (2.22), the following expression is obtained for the term associated with the resistive branch,

$$\left[\frac{\partial f}{\partial x_R}\phi_R(\tau) - \frac{\partial g}{\partial x_R}\psi_R(\tau)\right]\Delta x_R(t) + \left[\frac{\partial f}{\partial p_R}\phi_R(\tau) - \frac{\partial g}{\partial p_R}\psi_R(\tau)\right]\Delta p_R$$

(2.25)

Since one is only interested in the sensitivities, or variations of the circuit performance with respect to the process/device parameters (i.e., Δp_R), but not to the

branch currents/voltages (i.e., Δx_R), the following "adjoint network requirement" is imposed on equation (2.25),

$$\left[\frac{\partial f}{\partial x_R}\phi_R(\tau) - \frac{\partial g}{\partial x_R}\psi_R(\tau)\right] = 0 \tag{2.26}$$

Equation (2.26) describes how the branch current and the voltages should be related in the adjoint network that corresponds to a resistive branch in the original network. In the case of a simple, time-invariant linear resistance, equation (2.26) simplifies to $\psi_R(\tau) = R(p_R)\phi_R(\tau)$. In other words, the adjoint counterpart of a linear, time invariant resistor is still a resistor with the same resistance value.

For a capacitive element, where the branch equations describing this element is given by

$$\begin{aligned} v_C(t) &= f_C(x_C(t), p_C, t) \\ q_C(t) &= g_C(x_C(t), p_c, t) \\ i_C(t) &= \frac{d}{dt}q_C(t) \end{aligned} \tag{2.27}$$

where $q_C(t)$ is the charges stored in the capacitor. Again a perturbation introduced on the capacitive branch can be represented by,

$$\begin{aligned} \Delta v_C(t) &= \frac{\partial f_C}{\partial x_C}\Delta x_C(t) + \frac{\partial f_C}{\partial p_C}\Delta p_C \\ \Delta q_C(t) &= \frac{\partial g_C}{\partial x_C}\Delta x_C(t) + \frac{\partial g_C}{\partial p_C}\Delta p_C \\ \Delta i_C(t) &= \frac{d}{dt}\Delta q_C(t) \end{aligned} \tag{2.28}$$

Similarly, substituting equation (2.28) into equation (2.22) gives the following expressions for the terms associated with the capacitive branches,

$$\left[\frac{\partial f}{\partial x_C}\phi_C(\tau)\Delta x_C(t) - \frac{d}{dt}\left(\frac{\partial g}{\partial x_C}\Delta x_C(t)\right)\psi_C(\tau)\right] + \left[\frac{\partial f}{\partial p_C}\phi_C(\tau) - \frac{d}{dt}\left(\frac{\partial g}{\partial p_C}\right)\psi_C(\tau)\right]\Delta p_C \tag{2.29}$$

Imposing the "adjoint network requirement" as in the case of a resistive branch gives the following adjoint branch relationship

$$\frac{\partial f}{\partial x_C}\phi_C(\tau)\Delta x_C(t) - \frac{d}{dt}\left(\frac{\partial g}{\partial x_C}\Delta x_C(t)\right)\psi_C(\tau) = 0$$

(2.30)

However, equation (2.30) as given in this form cannot be explicitly converted to a circuit element. If the circuit is simulated for the time interval $t \in (t_0, t_f)$, and by letting $\tau = t_0 + t_f - t$, an integration equation (2.30) with respect to the time variable *t* from t_0 to t_f yields,

$$\int_{t_0}^{t_f} \left[\frac{\partial f}{\partial x_C}\phi_C(t_0 + t_f - t) + \frac{\partial g}{\partial x_C}\frac{d}{dt}\psi_C(t_0 + t_f - t)\right]\Delta x_C(t)dt - \frac{\partial g}{\partial x_C}\Delta x_C(t)\psi_C(t_0 + t_f - t)\Big|_{t_0}^{t_f} = 0 \tag{2.31}$$

where the expression $\frac{\partial g}{\partial x_C}\Delta x_C(t)\psi_C(t_0 + t_f - t)\Big|_{t_0}^{t_f}$ can be set to zero if $\psi_C(\tau = 0)$ and $\Delta x_C(t_0)$ are both zero. The condition $\Delta x_C(t_0) = 0$ can be satisfied if the initial branch current/voltages of the original capacitive branch are not subjected to any perturbations due to the perturbations in the process/device parameters. The condition $\psi_C(\tau = 0) = 0$ can be satisfied by setting the initial branch voltage in the adjoint network to zero. Equation (2.31) now reduces to

$$\int_{t_0}^{t_f} \left[\frac{\partial f}{\partial x_C}\phi_C(t_0 + t_f - t) + \frac{\partial g}{\partial x_C}\frac{d}{dt}\psi_C(t_0 + t_f - t)\right]\Delta x_C(t)dt = 0$$

Or, for each time step,

$$\frac{\partial f}{\partial x_C}\phi_C(\tau) + \frac{\partial g}{\partial x_C}\frac{d}{d\tau}\psi_C(\tau) = 0 \tag{2.32}$$

Equation (2.32) gives the branch relationship in the adjoint network corresponding to a capacitive branch in the original network. In the case of a linear, time invariant capacitor in the original network, the corresponding element in the adjoint network is again a linear time invariant capacitor with the same capacitance value, but with a properly set initial condition.

The observation that *τ*, the time variable for the adjoint network is set to $t_0 + t_f - t$ when obtaining equation (2.32) suggests that the time step in the adjoint network is in reversal with the time step of the original network. Table 2.1 summarizes the adjoint representation of some commonly used circuit element types.

Table 2.1: Summary of the adjoint equivalent of commonly used circuit elements. (Reproduced from [2.21]).

Element Type	Circuit Equation in Original Network	Circuit Equation in Adjoint Network	Sensitivity (s_B)
Open Circuit	$v_v(t) = V$ $i_v(t) = 0$	$\psi_v(t) = 0$ $\phi_v(t) = V$	
Short Circuit	$v_i(t) = 0$ $i_i(t) = I$	$\psi_i(t) = I$ $\phi_i(t) = 0$	
Resistive	$v_R(t) = f_R(x_R(t), p_R, t)$ $i_R(t) = g_R(x_R(t), p_R, t)$	$\frac{\partial f_R}{\partial x_R}\phi_R(\tau) = \frac{\partial g_R}{\partial x_R}\psi_R(\tau)$	$\frac{\partial f_R}{\partial p_R}\phi_R(\tau) - \frac{\partial g_R}{\partial p_R}\psi_R(\tau)$
Capacitive	$v_C(t) = f_C(x_C(t), p_C, t)$ $q_C(t) = g_C(x_C(t), p_c, t)$ $i_C(t) = \frac{d}{dt}q_C(t)$	$\frac{\partial f_C}{\partial x_C}\phi_C(\tau)$ $= \frac{\partial g_C}{\partial x_C}\frac{d}{d\tau}\psi_C(\tau)$	$\frac{\partial f_C}{\partial p_C}\phi_C(\tau)$ $-\frac{d}{dt}\left(\frac{\partial g_C}{\partial p_C}\right)\psi_C(\tau)$
Inductive	$\theta_L(t) = f_L(x_L(t), p_L, t)$ $i_L(t) = g_L(x_L(t), p_L, t)$ $v_L(t) = \frac{d}{dt}\theta_L(t)$	$\frac{\partial f_L}{\partial x_L}\frac{d}{d\tau}\phi_L(\tau)$ $= \frac{\partial g_L}{\partial x_L}\psi_L(\tau)$	$\frac{d}{dt}\frac{\partial f_L}{\partial p_L}\phi_L(\tau)$ $-\frac{\partial g_C}{\partial p_C}\psi_C(\tau)$

Once the adjoint network is constructed from the original circuit network, the sensitivities can be obtained by substituting the adjoint branch relationships into equation (2.22) to obtain

$$\begin{aligned}\sum_{k\in Input\ Port} &[\Delta v_k(t)\phi_k(\tau) - \Delta i_k(t)\psi_k(\tau)] \\ &= -\sum_R \left[\frac{\partial f_R}{\partial p_R}\phi_R(\tau) - \frac{\partial g_R}{\partial p_R}\psi_R(\tau)\right]\Delta p_R \\ &\quad -\sum_C \left[\frac{\partial f_C}{\partial p_R}\phi_C(\tau) - \frac{d}{dt}\frac{\partial g_C}{\partial p_C}\psi_C(\tau)\right]\Delta p_C - \cdots \\ &= -\sum_B s_B \Delta p_B\end{aligned} \tag{2.33}$$

where the left hand side of the equation is the summation over all excitations applied to the original and adjoint networks. The right hand side of the summation sums over all the branches in the original and adjoint networks. The term s_B refers to the sensitivity of each branch and is tabulated in the third column of Table 2.1.

To illustrate how the adjoint network approach can be used to find the sensitivities of arbitrary circuit response functions to circuit parameters, consider the simplest case where the performance of interest is the same as the circuit reaction (i.e., voltage across a current source or current through a voltage source) at the k^{th} port of the network. The

sensitivity of the response can be found by setting the excitation to all ports of the adjoint network to zero, except for the k^{th} port on which a unit excitation is applied. For example, as illustrated in Figure 2.4, the performance of interest is the voltage across the current source of k^{th} port in the original network, i.e., v_k. Then in the adjoint network, the k^{th} port is connected to a unit current source such that the adjoint current ϕ_k =1, whereas all other ports are either short-circuited (if the original port is connected to a voltage source) or open-circuited (if the original port is connected to a current source).

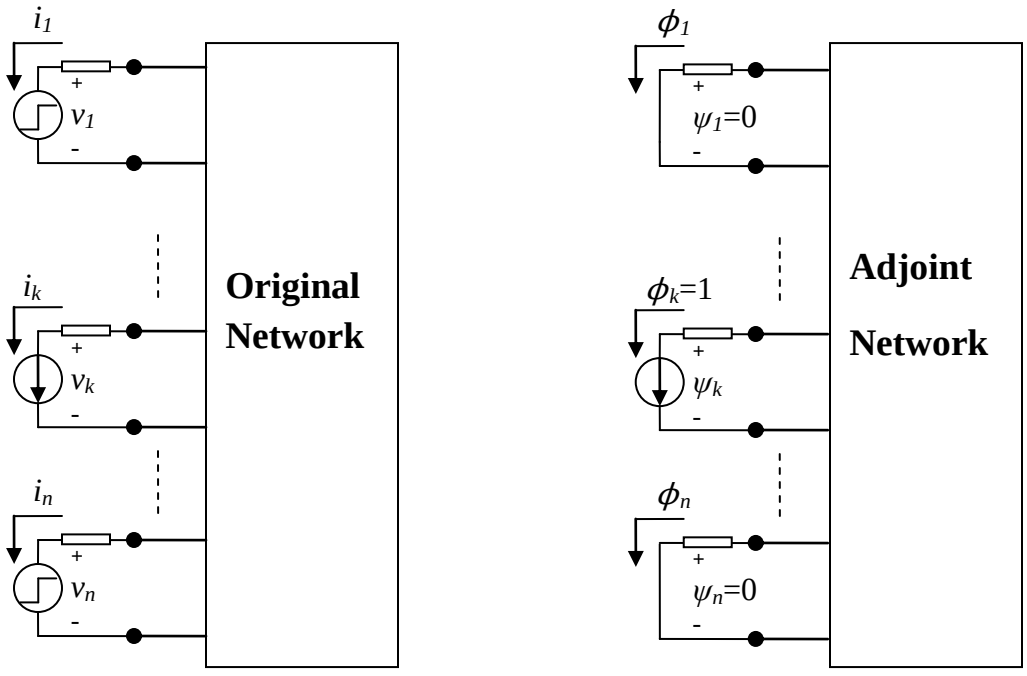

Figure 2.4: Illustration of using adjoint network for calculating sensitivities of a circuit performance.

By applying equation (**2.33**), the perturbation in the performance function is found to be,

$$\Delta v_k = -\sum_B s_B \Delta p_B \tag{2.34}$$

The sensitivity of the performance function v_k to each parameter can therefore be read off from the sensitivity coefficients s_B.

In a more general case, the performance function of interest maybe a function of the port voltage-current responses, and may involve more than one ports. The first order sensitivity of the performance *z* can be expressed as,

$$\Delta z = \sum_{\substack{k \in Voltage \\ Source}} \frac{\partial z}{\partial i_k} \Delta i_k + \sum_{\substack{k \in current \\ Source}} \frac{\partial z}{\partial v_k} \Delta v_k \tag{2.35}$$

where the first summation is over all ports that are connected to a voltage source in the original network, while the second summation is over all ports that are connected to a current source. Comparing equation (2.35) to the left hand side of equation (2.33), and by setting the port excitation of the adjoint network as

$$\phi_k = \frac{\partial z}{\partial v_k}$$

for all the ports that have an current source in the original network, and

$$\psi_k = -\frac{\partial z}{\partial i_k}$$

for all the ports that have an voltage source connected to in the original network.

With this excitation assigned to the adjoint network, equation (2.35) becomes

$$\Delta z = \sum_{\substack{k \in Voltage \\ Source}} \frac{\partial z}{\partial i_k} \Delta i_k + \sum_{\substack{k \in current \\ Source}} \frac{\partial z}{\partial v_k} \Delta v_k = \sum_{k \in input\ ports} \Delta v_k \phi_k - \Delta i_k \psi_k =$$
$$= -\sum_B s_B \Delta p_B$$

Thus, the sensitivity of the performances with respect to each parameter can be found by reading out the coefficients.

The sensitivity analysis is an efficient and easy to implement method for estimating the statistical variations in circuit performances given the parameters. The adjoint network approach gives a nearly constant runtime which does not increase as the number of parameter dimension increases. However, the linearly assumption in sensitivity analyses restricts its use to only small process/device variations.

References for Chapter 2

[2.1]. Nagel, Laurence W.,"SPICE2: A Computer Program to Simulate Semiconductor Circuits", EECS Department, University of California, Berkeley 1975. Available online: http://www.eecs.berkeley.edu/Pubs/TechRpts/1975/9602.html

[2.2]. Xin Li, Jiayong Le, Lawrence T. Pileggi, "Statistical Performance Modeling and Optimization", Journal Foundations and Trends in Electronic Design Automation, Volume 1 Issue 4, April 2006.

[2.3]. McKay, M.D., Beckman, R.J., Conover, W.J. (May 1979). "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code". Technometrics (American Statistical Association) 21 (2): 239–245.

[2.4]. Iman, R.L., Helton, J.C., and Campbell, J.E. (1981). "An approach to sensitivity analysis of computer models, Part 1. Introduction, input variable selection and preliminary variable assessment". Journal of Quality Technology 13 (3): 174–183.

[2.5]. R. Srinivasan, *Importance sampling - Applications in communications and detection*, Springer-Verlag, Berlin, 2002.

[2.6]. James Antonio Bucklew, *Introduction to rare event simulation*, Springer-Verlag, New York, 2004.

[2.7]. Xin Li, Jiayong Le, L.T. Pileggi, A. Strojwas, "Projection-based performance modeling for inter/intra-die variations", Proceeding of ICCAD, 2005.

[2.8]. R. H. Myers. *Response surface methodology: process and product optimization using designed experiments*, Wiley January 2009. ISBN: 978-0-470-17446-3.

[2.9]. D. C. Montgomery and S. M. Kowalski., *Design and analysis of experiments*. Wiley 2011. ISBN 978-0-470-16990-2.

[2.10]. Box, G. E. P. and Wilson, K.B. (1951) On the Experimental Attainment of Optimum Conditions (with discussion). Journal of the Royal Statistical Society Series B 13(1):1–45.

[2.11]. Box, G. E. P. and Draper, Norman. 2007. Response Surfaces, Mixtures, and Ridge Analyses, Second Edition [of Empirical Model-Building and Response Surfaces, 1987], Wiley.

[2.12]. *NIST/SEMATECH, e-Handbook of Statistical Methods*, http://www.itl.nist.gov/div898/handbook/, June, 2010.

[2.13]. Liu, F. "An efficient method for statistical circuit simulation," *IEEE/ACM International Conference on Computer-Aided Design(ICCAD) 2007*, pp.719-724, 4-8 Nov. 2007. San Jose.

[2.14]. Julier and J.K. Uhlmann, "Unscented Filtering and Nonlinear Estimation," Proc. IEEE, vol. 92, pp. 401 – 402, Mar. 2004.

[2.15]. J. Zhang, "The Calculating Formulae and Experimental Methods in Error Propagation Analysis," IEEE Trans. on Reliability, vol. 55, pp. 169 – 181, June 2006.

[2.16]. Julier, S. J., "The scaled unscented transformation", Proceedings of American Control conference, vol 6, pp 4555-4559. 2002.

[2.17]. W. Zhang, et.al, "Accuracy Analysis of Unscented Transformation of Several Sampling Strategies", 10th ACIS International Conference on Software Engineering, Artificial Intelligences, Networking and Parallel/Distributed Computing, pp. 377 - 380. May 2009.

[2.18]. Steiner, G.; Zangl, H.; Watzenig, D., "Generic statistical circuit design based on the unscented transformation and its application to capacitive sensor instrumentation", , 2005. ICIT 2005. IEEE International Conference on Industrial Technology, Issue: 14-17 Dec. 2005, pp. 108 - 113., Hong kong.

[2.19]. N. Nikolova, et.al. "Adjoint techniques for sensitivity analysis in High-frequency structure CAD", IEEE transaction on microwave theory and techniques. Vol. 52, No. 1, Januaray 2004. Pp 403- 419.

[2.20]. Z. Ilievski, "Adjoint transient sensitivity analysis in circuit simulation", Scientific Computing in Electrical Engineering Mathematics in Industry, 2007, Volume 11, Part II, 183-189.

[2.21]. T. Nguyen, "transient sensitivity computation for transistor level analysis and tuning", In Proceedings of the 1999 IEEE/ACM international conference on Computer-aided design (ICCAD '99). IEEE Press, Piscataway, NJ, USA, 120-123.

[2.22]. S. Director, "The generalized adjoint network and network sensitivities." IEEE transaction on Circuit theory, vol ct-16, No 3, August 1969. Pp 318-323.

[2.23]. C. A. Desoer and E. S. Kuh, *Basic Circuit Theory*, vol. 2. New York: McGraw-Hill, 1967, pp. 292-300.

Chapter 3 Interval Analysis for Gaussian Uncertainty Propagation

3.1 Introduction

An interval-value based circuit simulation engine is proposed to efficiently estimate the distributions of circuit performances. As opposed to Monte-Carlo simulation, in the proposed engine a single "interval-valued" circuit simulation is required to obtain the circuit performance distribution.

Variations in process and device parameters are first converted into these "interval" quantities. Secondly, an interval-valued SPICE simulator, in which all real number operations are replaced by interval operations, is developed to simulate a circuit. The simulation results are presented in interval forms, which can then be translated back to statistical distributions.

The algorithm is first developed for Gaussian process/device parameter variations, as described in this chapter The non-Gaussian case will be discussed in the next chapter.

The rest of the chapter is organized as follows: sections 3.2 and 3.3 give an overview of the interval quantities and their operations; section 3.4 provides the details on the proposed interval-value based circuit simulation algorithm; section 3.5 illustrates the algorithm on selected circuit examples and demonstrates the simulation accuracy; section 3.6 presents evaluations of the algorithm's scalability and runtime; section 3.7 concludes the chapter.

3.2 The Interval Quantities

The term "interval" refers to a quantity of the form [3.1] [3.2],

$$x = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i \tag{3.1}$$

where ε_i's are zero-mean, independently distributed standard normal (i.e., unit variance) random variables, referred to as the *noise variables*, while x_0 and x_i's are scalar coefficients.

By definition, the noise variables have the property,

$$\mathrm{E}(\varepsilon_i \varepsilon_j) = \begin{cases} 0 & if\ i \neq j \\ 1 & if\ i = j \end{cases} \tag{3.2}$$

Equation (3.1) can be interpreted as capturing a Gaussian distribution with mean and variance given by,

$$mean = \mathrm{E}\left(x_0 + \sum_{i=1}^{n} x_i \varepsilon_i\right) = x_0 + \sum_{i=1}^{n} x_i\, \mathrm{E}(\varepsilon_i) = x_0 \tag{3.3}$$

$$variance = \mathrm{E}\left(\sum_{i=1}^{n} x_i \varepsilon_i\right)^2 = \mathrm{E}\left(\sum_{i=1}^{n} x_i^2 \varepsilon_i^2 + 2\sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} x_i \varepsilon_i x_j \varepsilon_j\right) = \sum_{i=1}^{n} x_i^2 \tag{3.4}$$

In general, any interval quantity can be converted into a Gaussian distribution with the appropriate mean and variance using equations (3.3) and (3.4). In addition, this conversion is unique, i.e., an interval quantity can represent only one Gaussian distribution.

However, for a given Gaussian distribution, its equivalent interval representation is not unique. For example, the two interval quantities $x_1 = 1 + \varepsilon_1$ and $x_2 = 1 + \frac{\sqrt{2}}{2}\varepsilon_1 + \frac{\sqrt{2}}{2}\varepsilon_2$ represent the same Gaussian distribution with mean 1 and variance 1. In this work, we choose to use the representation that requires the minimum number of noise variables when converting a Gaussian random variable to interval. The details and implications of such a conversion procedure are given in section 3.4.1.

By considering interval quantities as capturing a certain Gaussian distribution, the covariance between two interval quantities can be calculated. For two interval quantities, *x* and *y*, with,

$$x = x_0 + \sum_{i=1}^{n} x_i \varepsilon_i$$

$$y = y_0 + \sum_{i=1}^{m} y_i \varepsilon_i$$

(3.5)

where n and m are the number of noise variables present in each interval representation, with n not necessarily equal to m. The covariance between x and y is given by

$$cov(x,y) = \mathrm{E}(x - x_o)(y - y_0) = \mathrm{E}\left(\sum_{i=1}^{n} x_i\varepsilon_i \sum_{j=1}^{m} y_j\varepsilon_j\right) = \sum_{i=1}^{\min(n,m)} x_i y_i \tag{3.6}$$

In general, for k interval quantities,

$$\begin{aligned} x_1 &= x_{10} + \sum_{i=1}^{n_1} x_{1i}\varepsilon_i \\ &\vdots \\ x_k &= x_{k0} + \sum_{i=1}^{n_k} x_{ki}\varepsilon_i \end{aligned} \tag{3.7}$$

The $k \times k$ covariance matrix can be obtained by first letting $n=\max(n_1, \ldots, n_k)$, and by re-writing the k interval quantities in matrix form, i.e.,

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix} = \begin{bmatrix} x_{10} \\ x_{20} \\ \vdots \\ x_{k0} \end{bmatrix} + \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{kn} \end{bmatrix} \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix} \tag{3.8}$$

with some of the coefficients in the matrix $\begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{kn} \end{bmatrix}$ set to zero as required for representing interval quantities with fewer than n noise variables.

Next, setting $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{bmatrix}$, $\boldsymbol{x}_0 = \begin{bmatrix} x_{10} \\ x_{20} \\ \vdots \\ x_{k0} \end{bmatrix}$, $A = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k1} & x_{k2} & \cdots & x_{kn} \end{bmatrix}$, and $\boldsymbol{\varepsilon} = \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{bmatrix}$, equation (3.8) can be further simplified to

$$\boldsymbol{x} = \boldsymbol{x}_0 + A\boldsymbol{\varepsilon} \tag{3.9}$$

The covariance matrix is obtained from equation (3.9),

$$\mathrm{E}(\boldsymbol{x}-\boldsymbol{x}_0)(\boldsymbol{x}-\boldsymbol{x}_0)^T = AA^T \tag{3.10}$$

3.3 Interval Operations

In this section, algebraic operations between interval quantities are defined. Without loss of generality, it is assumed that the interval quantities involved in the calculation have the same number of noise variables. Suppose that interval quantity *x* is represented by noise variables $\{\varepsilon_1, \varepsilon_2\}$, and interval quantity *y* is represented by noise variables $\{\varepsilon_2, \varepsilon_3\}$, then both *x* and *y* can be represented by the union of the two sets, i.e., noise variables $\{\varepsilon_1, \varepsilon_2, \varepsilon_3\}$, with zero assigned to the coefficients of the noise variables that were not present in *x* or *y*.

3.3.1 Scalar Multiplication

Multiplication of an interval quantity by a scalar constant is carried out by multiplying each coefficient of the noise variable by the scalar constant, as shown below,

$$\alpha x = \alpha\left(x_0 + \sum_{i=1}^{n} x_i\varepsilon_i\right) = \alpha x_0 + \sum_{i=1}^{n} \alpha x_i\varepsilon_i \tag{3.11}$$

where α is a scalar constant and *x* is an interval quantity. To verify the correctness of this definition, the interval quantity *x* is viewed as the equivalent of a Gaussian distribution of certain mean and variance. Equation (**3.11**) suggests that if a scalar α is multiplied by the random samples generated from this Gaussian distribution represented by *x*, the results follow another Gaussian distribution represented by αx, which is consistent with the rules of linear transformation of Gaussian random variables [3.7] [3.8].

3.3.2 Addition and Subtraction

The sum or difference of two interval quantities *x* and *y* can be obtained by adding or subtracting the corresponding coefficients of the noise variables, as follows,

$$x \pm y = \left(x_0 + \sum_{i=1}^{n} x_i\varepsilon_i\right) \pm \left(y_0 + \sum_{j=1}^{n} y_j\varepsilon_i\right) = (x_0 \pm y_0) + \sum_{i=1}^{n} (x_i \pm y_i)\varepsilon_i \tag{3.12}$$

The correctness of definition (3.12) can be verified by viewing *x* and *y* as the interval equivalent of two Gaussian distributions $N_1(m_1, \sigma_1)$ and $N_2(m_2, \sigma_2)$. The left hand side of equation (3.12) represents the distribution resulting from the sum(or difference) of two Gaussian random variables. According to probability theory [3.7][3.8], this is a Gaussian distribution with mean $m_1 + m_2$ (or $m_1 - m_2$) and variance $\sigma_1^2 + \sigma_2^2$ plus (or minus) twice the covariance between the two random variables. Such a distribution has an interval representation shown on the right hand side of equation (3.12).

3.3.3 **Multiplication**

When two interval quantities, *x* and *y* are multiplied together, the following expression is obtained,

$$xy = \left(x_0 + \sum_{i=1}^{n} x_i \varepsilon_i\right)\left(y_0 + \sum_{i=1}^{n} y_i \varepsilon_i\right) = x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + y_0 x_i)\varepsilon_i + \sum_{i=1}^{n}\sum_{j=1}^{n} x_i y_j \varepsilon_i \varepsilon_j \tag{3.13}$$

The term $\sum_{i=1}^{n}\sum_{j=1}^{n} x_i y_j \varepsilon_i \varepsilon_j$ in equation (3.13) is not a linear combination of noise variables. Therefore, the exact result of *xy* cannot be represented by an interval quantity. This is consistent with the observation that when two Gaussian random variables are multiplied together, the resulting distribution does not follow a Gaussian distribution [3.10].

With the aim to define all interval operations within the domain of intervals, a novel moment-preserving algorithm is developed to approximate the exact result of *xy* by an interval quantity.

As a first step, the result of *xy* is approximated by an interval quantity shown in equation (3.14). In this approximation, a new noise variable, ε_{n+1}, that has not been used in the definition of either *x* and *y* is introduced. Two additional scalar coefficients k_0 and k_1 are also introduced.

$$xy_{approx} = x_0 y_0 + \sum_{i=1}^{n} (x_0 y_i + y_0 x_i)\varepsilon_i + k_0 + k_1 \varepsilon_{n+1} \tag{3.14}$$

The values of the scalar coefficients k_0 and k_1 are determined by imposing moment preservation constraints shown in equation (3.15). By solving for the values of k_0 and k_1 that satisfy these constraints, the interval approximation is forced to be the Gaussian approximation that preserves the first and second moments of the distribution represented by the exact result.

$$\begin{aligned} \mathrm{E}(xy) &= \mathrm{E}\left(xy_{approx}\right) \\ \mathrm{E}(xy)^2 &= \mathrm{E}\left(xy_{approx}\right)^2 \end{aligned} \tag{3.15}$$

Substituting equations (3.13) and (3.14) into equation (3.15), and carrying out the expectation computations,

$$k_0 = \sum_{i=1}^{n} x_i y_i$$

$$k_1 = k_0 \left[k_0 + \left(\sum_{i=1}^{n} x_i^2 \right) \left(\sum_{i=1}^{n} y_i^2 \right) \right]$$

(3.16)

The 1st and 2nd moment-preserving interval multiplication is therefore defined as,

$$xy = x_0 y_0 + \sum_{i=1}^{n} x_i y_i + \sum_{i=1}^{n} (x_0 y_i + y_0 x_i) \varepsilon_i + k_0 \left[k_0 + \left(\sum_{i=1}^{n} x_i^2 \right) \left(\sum_{i=1}^{n} y_i^2 \right) \right] \varepsilon_{n+1}$$

(3.17)

3.3.4 **General Interval Operations**

In this section, general nonlinear operations between interval quantities are defined.

Any algebraic operation can be expanded into a series of additions/subtractions and multiplications by using Taylor Series Expansion. Using the definition for interval additions/subtractions and multiplication from previous sections, any interval algebraic operation can be performed.

For example, exponentiation of a zero-mean interval quantity *x* can be expanded as

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \cdots$$

(3.18)

which contains only additions and multiplications. In general, any operations involving interval quantities, $x_1, \ldots, x_n$ can be expanded around their mean values $x_{10}, \ldots, x_{n0}$ as

$$f(x_1, \ldots, x_n) = f(x_{10}, \ldots, x_{n0}) + \sum_{i=1}^{n} \frac{\partial f}{\partial x_i} (x_i - x_{i0}) + \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial f}{\partial x_i} \frac{\partial f}{\partial x_j} (x_i - x_{i0})(x_j - x_{j0}) + \cdots$$

(3.19)

which contains only multiplications and additions/subtractions. In practice, however, since our 1st and 2nd moment preserving requirement yields a system of two equations, it can only be used to estimate two additional constants. For this reason, and as we will show next, Taylor expansions are truncated so that all the non-linear terms are represented by an additional interval value that in defined by the two constants that will be "fixed" by the moment preserving requirement.

Next, some frequently used operations are defined directly using moment preservation algorithms, as detailed in sections 3.4.1.1 through 3.4.1.4.

3.4.1.1 Inversion/Division

The inverse of an interval quantity is given as,

$$\frac{1}{x} = \frac{1}{x_0 + \sum_{i=1}^{n} x_i \varepsilon_i} \tag{3.20}$$

By performing a second order Taylor series expansion on (3.20), the following is obtained,

$$\frac{1}{x} = \frac{1}{x_0} - \frac{1}{x_0^2}\sum_{i=1}^{n} x_i \varepsilon_i + \frac{1}{x_0^3}\left(\sum_{i=1}^{n} x_i \varepsilon_i\right)^2 \tag{3.21}$$

The interval approximation to equation (3.21) is given by

$$\left(\frac{1}{x}\right)_{approx} = \frac{1}{x_0} - \frac{1}{x_0^2}\left(\sum_{i=1}^{n} x_i \varepsilon_i\right) + k_0 + k_1 \varepsilon_{n+1} \tag{3.22}$$

where k_0 and k_1 are constants to be determined. Calculating the first and second moments of equation (3.21) gives

$$\mathrm{E}\left(\frac{1}{x}\right) = \frac{1}{x_0} + \frac{1}{x_0^3}\sum_{i=1}^{n} x_i^2 \tag{3.23}$$

And

$$\mathrm{E}\left(\frac{1}{x}\right)^2 = \left(\frac{1}{x_0} + \frac{1}{x_0^2}\sum_{i=1}^{n} x_i^2\right)^2 + \frac{1}{x_0^6}\left(-3\sum_{i=1}^{n} x_i^4 + 6\sum_{i=1}^{n}\sum_{j=1}^{n} x_i^2 x_j^2\right) \tag{3.24}$$

Imposing the moment preservation constraints on equation (3.22) gives

$$\mathrm{E}\left(\frac{1}{x}\right)_{approx} = \mathrm{E}\left(\frac{1}{x}\right)$$

$$\mathrm{E}\left(\frac{1}{x}\right)^2_{approx} = \mathrm{E}\left(\frac{1}{x}\right)^2$$

(3.25)

where the right hand side is given by equations (3.23) and (3.24). Solving the system of equations in (3.25) for the unknown coefficients k_0 and k_1 yields

$$k_0 = \frac{1}{x_0} + \frac{1}{x_0^3}\sum_{i=1}^{n} x_i^2$$

$$k_1 = \frac{1}{x_0^3}\sqrt{5\sum x_i^2 - 3\sum x_i^4}$$

(3.26)

Division between two interval quantities x and y can be calculated as a multiplication between x and the inversion of y.

3.4.1.2 Power

The second order Taylor series expansion of an interval quantity x raised to a constant scalar power α is given by,

$$x^\alpha = \left(x_0 + \sum_{i=1}^{n} x_i\varepsilon_i\right)^\alpha = x_0^\alpha + \alpha x_0^{\alpha-1}\sum_{i=1}^{n} x_i\varepsilon_i + \frac{\alpha(\alpha-2)x_0^{\alpha-3}}{2}\left(\sum_{i=1}^{n} x_i\varepsilon_i\right)^2$$

(3.27)

The interval approximation to the right hand side of equation (3.27) is

$$x^\alpha_{approx} = x_0^\alpha + \alpha x_0^{\alpha-1}\sum_{i=1}^{n} x_i\varepsilon_i + k_0 + k_1\varepsilon_{n+1}$$

(3.28)

The first moment of the right hand side of equation (3.27) is given by

$$x_0^\alpha + \frac{\alpha(\alpha-2)x_0^{\alpha-3}}{2}\sum_{i=1}^{n} x_i^2$$

(3.29)

and the second moment is given by

$$x_0^{2\alpha} + \alpha^2 x_0^{2\alpha-2} \sum_{i=1}^{n} x_i^2 + 2x_0^{\alpha} \frac{\alpha(\alpha-2)x_0^{\alpha-3}}{2} \sum_{i=1}^{n} x_i^2$$

$$+ \frac{\alpha^2(\alpha-2)^2 x_0^{2\alpha-6}}{4} \left(-3 \sum_{i=1}^{n} x_i^4 + 6 \sum_{i=1}^{n} \sum_{j=1}^{n} x_i^2 x_j^2 \right)$$

(3.30)

Imposing the moment preservation constrains by equating the first and second moments of equation (3.28) to equation (3.29) and (3.30), respectively, yields,

$$k_0 = \frac{\alpha(\alpha-2)x_0^{\alpha-3}}{2} \sum_{i=1}^{n} x_i^2$$

$$k_1 = \frac{\alpha(\alpha-2)x_0^{\alpha-3}}{2} \sqrt{-3 \sum_{i=1}^{n} x_i^4 + 5 \sum_{i=1}^{n} \sum_{j=1}^{n} x_i^2 x_j^2}$$

(3.31)

3.4.1.3 Exponentiation

The second order Taylor expansion of exponentiation of an interval quantity *x* is,

$$e^x = e^{x_0} \left(1 + \sum_{i=1}^{n} x_i \varepsilon_i + \left(\sum_{i=1}^{n} x_i \varepsilon_i \right)^2 \right)$$

(3.32)

The interval approximation of the right hand side of equation (3.32) is given by:

$$e_{approx}^{x} = e^{x_0} + e^{x_0} \sum_{i=1}^{n} x_i \varepsilon_i + k_0 + k_1 \varepsilon_{n+1}$$

(3.33)

Applying moment preservation constraints between equations (3.32) and (3.33) gives

$$k_0 = e^{x_0} \sum_{i=1}^{n} x_i^2$$

$$k_1 = e^{x_0} \sqrt{-3 \sum_{i=1}^{n} x_i^4 + 5 \sum_{i=1}^{n} \sum_{j=1}^{n} x_i^2 x_j^2}$$

(3.34)

3.4.1.4 Logarithm

The second order Taylor expansion of natural logarithm of an interval quantity *x* is,

$$\ln x = \ln x_0 \ + \frac{1}{x_0} \sum_{i=1}^{n} x_i \varepsilon_i - \frac{1}{2x_0^2} \left(\sum_{i=1}^{n} x_i \varepsilon_i \right)^2$$

(3.35)

The interval approximation of the right hand side of equation (**3.35**) is given by:

$$(\ln x)_{approx} = \ln x_0 + \frac{1}{x_0} \sum_{i=1}^{n} x_i \varepsilon_i + k_0 + k_1 \varepsilon_{n+1}$$

(3.36)

Applying moment preservation constraints between equations (**3.35**) and (**3.36**) yields

$$k_0 = -\frac{1}{2x_0^2} \sum_{i=1}^{n} x_i^2$$

$$k_1 = \frac{1}{2x_0} \sqrt{-3 \sum_{i=1}^{n} x_i^4 + 5 \sum_{i=1}^{n} \sum_{j=1}^{n} x_i^2 x_j^2}$$

(3.37)

3.3.5 **Accuracy of Interval Operations**

Linear interval operations, i.e., addition and subtraction, are by construction exact operations.

Interval multiplications are by construction exact up to the second statistical moments. Table 3.1 shows the numerical simulation results of various interval multiplications. In this table, the first four moments of the exact multiplication result,

simulated using 50,000 runs of Monte-Carlo simulations, is compared to that obtained from the interval multiplication approximation. Figure 3.1 compares the probability density function (*pdf*) obtained from the direct multiplication result, (simulated via MC) with that from the interval approximation for each of the column shown in Table 3.1. The small discrepancies observed in mean and variance between the Monte-Carlo simulations and interval multiplications are due to the random sample generation and the finite samples sizes used for the Monte-Carlo simulations.

Table 3.1: Comparison of the first four moments of the distribution obtained from interval multiplications to that from the exact multiplication result. The aim of the used approximation is to preserve only the first two moments (i.e. Mean and Variance).

Case #	1	2	3	4	5	6
Mean(MC)	0	-5.1253	-5.6939	27.9259	2.5187	-16.8955
Mean(interval)	0	-5.1200	-5.6940	28.000	2.6000	-16.9000
Relative error (%)	0	-0.1535	-0.0839	0.3441	0.5202	-0.0608
Variance(MC)	9.041	12.1285	8.4161	513.533	414.606	449.680
Variance(interval)	9.000	12.1342	8.4572	516.000	413.680	446.970
Relative error (%)	0.45	0.8715	0.0321	0.2796	0.7297	0.0636
Skewness(MC)	0	-0.89525	0.586979	1.02278	0.701181	-1.55943
Skewness(interval)	0	0	0	0	0	0
Relative error (%)	0	100	100	100	100	100
Kurtosis(MC)	9.084	4.06569	3.486760	4.3577	9.0631	6.525937
Kurtosis(interval)	3	3	3	3	3	3
Relative error (%)	66.97	26.21	13.69	31.16	66.90	54.03

* Skewness is calculated as $\frac{\mathrm{E}(x-\mu)^3}{\sigma^3}$ and kurtosis is calculated as $\frac{\mathrm{E}(x-\mu)^4}{\sigma^4}$

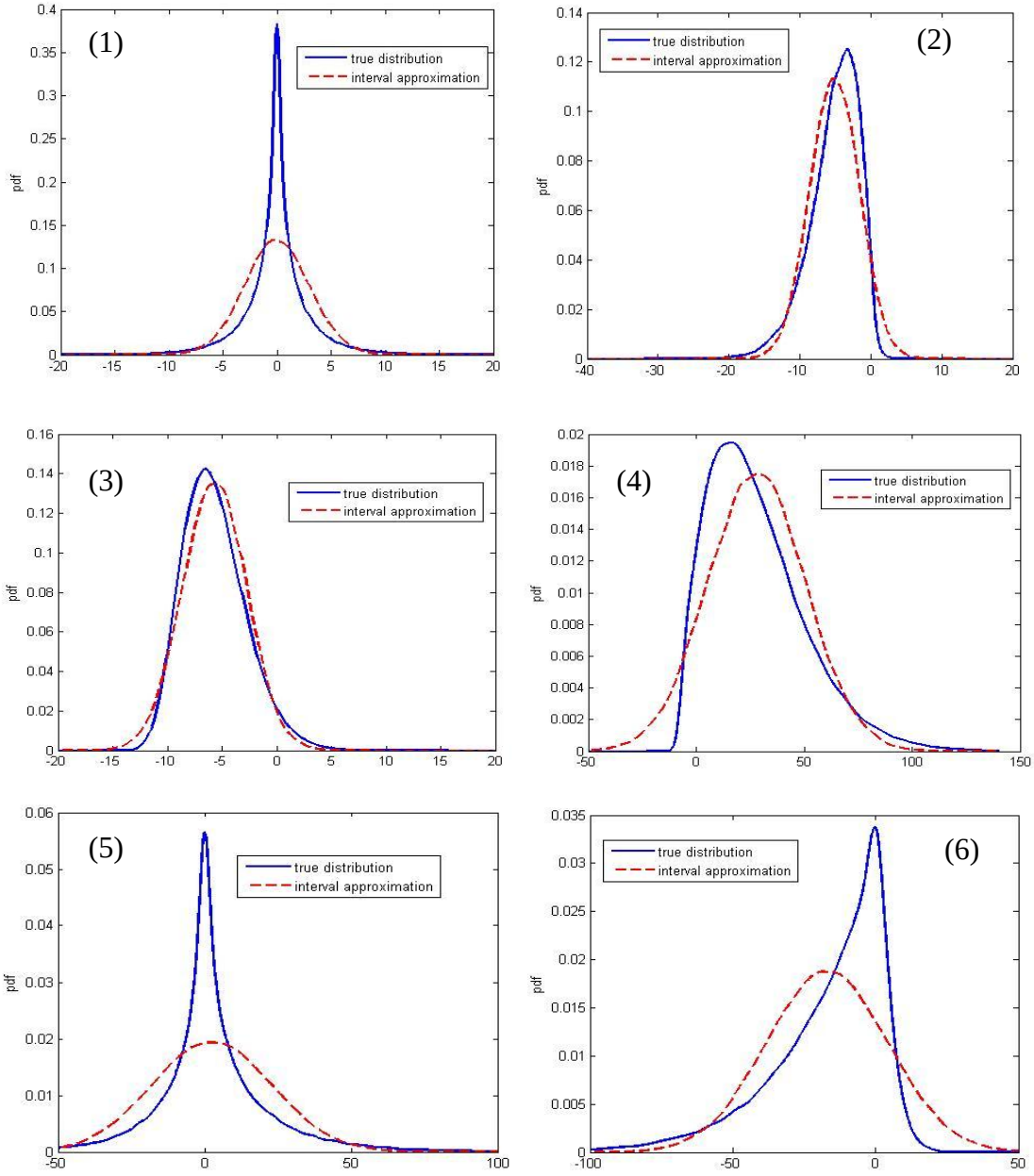

Figure 3.1: Comparison of the true distribution and the interval approximation after a multiplication operation. The 1st & 2nd moment-preserving algorithm is always forcing a Gaussian approximation to the multiplication result.

For other nonlinear interval operations discussed in section 3.3.4, a second order Taylor series expansion is first performed before moment preservation constraints are imposed. Therefore, the interval approximation is exact in terms of capturing the first and second order moments of the Taylor series expansion. However, truncation error exists between the Taylor series expansion and the original nonlinear operation. As an example,

the error introduced in interval square root operation is summarized in Table 3.2. The distribution of the exact result is simulated by Monte-Carlo simulation using 50,000 samples.

Table 3.2: Comparison of interval square root operation accuracy to Monte-Carlo(MC) simulations.

Quantity subject to square root operation	Mean (MC)	Mean (interval)	Relative Error	Std. Dev. (MC)	Std. Dev. (interval)	Relative Error
$23+4\varepsilon_1$	4.7775	4.7867	0.192%	0.4229	0.4187	0.978%
$30+10\varepsilon_1+2\varepsilon_2$	5.3845	5.4365	0.965%	1.0076	0.9460	6.114%
$20+0.5\varepsilon_1+2\varepsilon_2+3\varepsilon_3$	4.4532	4.4628	0.216 %	0.4133	0.4088	1.094%
$3+0.3\varepsilon_1$	1.7299	1.7310	0.064%	0.0871	0.0867	0.385%
$8+2\varepsilon_1+1\varepsilon_2+2\varepsilon_3+1\varepsilon_4$	2.7598	2.7996	1.455%	0.6266	0.5715	8.784%
$170+12\varepsilon_1+13\varepsilon_2$	13.0199	13.0296	0.074%	0.6814	0.6794	0.294%
$20.34+1.32\varepsilon_1+6.34\varepsilon_2$	4.4437	4.4807	0.832%	0.7673	0.7281	5.111%
$78.3+13\varepsilon_1+2\varepsilon_2$	8.8162	8.8330	0.190%	0.7535	0.7460	0.988%
$19.12+0.53\varepsilon_1+2\varepsilon_2+0.3\varepsilon_3$	4.3664	4.3694	0.068%	0.2401	0.2394	0.259%
$5.23+3.3\varepsilon_1$	2.1467	2.2229	4.566%	0.8844	0.7687	13.082%
$90+22\varepsilon_1+13\varepsilon_2+3.32\varepsilon_3+4.54\varepsilon_4$	9.3762	9.4356	0.634%	1.4493	1.3951	3.735%
$5+2\varepsilon_1+1.3\varepsilon_2$	2.1537	2.2022	2.3650%	0.6278	0.5514	12.170%

It should be noted that even though the error introduced in a single interval operation may be large, the error after a number of interval operations involving many interval quantities may be smaller due to the central limit theorem. In the context of circuit simulation, the resulting circuit performance distribution can resemble a Gaussian distribution as the variations in process/device parameters are propagated through circuit simulation steps. In this case, the proposed interval operations are sufficiently accurate to capture the final performance distribution.

3.4 Interval-valued Circuit Simulation

The general flow of the interval-valued circuit simulation algorithm is summarized in Figure 3.2. The inputs to the algorithm are process/device parameter variations specified in terms of a multivariate Gaussian distribution, i.e., mean vector and covariance matrix. The algorithm then converts these distributions into interval representations, and passes the intervals into a SPICE-like circuit simulator. The simulator is developed for handling interval calculations and producing the circuit performances in interval forms. The interval valued circuit performance can then be converted to distributions for statistical timing, yield analysis and design centering optimizations.

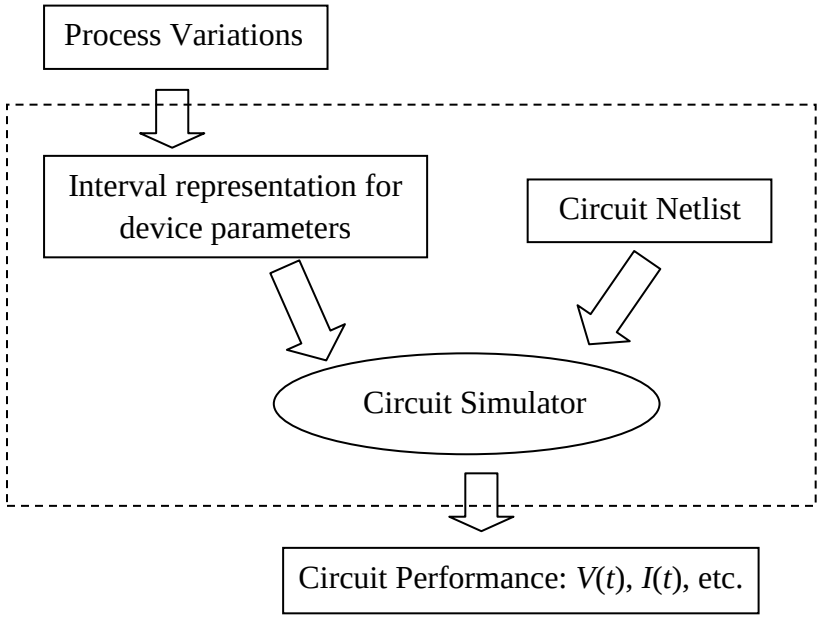

Figure 3.2: Overall flow of the interval-valued circuit simulation program.

3.4.1 **Conversion of Process Variations into Intervals**

3.4.1.1 Process Variation Specified using Hierarchical Variation Model

Process variations are often captured using a hierarchical model. Examples of such model can be found in [3.3].

To simplify the discussion, a two-level hierarchical model with global and local variation is used. The method outlined here can be extended to a model with an arbitrary level of hierarchies by assigning appropriate noise variables to each level of hierarchy.

In the case of a two-level hierarchal model, a noise variable common to all devices on a die is used to represent the global variation, while noise variables particular to each device on a die are used to represent the local variation. In mathematical form, let the parameter of interest be p, and assuming the devices on a die are numbered from 1,…, n, then the parameter value for each device in interval representation is given by,

$$\begin{aligned} p_1 &= p_{01} + \sigma_g \varepsilon_{global} + \sigma_l \varepsilon_{local1} \\ &\vdots \\ p_n &= p_{0n} + \sigma_g \varepsilon_{global} + \sigma_l \varepsilon_{localn} \end{aligned} \tag{3.38}$$

where σ_g and σ_l are the standard deviations of the global and local variations, respectively.

3.4.1.2 Process Variation Specified using Correlation

In addition to global and local variations, variation in process parameters can also be captured by using variance-covariance information. Assuming the availability of a variance-covariance matrix Σ, which can be derived, for example, from test chip measurements, the parameters $p_1,\ldots,p_n$ that give rise to such correlation structure can be written in interval forms as shown in equation (3.39), where $\varepsilon_1, \ldots, \varepsilon_n$ are the noise variables; $p_{1i}, \ldots, p_{ii}$ for $i=1,..,n$ are scalar coefficients whose values are to be determined from the variance-covariance matrix Σ.

$$\begin{aligned} p_1 &= p_{01} + p_{11}\varepsilon_1 \\ p_2 &= p_{02} + p_{21}\varepsilon_1 + p_{22}\varepsilon_2 \\ &\vdots \\ p_n &= p_{0n} + \sum_{i=1}^{n} p_{ni}\varepsilon_i \end{aligned} \tag{3.39}$$

The interval representation for $p_1,\ldots,p_n$ is not unique for a given covariance matrix. For example, the following is also a feasible interval representation:

$$\begin{aligned} p_1 &= p_{01} + \sum_{i=1}^{n} p_{1i}\varepsilon_i \\ p_2 &= p_{02} + \sum_{i=1}^{n} p_{2i}\varepsilon_i \\ &\vdots \\ p_n &= p_{0n} + \sum_{i=1}^{n} p_{ni}\varepsilon_i \end{aligned} \tag{3.40}$$

where the coefficients to be determined are $p_{1i}, \ldots, p_{ni}$ for $i=1,..,n$.

In general, since there are at most $\frac{n^2+n}{2}$ distinct entries in a covariance matrix, the minimum number of coefficients required in a feasible interval representation for $p_1,\ldots,p_n$ is $\frac{n^2+n}{2}$. The representation shown in equation (3.39) has exactly $\frac{n^2+n}{2}$ unknown coefficients, whereas the representation in equation (3.40) has n^2 unknown coefficients. Therefore, for minimum cost, the representation shown in equation (3.39) is adopted.

In order to determine the value of the coefficients in (3.39) from the covariance matrix Σ, the coefficients are first collected into an lower triangular matrix L such that

$$L = \begin{bmatrix} p_{11} & 0 & \cdots & 0 \\ p_{21} & p_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{bmatrix}$$

(3.41)

Equation (3.39) is then re-written in a matrix form using L,

$$\begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} = \begin{bmatrix} p_{01} \\ \vdots \\ p_{0n} \end{bmatrix} + L \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_n \end{bmatrix} \tag{3.42}$$

The covariance matrix for (3.42) is calculated as

$$\Sigma_{interval} = LL^T \tag{3.43}$$

Equating (3.43) to the desired covariance matrix Σ to obtain,

$$LL^T = \Sigma \tag{3.44}$$

Equation (3.44) can be solved to find the matrix L. A Cholesky decomposition on the covariance matrix Σ gives the matrix L directly. The cost of computing a Cholesky decomposition is $n^3/3$ flops, where n is the number of parameters.

3.4.2 **Interval-valued Circuit Simulator**

After casting all process variations into their respective interval representations, the circuit of interest is simulated using a SPICE-like simulation engine to obtain the circuit performances. However, unlike the conventional SPICE simulator for deterministic circuit analysis, the simulator used in this work handles calculations and simulations involving interval quantities.

Figure 3.3 shows the simulation steps of a transient circuit analysis implemented in conventional deterministic SPICE simulator, where t denotes time in a transient analysis, and h denotes the simulation step size.

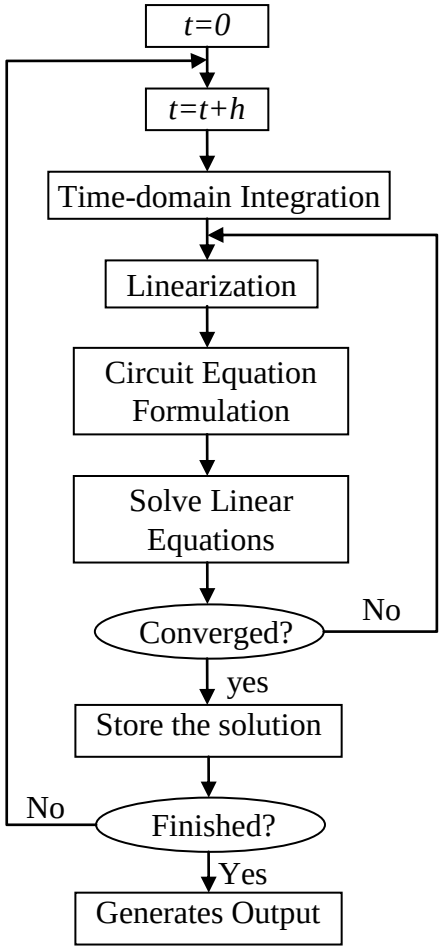

Figure 3.3: Flowchart of the conventional SPICE transient circuit simulation program. (Reproduced from [3.5]).

The interval-valued simulator follows a similar flow as that shown in Figure 3.3, except that selected steps in this flow are modified to handle interval quantities. To illustrate these modifications, consider for example of a simple RC circuit shown in Figure 3.4, where the nodes are labeled 0, 1 and 2. The resistance and capacitance values are *R* and *C*, respectively.

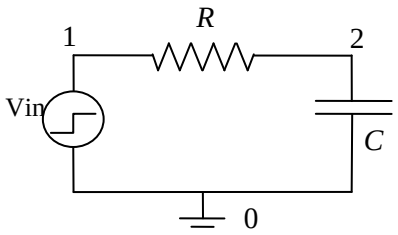

Figure 3.4: Example Circuit to illustrate transient SPICE simulation

The device equation that governs the resistor is

$$V = IR \tag{3.45}$$

and the equations governing the capacitor are:

$$V = Q/C$$
$$I = dQ/dt \tag{3.46}$$

where V and I denotes the voltage and current, respectively; Q denotes the total stored charges.

Starting with t=0 and initializing all voltages and currents to zero, the first step in a transient simulation is to perform a time-domain integration of device equations from t to $t+h$. The goal of such integration is to obtain a single expression that relates the device voltage and current at time $t+h$. In the RC example, the capacitor equations are integrated using a finite difference formula to obtain,

$$V_{t+h} = V_t + h\frac{d}{dt}\left(\frac{Q_{t+h}}{C}\right) = V_t + \frac{h}{C}I_{t+h} \tag{3.47}$$

where V_{t+h} , I_{t+h} and Q_{t+h} are the voltage, current and charges at time $t+h$, respectively. No time-domain integration is performed on the resistor equation since it is independent of time.

In the time-domain integration step, device equations are manipulated analytically without any numerical calculations involved. Therefore, this step is the same for both deterministic simulation and interval-valued simulation.

The next step in a transient simulation is a linearization step where the voltage-current equations obtained from time-domain integrations are linearized. In this RC example, equation (3.47) is already a linear function of V_{t+h} and I_{t+h}, so no additional linearization is required. In general, for a nonlinear voltage-current equation given by,

$$I_{t+h} = f(V_t, I_t, V_{t+h}) \tag{3.48}$$

where f is any nonlinear function. The linearized voltage-current equation is given by,

$$I_{t+h} = I_{eq} + G_{eq}V_{t+h} \tag{3.49}$$

with

$$Geq = \frac{df(V_t, I_t, V_{t+h})}{dV_{t+h}}$$
$$Ieq = f(V_t, I_t, V_{t+h}) - GeqV_{t+h} \tag{3.50}$$

Geq and *Ieq* can be a function of both V_t, I_t and V_{t+h}. The values for V_t, I_t are obtained from the simulation results of the previous time step. The values for V_{t+h} are assigned by using initial guesses obtained from the previous time step.

In the interval-valued circuit simulator, V_t and I_t are obtained from the previous time step and are interval quantities. The evaluations of *Geq* and *Ieq* therefore involve applying interval operations as outlined in section 3.3. However, it should be noted that the analytical expressions for *Geq* and *Ieq* are not different from those used in the deterministic circuit simulator.

Once a linearized circuit is obtained, a system of linear equations is established to describe the circuit topology based on Kirchhoff's current law and voltage law. In the RC example, the system is as follows

$$\begin{bmatrix} 1/R & -1/R & 1 \\ -1/R & 1/R + C/h & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_{1,t+h} \\ V_{2,t+h} \\ I_{10,t+h} \end{bmatrix} = \begin{bmatrix} 0 \\ CV_{2,t}/h \\ V_{in} \end{bmatrix} \tag{3.51}$$

where $V_{1,t+h}$, $V_{2,t+h}$ are the voltages at node 1 and 2 at time $t+h$; $I_{10,t+h}$ is the current from node 1 to 0 at time $t+h$; $V_{2,t}$ is the voltage at node 2 at time t. The known parameters in (3.51) are R, C, h and $V_{2,t}$. The values for $V_{1,t+h}$, $V_{2,t+h}$, $I_{10,t+h}$ are to be solved using standard procedure for a linear system of equations; one example of such procedure is *LU* decomposition coupled with backward substitution. Large, sparse systems can additionally benefit from more specialized procedures.

In the interval-valued circuit simulator, the linear system of equations is setup in the same way as in the deterministic case; however, the entries in the matrix are interval quantities. To obtain a solution, an interval-valued *LU* decomposition and backward substitution is required. *LU* decompositions and backward substitutions can be implemented using addition/subtraction and multiplication/division. Since each of those operations has an interval version defined, the interval-valued *LU* decomposition and backward substitution can be easily implemented.

After solving for the voltages and currents for time $t+h$, the solution is compared with the initial guess that was used in the linearization step when evaluating *Geq* and *Ieq*. In a deterministic simulator, if the differences between the solution and the initial guess are within a specific tolerance value *TOL*, the simulation is said to be converged, and the solution is stored for that time step. The simulation then moves to the next time step. Otherwise, *Geq* and *Ieq* are re-calculated using the voltage and current values obtained from the solution, and a new linear system of equations is formed and solved. The solution of the new system is again compared to the values used in calculating *Geq* and *Ieq* for convergence check.

In the interval version of the circuit simulation, the criteria for convergence is modified to handle interval quantities. For interval quantities x and y to agree with each other within a tolerance of *TOL*, their difference $z=x-y$ must satisfy the condition $|\mu_z \pm 3\sigma_z| \leq TOL$, where μ_z and σ_z is the mean and standard deviation, respectively, of the distribution represented by the interval quantity z.

3.4.3 **Analyzing Simulation Output**

Once a circuit is simulated using the interval-valued circuit simulator, the nodal voltages and branch currents at each time step of the circuit are obtained and stored in interval forms. Circuit performances can be calculated by applying any performance function on these nodal voltages and branch currents using interval operations.

In addition, performances represented in intervals can be converted to Gaussian distributions by applying equations $mean$=E$(x_0 + \sum_{i=1}^{n} x_i \varepsilon_i) = x_0 + \sum_{i=1}^{n} x_i$ E$(\varepsilon_i) = x_0$ (3.3) and$variance$=E $(\sum_{i=1}^{n} x_i \varepsilon_i)^2$ = E$\left(\sum_{i=1}^{n} x_i^2 \varepsilon_i^2 + 2\sum_{i=1}^{n}\sum_{j=1, j\neq i}^{n} x_i \varepsilon_i x_j \varepsilon_j\right) = \sum_{i=1}^{n} x_i^2$

(3.4).

3.4.4 **Pruning of Noise Variables**

When a circuit is simulated using the flow in Figure 3.3, at each time step of the simulation, the *required* interval quantities are the voltage/current values obtained from the current and previous time step, and the process/device parameters. If the circuit has *m* circuit elements and *p* interval device parameters, then the total number of *required* interval quantities is *k=2m+p*. The *required* interval quantities are differentiated from the *intermediate* interval quantities, which are the result of the intermediate interval calculations during the simulation. The *intermediate* interval quantities can be discarded once the correct values of the *required* interval quantities are obtained.

By viewing interval quantities as the equivalent of Gaussian random variables, the procedure in section 3.4.1.2 suggests that the maximum number of noise variables required for representing *n* interval quantities is *n*. Interval quantities that use more than *n* noise variables can be reduced to a representation that involves only *n* noise variables while still representing the same Gaussian distributions. In the context of interval-valued circuit simulation, the maximum number of noise variable required at each iteration is the same as the number of *required* interval quantities, i.e., *2m+p*.

In practice, however, the number of noise variables can be significantly larger than *2m+p* due to the introduction of *intermediate* interval values during the circuit simulation. The *intermediate* values are only important in obtaining the *required* values and can be thrown away once the *required* values are calculated. Thus, one can reduce the number of variables by restructuring the interval representations for the *required* values at the end of each iteration in the simulation. Such restructuring can significantly reduce the number of noise variables involved in an interval operation. Section 3.6 will show that the interval-valued circuit simulation runtime is directly proportionate to the number of noise variables involved in an interval operation.

The restructuring of the interval representation of the *required* quantities is carried out by first obtaining the mean and variance-covariance matrix using equation (3.3), (3.4) and (3.10); and then constructing the interval representations from the mean and variance-covariance matrix using the procedure outlined in section 3.4.1.2.

3.5 Illustrative Examples and Simulation Accuracy

In the following sections, example circuits are simulated using the proposed interval-based statistical circuit simulation algorithm, and the simulation results are compared to Monte-Carlo simulations for accuracy verification. A large number of samples (i.e., 10,000 and 50,000) are used in the Monte-Carlo simulations to make sure that the Monte-Carlo simulations match the true distributions as close as possible. For circuits with relatively large number of parameters, a Monte-Carlo simulation with 50,000 samples is used to minimize the errors caused by random sampling.

3.5.1 RC Ladder Circuit

To illustrate the algorithm outlined in section 3.4, a simple RC ladder circuitry is simulated using the interval-value based circuit simulator. The result is compared against Monte-Carlo simulations to verify its accuracy.

The schematic of the circuit which consists of 100 stage RCs is shown in

Figure 3.5. The resistors and capacitors are labeled from 1 to 100. In addition, it is assumed that this RC ladder circuit is situated across the die such that R_1 and R_{100} are at the edge of the die while R_{50} is in the center of the die. Analysis of this type of circuit is very common in understanding the behavior of IC interconnects. In this example, the performance of interest is the transient behavior of the output voltage across the last capacitor in the ladder when a step input is applied.

For illustration purposes , a two-level hierarchical process variability model, i.e. global and local, is assumed for the resistance value in the circuit. In addition, a systematic die-level deterministic variation is assumed to impact the resistance values of the ladder. The systematic variation has a smooth, second order parabolic-shape, with zero variation at the edges of the die and reaches a maximum of 2% variations at the center of the die. The complete formulations of the resistance values in terms of interval quantities are shown below,

$$\begin{aligned} R_1 &= R_0 + 0.0\%\, R_0 + 3\% R_0 \varepsilon_g + 1\% R_0 \varepsilon_1 \\ &\vdots \\ R_{50} &= R_0 + 2.0\% R_0 + 3\% R_0 \varepsilon_g + 1\% R_0 \varepsilon_{50} \\ &\vdots \\ R_{100} &= R_0 + 0.0\% R_0 + 3\% R_0 \varepsilon_g + 1\% R_0 \varepsilon_{100} \end{aligned} \tag{3.52}$$

This circuit with interval-valued resistances is then simulated using the simulator described in Section 3.4.2. At the end of each time step the output voltage across C_{100} is stored as an interval quantity.

Figure 3.6 plots the statistics (i.e., the mean, ±1σ and ±3σ of the distribution) of the distribution represented by the interval quantity *Vout* obtained from interval circuit simulations, and compared them against that obtained from a 50,000-sample Monte Carlo simulation. A good match (with error less than 1.5%) in the statistics of the output waveform is observed between the interval simulation and Monte Carlo simulation.

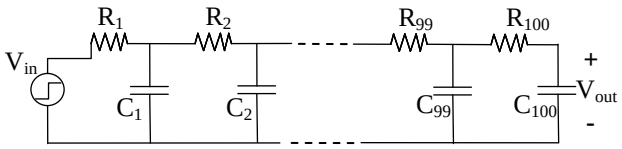

Figure 3.5: RC ladder test circuit considered for illustrating the interval valued circuit simulation.

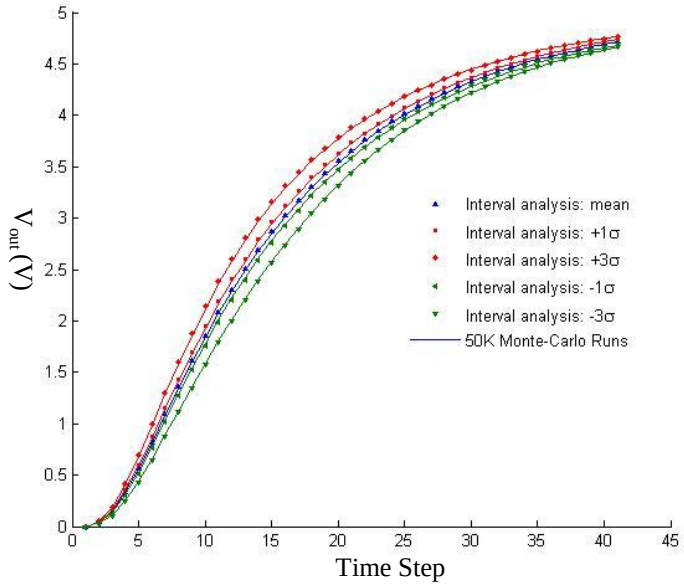

Figure 3.6: Distribution of output voltage transient response obtained from interval-value based simulation and from 50,000 runs of Monte-Carlo analysis.

3.5.2 **Transistor Circuit**

As a second example, the transistor circuit shown in Figure 3.7 is simulated using interval circuit simulations. The performance of interested is the transient response of the output voltage *Vout* due to a step input. In addition, the transistor threshold voltage V_{th} and the transistor drive strength k are assumed to be subjected to process variations, and have the following interval representation,

$$V_{th} = V_{th0} + 0.2V_{th0}\varepsilon_1 + 0.1V_{th0}\varepsilon_c$$
$$k = k_0 + 0.2k_0\varepsilon_2 + 0.2\,k_0\varepsilon_c \tag{3.53}$$

The circuit is then simulated using interval simulations and the transient response of the output voltage is plotted in Figure 3.8. The response obtained from 10,000-sample Monte Carlo simulation is also plotted on the same graph. The error is within 1% during the entire transition time period.

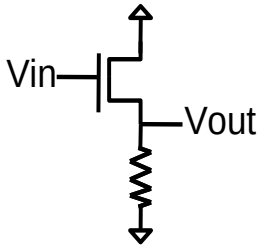

Figure 3.7: Simple transistor circuit analysis with device/process parameter variations.

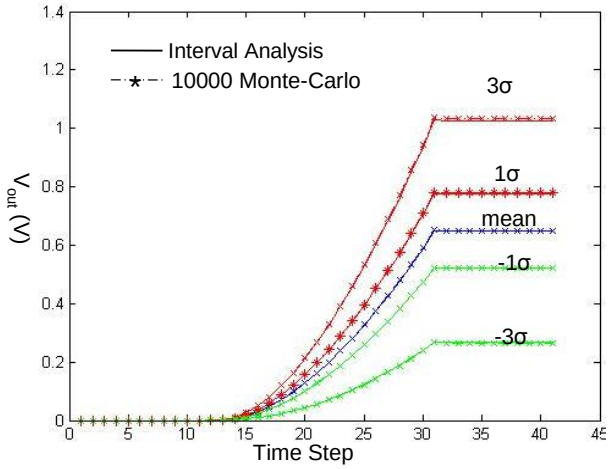

Figure 3.8: Distribution of the output voltage transient response obtained from interval-valued based simulation on the transistor circuit in Figure 3.7. Comparison is made against a 10,000-sample Monte-Carlo analysis.

3.5.3 **Comparison of Simulation Accuracy**

The accuracy of the interval-valued circuit simulator is compared to the response surface modeling technique [3.4][3.6]. Figure 3.9 plots the output waveform distributions for the transistor circuit in Figure 3.7 obtained from (a) 10,000 Monte-Carlo analysis; (b) first order response surface modeling with central-composite design, and (c) the interval-value based analysis. For the response surface modeling, six points are selected on the output transient waveform and a response surface model is obtained for each of the six points.

With respect to the Monte Carlo simulations, the first order response surface modeling technique has a maximum estimation error of 30-40%. On the other hand, the maximum estimation error for the interval-value based analysis is below 1%.

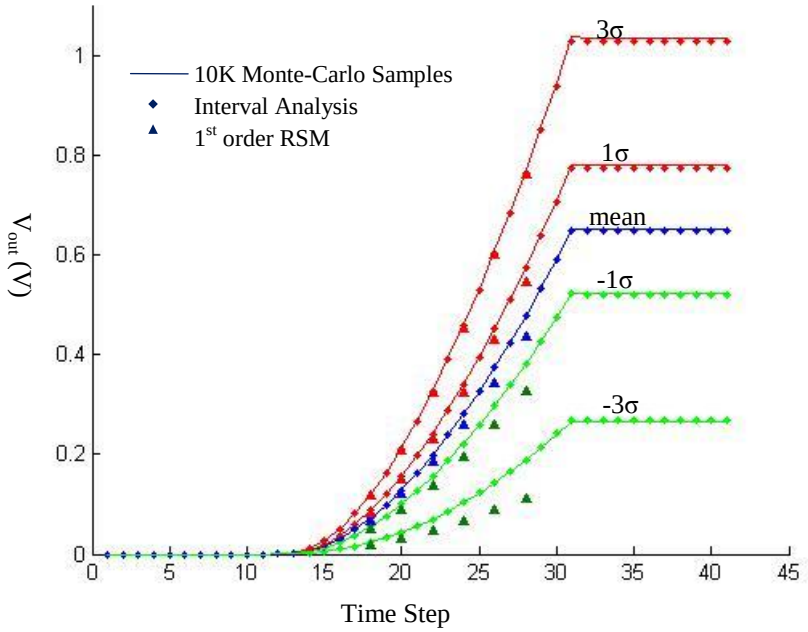

Figure 3.9: Accuracy comparison of the output waveform distributions for transistor circuit in Figure 3.7 obtained from (a) 10000 Monte-Carlo analysis; (b) first order response surface modeling with central-composite design, and (c) the interval-value based analysis.

3.6 Runtime and Scalability

3.6.1 Runtime estimation

In this section, an analytical estimation of the runtime of the interval-value based engine is carried out. Given any circuit and process variability models, the interval-value based circuit analysis requires only one circuit simulation. Therefore, the runtime penalty for an interval-value based simulation, as compared to a deterministic, real valued simulation, only comes from the runtime penalty involved in each interval operation. In this analysis we assume that the *average* number of noise variables per interval operation in the circuit simulation is *n*, and that the *average* number of shared noise variables between any two interval quantities involved in an interval operation is *m*. Table 3.3 lists the runtime penalty for the five interval operations: addition, subtraction, multiplication, inversion and division. The runtime penalty is calculated as the number of FLOPS required for interval operations divided by that required in real number operations.

Table 3.3: Runtime penalty for interval valued summation, subtraction, multiplication, inversion and division.

Operation	Runtime Penalty (FLOPS in Interval calculation / FLOPS

	in deterministic calculation)
Summation/Subtraction	*1+m*
Multiplication	*2n+2m+3*
Inversion	*3n+5*
Division	*2.5n+m+4*

The overall runtime penalty of the simulation engine is determined by the occurrence probability of summations, subtractions, multiplications and divisions. This probability depends on many factors including the circuit topology, the input signals, and the type of circuit analysis (e.g., transient analysis, DC analysis, etc). For the purpose of estimation, we will consider an equal probability of occurrence among the four operations: addition, subtraction, multiplication and division (inversion is not considered separately, but counted as part of the division operation). By adding up the corresponding runtime penalties in Table 3.3 and dividing the sum by four, the overall runtime penalty for the interval-value based circuit simulation is found to be $1.125n+1.25m+2.25$.

Thus the runtime of the interval-based circuit simulation of a circuit with *c* circuit elements, is given by $O(n+m)O(c^3)$, where the second term is the runtime for a single real-valued circuit simulation.

3.6.2 **Scalability**

In the interval-valued circuit simulation, the size of the problem depends on (a) the number of process parameters that needs to be captured statistically; (b) the average number of noise variables for each statistical process parameter[1] . We will look at how the runtime scales with respect to each of the three factors listed above.

Figure 3.10 plots the runtime and the normalized runtime of the interval-valued simulation as a function of the number of statistical process variables modeled. The test circuit used is the transistor circuit in section 3.5. For each run, an increasing number of transistor parameters are modeled by interval-values. For each parameter, only the global and local variations are captured. For comparison purposes, the runtime of 1,000 and 5,000-sample of Monte-Carlo analysis, and 1st order response surface modeling are also plotted. The Monte-Carlo sample sizes used in this comparison are the typical sample size used when simulating circuits of this scale. For larger sample sizes, the Monte-Carlo estimates do not differ much from the estimated obtained using these smaller sample sizes, i.e., the simulation has converged. The normalized runtime is defined as the runtime of the interval analysis divided by one single run of a deterministic, real-valued circuit simulation on the same test circuit. As expected, the runtime increases linearly as the number of interval-valued process variables.

Figure 3.11 shows the runtime as a function of the variability model complexity, particularly, the structure of the spatial correlation matrix. Here, a 17 stage RC ladder is

[1] This corresponds to the complexity of the variability model. Multiple noise variables are typically used to capture correlations or various hierarchical levels of variability such as within die, within wafer, etc.) and (c) the size of the circuit (i.e., the number of nodes and circuit elements

used and three types of variability model are considered. In the first model, the parameters, i.e., the resistances value, are considered to be totally uncorrelated. Thus, only one noise variable per parameter is required. In the second model, the parameters are considered to be fully correlated with a correlation length larger than the chip size, i.e., every entry in the spatial correlation matrix is non-zero. In this case, an average of $n/2$ noise variables per parameter is required. Here n is the total number of parameters. In the last model, the parameters are considered correlated but with a spatial correlation length equal to half of the chip size. The average number of noise variables per parameter is $n/4$, with n the total number of parameters. In Figure 3.11, the runtime is normalized to a single deterministic circuit simulation.

Figure 3.12 shows the runtime dependence of the interval-valued simulation on circuit size. The test circuit used is an RC ladder with an increasing number of circuit elements. The resistances are subjected to global and local variations only. For reference, the runtime of 500 and 1,000 runs of Monte-Carlo simulations are also plotted as a function of the circuit size in Figure 3.12. In the case of RC ladder example, the runtime of the interval-valued analysis becomes worse than 500 runs of Monte-Carlo simulations when the circuit element sizes becomes larger than 23; and by extrapolating the curves, the runtime of the interval-valued simulation becomes worse than 1,000 runs of Monte-Carlo simulation when the number of elements in the circuit reaches 150.

Figure 3.13 shows the dependence of the runtime for interval analysis on circuit size for transistor circuits. The circuits being analyzed are: a) inverter with one pull-up and one pull-down transistor; b) inverter with 2 pull-up and pull-down transistors; c) inverter with 3 pull-up and pull-down transistors; d) 2-input NAND gate; e) 3-input NAND gate and f) 4-input NAND gate. The runtime is plotted in Figure 3.13 as a function of number of transistors in the circuit for each of the inverter and NAND topology. For comparison, the runtime of the 500 and 1,000 runs of Monte-Carlo analysis for the corresponding topology is also plotted. By normalizing the runtime of interval analysis to a single run of deterministic circuit simulation for that circuit, a linear relationship is found between the normalized runtime and the number of transistors in the circuit for both inverter and NAND topology. Note that the results presented here come from a rather straightforward, un-optimized implementation of the interval operations. We expect that further runtime improvements are possible. For example, one can develop a scheme where only the number of quantities represented as intervals is minimized, or where aggressive error variable pruning would result in sparse matrices that could be further exploited for computational efficiency.

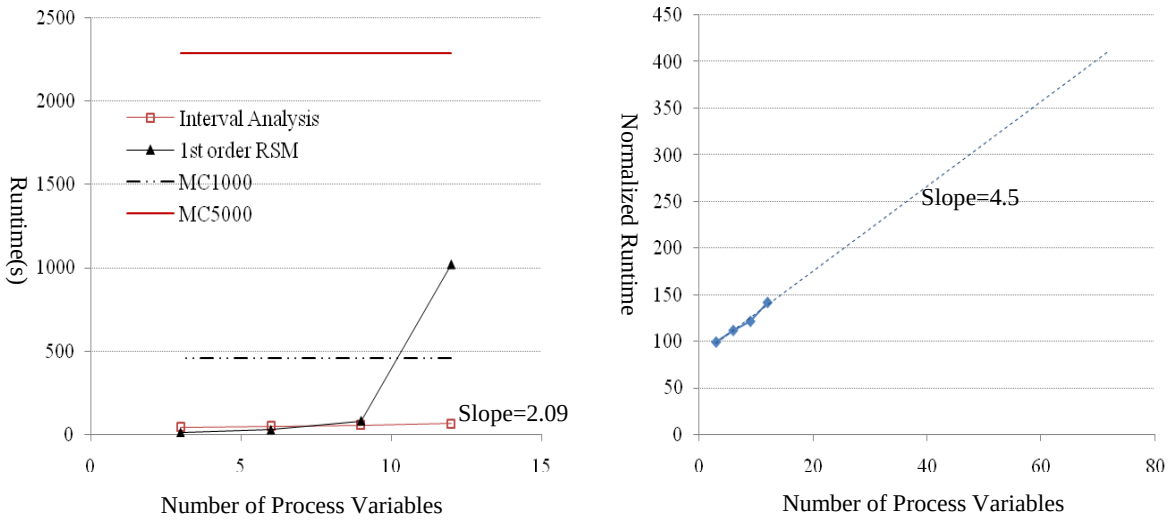

Figure 3.10: Runtime and the normalized runtime of the interval-valued simulation as a function of the number of statistical process variables modeled. For comparison, the runtimes of 1,000 and 5,000-sample of Monte-Carlo analysis, and 1st order response surface modeling is also plotted.

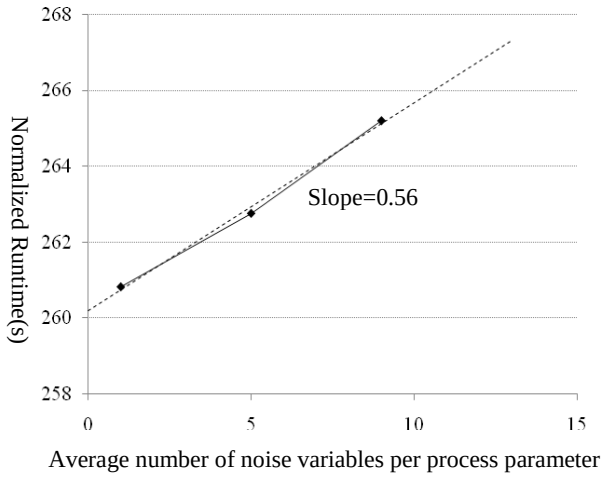

Figure 3.11: Normalized runtime of the interval-valued simulation as a function of the process variability model complexity. The test circuit used is the 17 stage RC ladder.

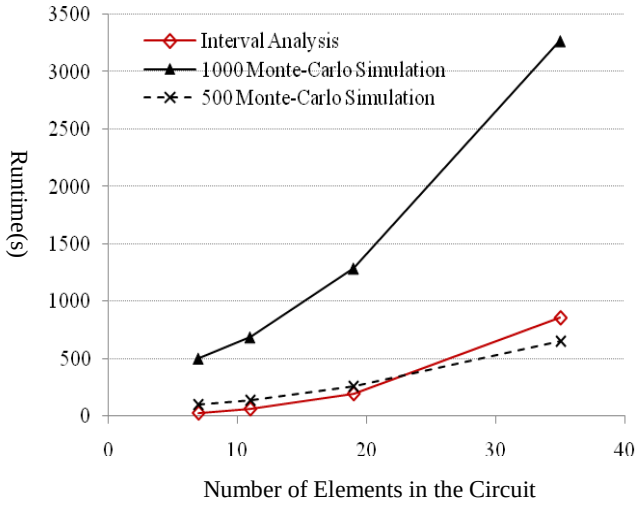

Figure 3.12: Runtime of the interval-valued simulation as a function of the circuit sizes. The test circuit used is the RC ladder.

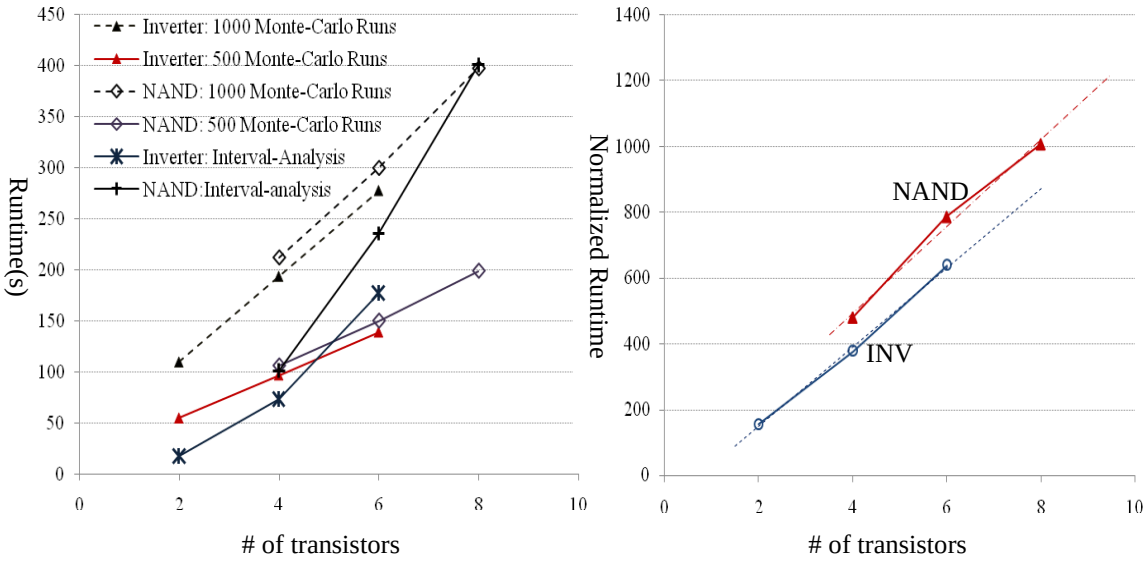

Figure 3.13: Absolute and normalized runtime of interval-valued simulation as a function of the number of transistors in circuit for inverter and NAND topologies.

3.7 Summary

An interval-value based transistor-level circuit simulation method is presented for efficient statistical circuit simulation and statistical performance estimation. This engine

can facilitate circuit design optimizations under spatial, random and systematic variations. In the proposed algorithm, a moment preserving interval-based arithmetic is used to handle operations between correlated (either spatially or not) device, process and design parameters. Excellent accuracy and reasonable runtimes are observed in simple RC and transistor circuits. It is shown analytically that the runtime of the interval valued circuit simulator is on the order of $O(n+m)O(c^3)$ where n is the average number of noise variables per interval operation, m is the average number of overlapping noise variables in the simulator and c is the size of the circuit. The runtime scales linearly in the number of process variables to be modeled statistically and in the complexity of the variability model used.

References for Chapter 3

[3.1]. L. H. de Figueiredo and J. Stolfi, "Affine arithmetic: concepts and applications." *Numerical Algorithms* **37** (1–4), 147–158. 2004.

[3.2]. Ma, J. D. and Rutenbar, R. A. 2005. "Fast interval-valued statistical interconnect modeling and reduction." In Proceedings of the 2005 international Symposium on Physical Design (San Francisco, California, USA, April 03 - 06, 2005). ISPD '05.

[3.3]. Kun Qian, Costas J. Spanos, "A Comprehensive Model of Process Variability for Statistical Timing Optimization", in Proceedings of SPIE, San Jose, 2008.

[3.4]. B. P. Harish, N. B., and Mahesh B. Patil, "On a Generalized Framework for Modeling the Effects of Process Variations on Circuit Delay Performance Using Response Surface Methodology", IEEE Trans on CAD of IC. 26(3): 606-614 (2007).

[3.5]. Nagel, L. W. (1975). SPICE 2: A Computer Program to Simulate Semiconductor Circuits. Dept. of Electrical Engineering and Computer Sciences. Berkeley, University of California, Berkeley. Ph. D Dissertation.

[3.6]. Li X. and Pileggi L., "Statistical performance modeling and optimization", Foundations and trends in EDA. Vol 1, No. 4, 331-490(2006).

[3.7]. Melvin Dale Springer (1979). The Algebra of Random Variables. Wiley. ISBN 0-471-01406-0.

[3.8]. Rohatgi, V.K., 1976. An Introduction to Probability Theory Mathematical Statistics. Wiley, New York.

[3.9]. Johnson, Noeman L.; Kotz, Samuel; Balakrishnan, N. (1995). Continuous Univariate Distributions Volume 2, Second edition. Wiley. p. 306. ISBN 0-471-58494-0.

[3.10]. Springer, MD and Thompson, WE (1970). "The distribution of products of beta, gamma and Gaussian random variables". SIAM Journal on Applied Mathematics 18 (4): 721–737.

Chapter 4 Interval Analysis for Non-Gaussian Uncertainty Propagation

4.1 Introduction

The previous chapter discussed an interval based statistical circuit simulation algorithm for Gaussian distribution. In this chapter, a similar algorithm is developed for the case when the process/device parameters follow non-Gaussian distributions.

The combination of random and systematic variability in state of the art IC technologies [4.9][4.10] often results in non-Gaussian distributions of key performance parameters. For example, wafer-level systematic variations cause the chip means to shift depending on the chip location on the wafer. If the overall distribution for all the chips on a wafer is examined, the probability density function (*pdf*) is the aggregate of all the individual chip distributions. Assuming that each chip distribution has a Gaussian *pdf* with means determined by the wafer-level systematic variations, then the overall distribution is a sum of these Gaussian *pdf*s and it can be described as a *mixture* of Gaussian distributions.

Non-Gaussian distributions of the process/device parameters can also be found in the case when process information is not completely known. This usually occurs in the early stage of a process development where only the process window, i.e., the lower and upper process limits on the parameter values, is known. In this case, the parameter distribution density function can fall anywhere inside the process window, as demonstrated in Figure 4.1, resulting an overall non-Gaussian distribution.

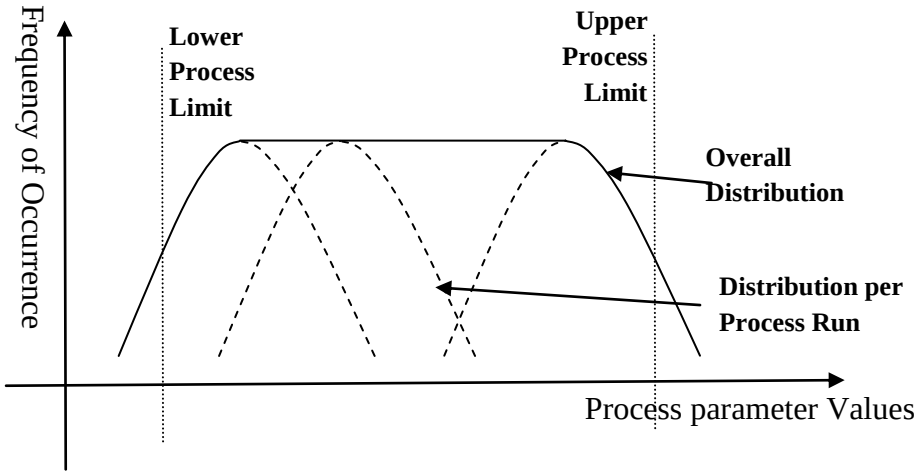

Figure 4.1: Demonstration of the parameter distribution as a result of knowing only the upper and lower process limit of the parameter.

In this chapter, Mixture of Gaussian (MOG) distributions are used to approximate all non-Gaussian distributions. Similar to interval propagation for Gaussian distributions as discussed in Chapter 3, an interval propagation scheme for mixture of Gaussians is developed to estimate the transistor-level circuit performance distribution without costly Monte-Carlo simulations. The rest of the chapter is organized as follows: section 4.2 reviews the mathematical tools used to capture a mixture of Gaussian distribution using interval representations; section 4.3 details the algorithms used to estimate non-Gaussian performance of interest using interval representations and propagation; section 4.4 presents some circuit examples to demonstrate the proposed algorithm; section 4.5 concludes the chapter.

4.2 Interval Representation for Mixture Of Gaussian(MOG) Distributions

In this section, an overview of the mixture of Gaussian distribution and its equivalent interval representation is provided.

4.2.1 Mixture of Gaussian(MOG) Distribution

The probability density function (*pdf*) of a mixture of Gaussian distribution is given by [4.13],

$$pdf(\boldsymbol{x}) = \sum_{i=1}^{C} w_i N(\mu_i, \Sigma_i) \tag{4.1}$$

where $N(\mu_i, \Sigma_i)$ for $i=1,\ldots,C$ are Gaussian probability density functions with mean μ_i and covariance matrix Σ_i; w_i are real numbers between 0 and 1 and satisfying the condition $\sum_{i=1}^{C} w_i = 1$.

In equation (4.1), the distribution $N(\mu_i, \Sigma_i)$ are referred to as the Gaussian component of the mixture *pdf*, while w_i is referred to as the mixing probability of the i^{th} Gaussian component. The value of w_i corresponds to the probability that a random sample is drawn from the i^{th} Gaussian component. c is the total number of Gaussian components in a mixture *pdf*. The probability density function (*pdf*) for a 2-component mixture of Gaussian distribution with equal mixing probability is depicted in Figure 4.2.

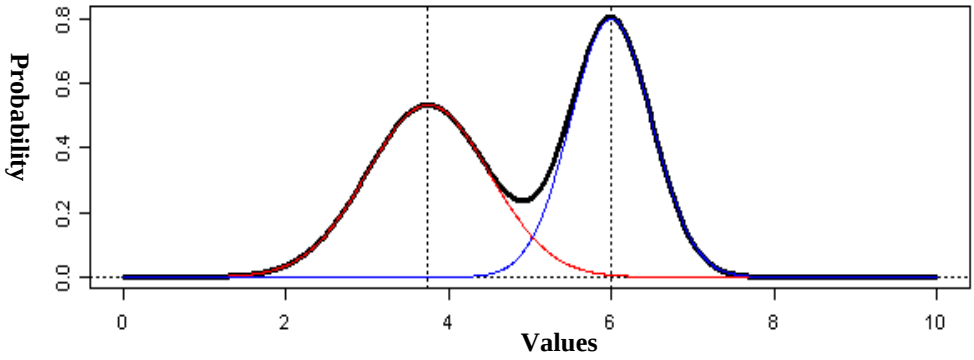

Figure 4.2: Probability density function of a mixture of Gaussian distribution with two components and equal mixing probability.

4.2.2 **Interval MOG Representation**

The interval representation [4.1] for MOG distributions is shown in equation (4.2).

$$\boldsymbol{x} = \tilde{\boldsymbol{w}}^T \begin{bmatrix} \sum_i a_{1i}\boldsymbol{\varepsilon}_i + x_{10} \\ \vdots \\ \sum_i a_{Ci}\boldsymbol{\varepsilon}_i + x_{C0} \end{bmatrix} \tag{4.2}$$

where $\tilde{\boldsymbol{w}}$ is a c-dimensional random vector that takes on the value $[1,0,\ldots,0]^T$ with probability w_1, $[0,1,0,\ldots,0]^T$ with probability w_2, and up to $[0,\ldots,0,1]^T$ with probability w_C.

The terms, $\sum_i a_{ki}\boldsymbol{\varepsilon}_i + x_{k0}$ for $k=1,\ldots,C$, in equation (4.2) are the interval representations for Gaussian distributions, as those discussed in Chapter 3. There is a total of c such Gaussian interval representations the captures the c Gaussian components in a mixture. The random vector $\tilde{\boldsymbol{w}}$ acts as the mixing variable such that with a probability w_1, the samples will be drawn from the first Gaussian components, probability w_2 of drawing from the second, and so on.

Equation (4.2) can be converted into matrix-vector forms as,

$$\boldsymbol{x} = \widetilde{\boldsymbol{w}}^T \left(\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{C1} & \cdots & a_{Cn} \end{bmatrix} \begin{bmatrix} \boldsymbol{\varepsilon}_1 \\ \vdots \\ \boldsymbol{\varepsilon}_n \end{bmatrix} + \begin{bmatrix} x_{10} \\ \vdots \\ x_{C0} \end{bmatrix} \right) = \widetilde{\boldsymbol{w}}^T (A\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_0) \tag{4.3}$$

where $A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{C1} & \cdots & a_{Cn} \end{bmatrix}$, $\tilde{\boldsymbol{\varepsilon}} = \begin{bmatrix} \boldsymbol{\varepsilon}_1 \\ \vdots \\ \boldsymbol{\varepsilon}_n \end{bmatrix}$ and $\tilde{x}_0 = \begin{bmatrix} x_{10} \\ \vdots \\ x_{C0} \end{bmatrix}$. Note that the random vectors $\widetilde{\boldsymbol{w}}$ and $\tilde{\boldsymbol{\varepsilon}}$ are independent of each other.

For the rest of the chapter, variables with a bold face are used to denote random variables, and variables with a tilde on the top are used to denote vectors.

The interval representation for Gaussian distribution, can be viewed as a special case of equation (4.2) when the number of Gaussian components *c*=1.

Thus, the l^{th} statistical moment of the MOG as described by equation (4.3) can be calculated as

$$\begin{aligned} \mathrm{E}(\boldsymbol{x}^l) &= \mathrm{E}\big(\widetilde{\boldsymbol{w}}^T (A\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_0)\big)^l \\ &= w_1 \mathrm{E}\big([1,0,\ldots,0](A\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_0)\big)^l + w_2 \mathrm{E}\big([0,1,\ldots,0](A\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_0)\big)^l \\ &\qquad + \ldots + w_C \mathrm{E}\big([0,\ldots 0,1](A\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_0)\big)^l \\ &= [w_1 \cdots w_c] \begin{bmatrix} \mathrm{E}(a_1\varepsilon + x_{10})^l \\ \vdots \\ \mathrm{E}(a_C\varepsilon + x_{C0})^l \end{bmatrix} \end{aligned} \tag{4.4}$$

where $a_1,\ldots\ a_C$ are rows from the matrix A in equation (4.3). In other words, the l^{th} moment of a mixture of Gaussian interval is the sum of the l^{th} moments of all the interval Gaussian components weighted by the mixing probability.

If $\boldsymbol{x}$ and $\boldsymbol{y}$ are two distinct mixture of Gaussian intervals in the form of equation (4.3), i.e.,

$$\begin{aligned} \boldsymbol{x} &= \widetilde{\boldsymbol{w}}_{\boldsymbol{x}}{}^T (A_x\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{x0}) \\ \boldsymbol{y} &= \widetilde{\boldsymbol{w}}_{\boldsymbol{y}}{}^T \big(A_y\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{y0}\big) \end{aligned} \tag{4.5}$$

Then the covariance between the two can be calculated as

$$\begin{aligned} cov(\boldsymbol{x}, \boldsymbol{y}) &= \mathrm{E}\big(\boldsymbol{x} - \mathrm{E}(\boldsymbol{x})\big)\big(\boldsymbol{y} - \mathrm{E}(\boldsymbol{y})\big) \\ &= \mathrm{E}(\widetilde{\boldsymbol{w}}_{\boldsymbol{x}}^T A_x \tilde{\boldsymbol{\varepsilon}}\tilde{\boldsymbol{\varepsilon}}^{\boldsymbol{T}} A_y^T \widetilde{\boldsymbol{w}}_y) \\ &= \mathrm{E}(\widetilde{\boldsymbol{w}}_x^T) A_x A_y^T \mathrm{E}\big(\widetilde{\boldsymbol{w}}_y\big) \end{aligned}$$

(4.6)

where $\mathrm{E}(\tilde{\boldsymbol{w}}_x^T)$ and $\mathrm{E}(\tilde{\boldsymbol{w}}_y^T)$ are the expectations and equals to the vector of mixing probabilities, i.e., $[w_{1x},\ldots,w_{cx}]^{\mathrm{T}}$ for $\boldsymbol{x}$ and $[w_{1y},\ldots,w_{cy}]^{\mathrm{T}}$ for $\boldsymbol{y}$.

4.2.3 From MOG distributions to Interval Representation

Assume that an *n*-dimensional multivariate MOG distribution with *c* Gaussian components is given in the form of equation (4.1) where the mean μ_i is a *n*-dimensional vector and covariance Σ_i is *n*×*n* matrix for each Gaussian component *i*. In Chapter 3, a procedure has been developed to convert *n* correlated Gaussian random variables (that have the joint probability density function given by a *n*-dimensional multivariate Gaussian *pdf*) into *n* interval quantities. The same procedure is used here to convert each Gaussian component in equation (4.1) into a set of *n* interval quantities, resulting into a total of *c*×*n* interval quantities for *c* Gaussian components. These interval quantities are denoted by $\boldsymbol{a}_{pq}$, for *p=1,…,c* and *q=1,…,n.*

To construct the interval MOG representation, one interval MOG quantity is assigned to represent one dimension in the original distribution. The set of *n* interval MOG quantities for an *n*-dimensional MOG distribution is given by,

$$\boldsymbol{x}_j = \tilde{\boldsymbol{w}}_j{}^T\left(A_j\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{0j}\right),\ \ j = 1,\ldots,n \tag{4.7}$$

where $\tilde{\boldsymbol{w}}_j$ and $\tilde{x}_{0j}$ have the dimension *c*×1. In addition, the vector $\left(A_j\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{0j}\right)$ is assigned to the interval quantities obtained from each individual Gaussian component,

$$\left(A_j\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{0j}\right) = \begin{bmatrix}\boldsymbol{a}_{1j}\\ \boldsymbol{a}_{2j}\\ \vdots\\ \boldsymbol{a}_{cj}\end{bmatrix},\ \ j = 1,\ldots,n \tag{4.8}$$

The random vector $\tilde{\boldsymbol{w}}_j$ is the same for *j*=1,…, n, and has expectation equals to $[w_1,\ldots,w_c]^T$, the set of mixing probabilities from the original MOG distributions.

4.3 Interval MOG propagation in Statistical Circuit Simulation

In this section, a transistor-level circuit simulation algorithm is developed for process/device variations that follow non-Gaussian distributions. In this work, it is assumed that non-Gaussian distributions of the process/device parameters can be approximated by a mixture of Gaussian distributions. The most commonly used algorithm to find an MOG approximation to an arbitrary distribution is the Expectation-Maximization algorithm [4.4]. In this algorithm, the most likely mixture of Gaussian distribution is fitted to match the true distribution. The approximation accuracy is determined by the number of Gaussian components used. For a mixture of Gaussian

distribution containing an arbitrarily large number of components, any distribution can be approximated [4.13].

The variations in process/device parameters are first approximated by an MOG and then converted into interval MOG representations, using the procedure outlined in section 4.2.3. This information is passed into a circuit simulation engine that handles interval MOG quantities. The simulation engine outputs circuit performances represented in interval MOG forms which can be used to extract any distribution statistics of interest.

In the following discussions, the circuit performances of interest are divided into non-transient and transient performances. Non-transient performances are those specified by a single value, such as the circuit propagation delay or total power dissipation; whereas transient performances are a set of values obtained for each time interval during a circuit transient transition. Example of transient performance is the switching behavior of an inverter.

4.3.1 **Problem Formulation**

The circuit performance of interest is denoted as $\boldsymbol{z}$, and the set of all random process/device parameters is denoted using a random vector $\tilde{\boldsymbol{p}}$.

For non-transient performances, the performance $\boldsymbol{z}$ is related to the process/device parameters by a nonlinear function *f*, i.e.,

$$\boldsymbol{z} = f(\tilde{\boldsymbol{p}}, \tilde{v}_0) \tag{4.9}$$

where the vector $\tilde{v}_0$ collects all the deterministic parameters in the circuit. Examples of deterministic parameters include the initial states of the circuit nodal voltages, and the process/device parameters that are not subjected to process variations. The functional form of *f* is related to the performance of interest and the circuit topology. A method to approximate *f* will be discussed later in the section.

The random process/device parameters are given in interval MOG form,

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_k \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{w}}_{p1}^T (A_{p1}\tilde{\boldsymbol{\varepsilon}} + \tilde{p}_{10}) \\ \vdots \\ \tilde{\boldsymbol{w}}_{pk}^T (A_{pk}\tilde{\boldsymbol{\varepsilon}} + \tilde{p}_{k0}) \end{bmatrix} \tag{4.10}$$

Equation (4.10) is extracted from the parameter distribution obtained from silicon measurement data such as test structure characterization, but also from device level simulations and other process and lithography simulations.

In addition, the non-transient performance $\boldsymbol{z}$ is assumed to also follow a MOG distribution represented by an interval MOG, with the same set of the noise variables as those present in the interval MOG representations for parameters $\tilde{\boldsymbol{p}}$.

$$\boldsymbol{z} = \tilde{\boldsymbol{w}}_z^T (A_z\tilde{\boldsymbol{\varepsilon}} + \tilde{z}_0) \tag{4.11}$$

The goal is to find the values of $\tilde{\boldsymbol{w}}_z$, A_z and $\tilde{z}_o$ that define the interval MOG representation of $\boldsymbol{z}$ in equation (4.11).

For transient performances, i.e., performances that are a function of time, the problem then is to find, at each time step *t*,

$$\boldsymbol{z}_t = g_t(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{x}}_t, \tilde{v}_0) \tag{4.12}$$

where $\boldsymbol{z}_t$ is the transient performance of interest; $\tilde{\boldsymbol{x}}_t$ and $\tilde{\boldsymbol{x}}_{t-1}$ are random vectors capturing all the circuit nodal voltages/branch currents at time *t* and *t*-1, respectively. Vector $\tilde{v}_0$ captures all the non-statistical parameters just as in the non-transient analysis. The function g_t is specific to the performance measure of interest and assumed to be given. For example, if one is interested in the output voltages at circuit nodes *i*, then $g_t = \tilde{\boldsymbol{x}}_t(i)$.

The quantities $\tilde{\boldsymbol{x}}_t$ and $\tilde{\boldsymbol{x}}_{t-1}$ are related in a transient circuit simulation. Recall from Chapter 3, that $\tilde{\boldsymbol{x}}_t$ is calculated by solving a linear system of equations constructed by applying Kirchhoff's current and voltage laws on a linearized circuit at operating conditions given by $\tilde{\boldsymbol{x}}_t$ and an initial guess of $\tilde{\boldsymbol{x}}_{t-1}$. At the end of each time step, the following equation holds

$$\tilde{\boldsymbol{x}}_t = M(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t-1})^{-1} I(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t-1}) = B_t(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t-1}) \tag{4.13}$$

where *M* is the matrix containing all the linearized circuit element values, and *I* is a vector containing the input voltage and current source values. Since both *M* and *I* are a function of $\tilde{\boldsymbol{x}}_t$, $\tilde{\boldsymbol{x}}_{t-1}$ and the set of process parameters $\tilde{\boldsymbol{p}}$, they can be combined to a single function B_t.

In the same fashion as in the non-transient problem, the random process/device parameters are given in interval MOG forms,

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_k \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{w}}_{p1}^T (A_{p1}\tilde{\boldsymbol{\varepsilon}} + \tilde{p}_{10}) \\ \vdots \\ \tilde{\boldsymbol{w}}_{pk}^T (A_{pk}\tilde{\boldsymbol{\varepsilon}} + \tilde{p}_{k0}) \end{bmatrix} \tag{4.14}$$

In addition, at time step *t*, the distribution of $\tilde{\boldsymbol{x}}_{t-1}$ is also known since it is obtained from the analysis at the previous time step. The goal is to find the distribution for $\tilde{\boldsymbol{x}}_t$ and consequently the distribution for $\boldsymbol{z}_t$. As in the non-transient case, the distribution of $\tilde{\boldsymbol{x}}_t$ and $\boldsymbol{z}_t$ is assumed to be captured by an interval MOG, i.e.,

$$\tilde{\boldsymbol{x}}_t = \begin{bmatrix} \boldsymbol{x}_{1,t} \\ \vdots \\ \boldsymbol{x}_{m,t} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{w}}_{x_1,t} (A_{x_1,t}\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{1,t,0}) \\ \vdots \\ \tilde{\boldsymbol{w}}_{x_m,t} (A_{x_m,t}\tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{m,t,0}) \end{bmatrix} \tag{4.15}$$

and

$$\boldsymbol{z}_t = \widetilde{\boldsymbol{w}}_{z_t}^T (A_{z_t} \tilde{\boldsymbol{\varepsilon}} + \tilde{z}_{0_t}) \tag{4.16}$$

where the values for $\widetilde{\boldsymbol{w}}_{z_t}$, A_{z_t} and $\tilde{z}_{0_t}$ are to be determined.

In addition, note that $\widetilde{\boldsymbol{x}}_{t-1}$ should have the same form of $\widetilde{\boldsymbol{x}}_t$, i.e., an interval MOG representation,

$$\widetilde{\boldsymbol{x}}_{t-1} = \begin{bmatrix} \boldsymbol{x}_{1,t-1} \\ \vdots \\ \boldsymbol{x}_{m,t-1} \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{w}}_{x_1,t-1} (A_{x_1,t-1} \tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{1,t-1,0}) \\ \vdots \\ \widetilde{\boldsymbol{w}}_{x_m,t-1} (A_{x_m,t-1} \tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{m,t-1,0}) \end{bmatrix} \tag{4.17}$$

4.3.2 **Solution for Non-Transient Performances**

Recall from the previous section that the problem is to find the interval MOG representation of $\boldsymbol{z}$, with

$$\boldsymbol{z} = f(\widetilde{\boldsymbol{p}}, \tilde{v}_0) \tag{4.18}$$

$$\boldsymbol{z} = \widetilde{\boldsymbol{w}}_z^T (A_z \tilde{\boldsymbol{\varepsilon}} + \tilde{z}_0) \tag{4.19}$$

and the distribution of the device/process parameters given in interval MOG form

$$\widetilde{\boldsymbol{p}} = \begin{bmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_k \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{w}}_{p1}^T (A_{p1} \tilde{\boldsymbol{\varepsilon}} + \tilde{p}_{10}) \\ \vdots \\ \widetilde{\boldsymbol{w}}_{pk}^T (A_{pk} \tilde{\boldsymbol{\varepsilon}} + \tilde{p}_{k0}) \end{bmatrix} \tag{4.20}$$

To obtain a solution for $\boldsymbol{z}$, a 2nd order Taylor series expansion is first performed on the function *f*, i.e.,

$$f(\widetilde{\boldsymbol{p}}, \tilde{v}_0) \cong f(\mathrm{E}(\widetilde{\boldsymbol{p}}), \tilde{v}_0) + \nabla_f^T \big(\widetilde{\boldsymbol{p}} - \mathrm{E}(\widetilde{\boldsymbol{p}})\big) + \big(\widetilde{\boldsymbol{p}} - \mathrm{E}(\widetilde{\boldsymbol{p}})\big)^T H_f \big(\widetilde{\boldsymbol{p}} - \mathrm{E}(\widetilde{\boldsymbol{p}})\big) \tag{4.21}$$

where $\mathrm{E}(\cdot)$ stands for the expectation. ∇_f^T and H_f are the gradient and Hessian, respectively, of the function *f* evaluated at the nominal parameter values $\mathrm{E}(\widetilde{\boldsymbol{p}})$.

$$\nabla_f = \begin{bmatrix} \left.\frac{\partial f}{\partial \boldsymbol{p}_1}\right|_{\mathrm{E}(\tilde{\boldsymbol{p}})} \\ \vdots \\ \left.\frac{\partial f}{\partial \boldsymbol{p}_k}\right|_{\mathrm{E}(\tilde{\boldsymbol{p}})} \end{bmatrix} \tag{4.22}$$

$$H_f = \begin{bmatrix} \left.\frac{\partial^2 f}{\partial \boldsymbol{p}_1^2}\right|_{\mathrm{E}(\tilde{\boldsymbol{p}})} & \cdots & \left.\frac{\partial^2 f}{\partial \boldsymbol{p}_1 \boldsymbol{p}_k}\right|_{\mathrm{E}(\tilde{\boldsymbol{p}})} \\ \vdots & \ddots & \vdots \\ \left.\frac{\partial^2 f}{\partial \boldsymbol{p}_1 \boldsymbol{p}_k}\right|_{\mathrm{E}(\tilde{\boldsymbol{p}})} & \cdots & \left.\frac{\partial^2 f}{\partial \boldsymbol{p}_k^2}\right|_{\mathrm{E}(\tilde{\boldsymbol{p}})} \end{bmatrix} \tag{4.23}$$

There are many different methods to obtain the gradient and the hessian in equation (4.22) and (4.23). One method is the direct perturbation analysis or sensitivity analysis. In this approach, the performance of interest, $\boldsymbol{z} = f(\tilde{\boldsymbol{p}}, \tilde{v}_0)$, is simulated using the nominal parameter values $\mathrm{E}(\boldsymbol{p}_i)$ and using perturbed parameter values, e.g., $\mathrm{E}(\boldsymbol{p}_i) \pm h$, where h is a very small quantity. The gradient and Hessian matrix are then calculated numerically from these performance values, i.e., $f(\mathrm{E}(\tilde{\boldsymbol{p}}), \tilde{v}_0)$ and $f(\mathrm{E}(\tilde{\boldsymbol{p}}) \pm h, \tilde{v}_0)$, by numerical differentiation. Examples of numerical differentiation formulas are given in equation (4.24), (4.25) and (4.26) for the first and second order derivations of the function f with respect to the parameters.

$$\frac{\partial f}{\partial \boldsymbol{p}_i} \cong \frac{f(\mathrm{E}(\boldsymbol{p}_i) + h) - f(\mathrm{E}(\boldsymbol{p}_i) - h)}{2h} \tag{4.24}$$

$$\begin{aligned} \frac{\partial^2 f}{\partial \boldsymbol{p}_i \partial \boldsymbol{p}_j} \cong \frac{1}{4h^2} \big[& f\big(\mathrm{E}(\boldsymbol{p}_i) + h, \mathrm{E}\big(\boldsymbol{p}_j\big) + h\big) - f\big(\mathrm{E}(\boldsymbol{p}_i) - h, \mathrm{E}\big(\boldsymbol{p}_j\big) + h\big) \\ & - f\big(\mathrm{E}(\boldsymbol{p}_i) + h, \mathrm{E}\big(\boldsymbol{p}_j\big) - h\big) + f\big(\mathrm{E}(\boldsymbol{p}_i) - h, \mathrm{E}\big(\boldsymbol{p}_j\big) - h\big) \big], \qquad i \neq j \end{aligned} \tag{4.25}$$

$$\frac{\partial^2 f}{\partial \boldsymbol{p}_i^2} \cong \frac{1}{h^2} \big[f(\mathrm{E}(\boldsymbol{p}_i) + h) - 2f\big(\mathrm{E}(\boldsymbol{p}_i)\big) + f(\mathrm{E}(\boldsymbol{p}_i) - h) \big] \tag{4.26}$$

The error associated with these approximations is $\mathrm{O}(h^2)$ for both first and second derivatives. For a total of k parameters, $2k$ circuit simulations are required for gradient evaluations, i.e., 2 circuit simulations per parameter. To obtain the diagonal entries of the hessian matrix, only one additional circuit simulation at the nominal value is required. Each off-diagonal entry requires 4 additional circuit simulations. Since there are $k(k\text{-}1)/2$

distinct off-diagonal entries, the number of circuit simulations is $2k(k\text{-}1)$. The total number of circuit simulations required to obtain both the gradient and the hessian matrix is therefore $2k^2+1$.

Another method to calculate the gradient and Hessian matrix in equation (4.22) and (4.23) is to use an adjoint network. The details of how an adjoint network works can be used for gradient calculation are provided in Chapter 2. In gradient evaluations, the two circuit simulations are required, one for the original circuit, and one for the adjoint network circuit, regardless of the number of parameters involved. For Hessian evaluations, an adjoint of the adjoint network used for gradient evaluation is constructed, which results in a total of *2k* circuit evaluations for *k* parameters. The adjoint network approach for first order and second order derivatives can be also found in [4.5], [4.6] and [4.7].

Further reduction of the runtime for gradient and Hessian evaluation is possible by reducing the value of *k*. In the previous analysis, all the derivatives are calculated with respect to the parameters in $\tilde{\boldsymbol{p}}$. However, the parameters are represented by interval MOG representations using the noise variables $\tilde{\boldsymbol{\varepsilon}}$. Therefore, the gradient and Hessian with respect to the parameters can be obtained by calculating the derivates with respect to the noise variables instead, as shown in equation (4.27) for the gradient estimation. The Hessian can be similarly obtained. For parameter p_i,

$$\frac{\partial f}{\partial p_i} = \sum_{j=1}^{n} \frac{\partial f}{\partial \varepsilon_j}\frac{\partial \varepsilon_j}{\partial p_i} = \sum_{j=1}^{n} \frac{\partial f}{\partial \varepsilon_j}\frac{1}{\mathrm{E}\left[\tilde{\boldsymbol{w}}_{p_i}^{T} Col_j\left(A_{p_i}\right)\right]} \tag{4.27}$$

where *n* is the total number of noise variables, and $Col_j(\cdot)$ stands for the j^{th} column of a matrix.

This allows for a significant runtime reduction when the number of noise variables are fewer than the actual number of parameters, which is the case when the parameters are highly correlated. Strong correlations are often observed when the circuit is tightly packed with elements situated close to each other. Otherwise, such as in the case when all parameters are independent, the calculation is performed with respect to parameters as usual.

Once the gradient and the Hessian matrix of function *f* is obtained, the next step is to determine the distribution of $\boldsymbol{z}$ from the second order Taylor series expansion of *f*. To completely specify the distribution of $\boldsymbol{z}$, the distribution for $\tilde{\boldsymbol{w}}_z$ and the values for A_z and $\tilde{z}_0$ in the interval MOG representation must be determined.

The number of Gaussian mixtures required in capturing the statistical behavior of $\boldsymbol{z}$ depends on the accuracy required for the interval MOG approximation with respect to the actual distribution.

If only one Gaussian component is used, the interval MOG approximation for $\boldsymbol{z}$ represents a Gaussian distribution and therefore can accurately capture only the first and second moments of the true distribution. If two Gaussian components are used, then the interval MOG approximation for $\boldsymbol{z}$ can accurately capture up to the fifth moment of the true distribution [4.11][4.12].

In general, for an interval MOG approximation to accurately capture the first L moments of a distribution, the minimum number of Gaussian components required is $\left\lceil \frac{L+1}{3} \right\rceil$, where $\lceil \cdot \rceil$ denotes the ceiling operation. In an interval MOG representation, each Gaussian component is completely specified by a mean and a variance, resulting in a total of 2c degree of freedom, where c is the number of Gaussian components. In addition, the mixing probabilities $\tilde{\boldsymbol{w}}$ add another c-1 degree of freedom. Therefore, the total degrees of freedom in an interval MOG representation is given by 3c-1. This number has to be greater or equal to L in order for the interval MOG representation to capture the first L moment of the true distribution. Thus, the minimum number of Gaussian components required is found by setting 3c-1 equal to L.

The user may choose the value of L to control the accuracy of the circuit performance estimation.

Once the number of Gaussian components in the interval MOG representation for z has been determined, the next step is to find the values of $\tilde{\boldsymbol{w}}_z$, $\tilde{z}_0$ and A_z. The total number of unknowns is $c(n+2)-1$ calculated by summing up the following three numbers: $c-1$ unknowns for specifying the distributions of $\tilde{\boldsymbol{w}}_z$; c unknowns for specifying the mean vector $\tilde{z}_0$; and nc unknowns for specifying the matrix A_z, where n is the number of noise variables used in the interval MOG representation for the parameters $\tilde{\boldsymbol{p}}$.

A total of number of L moment-matching equations are set up to reinforce the moment-matching requirement in the estimation of the distribution of $\boldsymbol{z}$, i.e.,

$$\mathrm{E}\left({f_{approx}}^{l} \right) = \mathrm{E}(\boldsymbol{z}^l), \qquad l = 1, \ldots, L \tag{4.28}$$

where f_{approx} is the second-order Taylor series approximation of f as shown in equation (4.21). The left and right hand side of equation (4.28) can be obtained analytically by following the calculations shown in equation (4.4) and (4.6).

In addition, the following equations ensure that the correlation between the performance $\boldsymbol{z}$ and the parameter $\tilde{\boldsymbol{p}}$ is preserved,

$$\mathrm{E}\left[\left(\frac{\partial f_{approx}}{\partial \varepsilon_i} \right)^q \right] = \mathrm{E}\left(\frac{\partial \boldsymbol{z}}{\partial \varepsilon_i} \right)^q, \quad q = 1, \ldots, Q \tag{4.29}$$

for i=1,…,n, where n is the total number of noise variables used in characterizing the parameters $\tilde{\boldsymbol{p}}$. Q is set to the floor of $\frac{c(n-1)}{n}$ where c is the number of Gaussian components in $\boldsymbol{z}$. Equations (4.28) and (4.29) together counts to a total of $c(n+2)-1$ equations involving the unknowns $\tilde{\boldsymbol{w}}_z$, A_z and $\tilde{z}_0$, with $c = \left\lceil \frac{L+1}{3} \right\rceil$. These equations can be solved for the values of the unknowns. Note that all these equations contain only polynomials of the unknowns, therefore any nonlinear solvers using algorithms such as Gauss-Newton or Levenberg-Marquardt will behave well in finding the solutions.

Authors in [4.2] and [4.3] also have discussed specific algorithms for solving systems of polynomial equations.

4.3.3 **Solution for Transient Performances**

In the case of transient performances in which the performance values are required for every time step within a period of time, the problem formulation is, as discussed previously,

$$\boldsymbol{z}_t = g_t(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_{t-1}, \tilde{\boldsymbol{x}}_t, \tilde{v}_0) \tag{4.30}$$

where $\tilde{\boldsymbol{x}}_t$ and $\tilde{\boldsymbol{x}}_{t-1}$ are the circuit nodal and branch current at time step t and t-1, respectively . $\tilde{\boldsymbol{x}}_t$ and $\tilde{\boldsymbol{x}}_{t-1}$ are related by the transient circuit simulation equations at time step t as

$$\tilde{\boldsymbol{x}}_t = M(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t-1})^{-1} I(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t-1}) = B_t(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t-1}) \tag{4.31}$$

We also require that all the quantities are represented in interval MOG form, such that,

$$\boldsymbol{z}_t = \tilde{\boldsymbol{w}}_{z_t}{}^T (A_{z_t} \tilde{\boldsymbol{\varepsilon}} + \tilde{z}_{0t}) \tag{4.32}$$

and

$$\tilde{\boldsymbol{p}} = \begin{bmatrix} \boldsymbol{p}_1 \\ \vdots \\ \boldsymbol{p}_k \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{w}}_{p1}^T (A_{p1} \tilde{\boldsymbol{\varepsilon}} + \tilde{p}_{10}) \\ \vdots \\ \tilde{\boldsymbol{w}}_{pk}^T (A_{pk} \tilde{\boldsymbol{\varepsilon}} + \tilde{p}_{k0}) \end{bmatrix} \tag{4.33}$$

$$\tilde{\boldsymbol{x}}_{t-1} = \begin{bmatrix} \boldsymbol{x}_{1,t-1} \\ \vdots \\ \boldsymbol{x}_{m,t-1} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{w}}_{x_1,t-1} (A_{x_1,t-1} \tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{1,t-1,0}) \\ \vdots \\ \tilde{\boldsymbol{w}}_{x_m,t-1} (A_{x_m,t-1} \tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{m,t-1,0}) \end{bmatrix} \tag{4.34}$$

$$\tilde{\boldsymbol{x}}_t = \begin{bmatrix} \boldsymbol{x}_{1,t} \\ \vdots \\ \boldsymbol{x}_{m,t} \end{bmatrix} = \begin{bmatrix} \tilde{\boldsymbol{w}}_{x_1,t} (A_{x_1,t} \tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{1,t,0}) \\ \vdots \\ \tilde{\boldsymbol{w}}_{x_m,t} (A_{x_m,t} \tilde{\boldsymbol{\varepsilon}} + \tilde{x}_{m,t,0}) \end{bmatrix} \tag{4.35}$$

At each time step during the circuit transient response period, the circuit nodal voltage/branch currents $\tilde{\boldsymbol{x}}_t$ are solved to obtain their interval MOG representations; this information is used to calculate the performance of interest $\boldsymbol{z}_t$ at that time step. The

algorithm then proceeds to the next time step with $\tilde{x}_t$ used as the inputs for the circuit nodal voltage/current from the previous time step.

In a single time step, there are two interval MOG estimations required: (1) estimating the interval MOG representation for the distribution of $\tilde{\boldsymbol{x}}_t$ from that of $\tilde{\boldsymbol{x}}_{t-1}$ and $\tilde{\boldsymbol{p}}$; and (2) estimating the interval MOG representation for the performance at that time step $\boldsymbol{z}_t$ from $\tilde{\boldsymbol{x}}_t$, $\tilde{\boldsymbol{x}}_{t-1}$ and $\tilde{\boldsymbol{p}}$. The discussion begins with the first estimation.

$\tilde{\boldsymbol{x}}_t$ is related to $\tilde{\boldsymbol{x}}_{t-1}$ and $\tilde{\boldsymbol{p}}$ by equation (4.31). However, equation (4.31) is not in a closed form since $\tilde{\boldsymbol{x}}_t$ appears on both sides. To obtain the best closed form approximation to equation (4.31), a deterministic circuit simulation is first performed at the nominal conditions, i.e., $\mathrm{E}(\tilde{\boldsymbol{x}}_{t-1})$ and $\mathrm{E}(\tilde{\boldsymbol{p}})$, in order to find the nominal value for $\tilde{\boldsymbol{x}}_t$, denoted by $\tilde{x}_{t,nominal}$. Then the right hand side of equation (4.31) is evaluated at $\tilde{x}_{t,nominal}$ in order to remove its dependence on $\tilde{\boldsymbol{x}}_t$, i.e.,

$$\tilde{\boldsymbol{x}}_t = B_t(\tilde{\boldsymbol{p}}, \tilde{\boldsymbol{x}}_t, \tilde{\boldsymbol{x}}_{t-1}) \cong B_{t,closed}(\tilde{\boldsymbol{p}}, \tilde{x}_{t,nominal}, \tilde{\boldsymbol{x}}_{t-1}) \tag{4.36}$$

where $B_{t,closed}$ is the closed form approximation to B_t.

The same algorithm used for non-transient performance estimation from section 4.3.2 is applied here to estimate the interval MOG representations for $\tilde{\boldsymbol{x}}_t$. Here the function $B_{t,closed}$ is viewed as the nonlinear function f used in section 4.3.2. One subtlety to note here is that $\tilde{\boldsymbol{x}}_t$ is a vector instead of a single value, therefore the algorithm from section 4.3.2 is applied once for each dimension of $\tilde{\boldsymbol{x}}_t$ by pre-multiplying a vector constant to $B_{t,closed}$ to pick the correct dimension. For example, to pick the first dimension, the vector [1,0, ... ,0] is multiplied to $B_{t,closed}$.

The calculation of the gradient and Hessian of $B_{t,closed}$ does not involve any additional sensitivity analyses or circuit simulations. Recall from equation (**4.13**) and the discussion of SPICE simulation in Chapter 3 section , that $B_{t,closed}$ can be expressed as,

$$B_{t,closed}(\tilde{\boldsymbol{p}}, \tilde{x}_{t,nominal}, \tilde{\boldsymbol{x}}_{t-1}) = M(\tilde{\boldsymbol{p}}, \tilde{x}_{t,nominal}, \tilde{\boldsymbol{x}}_{t-1})^{-1} I(\tilde{\boldsymbol{p}}, \tilde{x}_{t,nominal}, \tilde{\boldsymbol{x}}_{t-1}) \tag{4.37}$$

where the entries in the matrix M are obtained from evaluating the numerically integrated and linearized device equations, and the vector I contains the values of the input voltage and current sources. Both of them consist of known, analytical and closed form equations of $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{x}}_{t-1}$, with the functional form independent of the particular time step. Therefore, the analytical expression for the gradient and the Hessian of $B_{t,closed}$ can be calculated beforehand. During the simulation, the calculation only involves evaluating these expressions at the desired nominal values for $\tilde{\boldsymbol{p}}$ and $\tilde{\boldsymbol{x}}_{t-1}$.

In order to obtain the interval MOG representation for $\boldsymbol{z}_t$, again a second order Taylor series expansion is performed on the function g_t and the same propagation procedure as outlined in section 4.3.2 is used, i.e.,

$$\boldsymbol{z}_t \cong g_t\big((\mathrm{E}(\widetilde{\boldsymbol{p}}), \mathrm{E}(\widetilde{\boldsymbol{x}}_{t-1}), \mathrm{E}(\widetilde{\boldsymbol{x}}_t), v_0\big) + \nabla_{g_t}^T\big|_{\mathrm{E}(\widetilde{\boldsymbol{p}}),\mathrm{E}(\widetilde{\boldsymbol{x}}_{t-1}),\mathrm{E}(\widetilde{\boldsymbol{x}}_t)} \begin{bmatrix} \widetilde{\boldsymbol{p}} - \mathrm{E}(\widetilde{\boldsymbol{p}}) \\ \widetilde{\boldsymbol{x}}_{t-1} - \mathrm{E}(\widetilde{\boldsymbol{x}}_{t-1}) \\ \widetilde{\boldsymbol{x}}_t - \mathrm{E}(\widetilde{\boldsymbol{x}}_t) \end{bmatrix}$$
$$+ \begin{bmatrix} \widetilde{\boldsymbol{p}} - \mathrm{E}(\widetilde{\boldsymbol{p}}) \\ \widetilde{\boldsymbol{x}}_{t-1} - \mathrm{E}(\widetilde{\boldsymbol{x}}_{t-1}) \\ \widetilde{\boldsymbol{x}}_t - \mathrm{E}(\widetilde{\boldsymbol{x}}_t) \end{bmatrix}^T H_{g_t}\big|_{\mathrm{E}(\widetilde{\boldsymbol{p}}),\mathrm{E}(\widetilde{\boldsymbol{x}}_{t-1}),\mathrm{E}(\widetilde{\boldsymbol{x}}_t)} \begin{bmatrix} \widetilde{\boldsymbol{p}} - \mathrm{E}(\widetilde{\boldsymbol{p}}) \\ \widetilde{\boldsymbol{x}}_{t-1} - \mathrm{E}(\widetilde{\boldsymbol{x}}_{t-1}) \\ \widetilde{\boldsymbol{x}}_t - \mathrm{E}(\widetilde{\boldsymbol{x}}_t) \end{bmatrix} \tag{4.38}$$

4.3.4 **Runtime Analysis**

In the non-transient problem as discussed in section 4.3.2, the runtime of the propagation algorithm is determined by the following portions:

1. Calculation of the gradient ∇_f and Hessian matrix H_f of function f, around the nominal parameter values, $\mathrm{E}(\widetilde{\boldsymbol{p}})$;
2. Calculation of the moments $\mathrm{E}(\boldsymbol{z}^l)$, $\mathrm{E}\big(f_{approx}{}^l\big)$, $\mathrm{E}\left(\frac{\partial z}{\partial \varepsilon_i}\right)^q$ and $\left(\frac{\partial f_{approx}}{\partial \varepsilon_i}\right)^q$;
3. Formulating the equations in (4.28) and (4.29); and
4. Solving the system of equations by using a non-linear solver.

If the number of process/device parameters is *k*, and the number of noise variables is *n*, then the runtime for portion (1) is O(*min*(*k*,*n*)) multiplied by the runtime of one circuit simulation if adjoint network approach is used, and O(*min*(*k*,*n*)^2) multiplied by the runtime of one circuit simulation if direct sensitivity method is used . In the case where a SPICE simulation is required, the runtime for a single circuit simulation is generally in the order of the size of the circuit to the power 3. The runtime of portion (2) and portion (3) is O(*Cnk* max(*Q*, *L*)) where *C* is the maximum number of Gaussian components involved, *n* is the total number of noise variables, and *L* is the number of moments required to preserve, *Q* is defined in equation (4.29). The runtime for portion (4) is largely dependent on the type of solver used, the readers are referred to [4.14]-[4.17] for details. We argue that when the circuit is simulated using SPICE, as in the case of standard cell simulations, critical path simulations and analog circuit simulations, the runtime of portion (1) dominates over the other portions.

In the non-transient problem as discussed in section 4.3.3, the cost is divided into the calculation of $\widetilde{\boldsymbol{x}}_{t-1}$ and the calculation of $\boldsymbol{z}_t$ for each time step. However, in this case, the cost for Gradient and Hessian is very small. The analytical expressions for the Gradient and Hessian matrix used to obtain $\widetilde{\boldsymbol{x}}_{t-1}$ is pre-computed, and the cost only consists of numerical evaluations of these expressions. In addition, since the function g_t that relates the performance $\boldsymbol{z}_t$ is a given function, thus the Gradient and the Hessian can be obtained analytically. The main cost incurred in each time step is the moment calculation (portion (2) in the above list), moment matching equation formulation (portion (3) in the above list) and nonlinear equation solving (portion (4) in the above list). The cost follows the same analysis as in the case of the non-transient problem, but multiplied by the number of nodes in the circuit, i.e., the size of the vector $\widetilde{\boldsymbol{x}}_{t-1}$. Thus,

the runtime is dominated by O(*mCnk* max(*Q*, *L*)) where *m* is the dimension of the vector $\tilde{\boldsymbol{x}}_{t-1}$, *C* is the maximum number of Gaussian components involved, *n* is the total number of noise variables, *L* is the number of moments required to preserve, *Q* is defined in equation (4.29).

4.3.5 **Scalability**

In the previous section, the algorithm runtime is analyzed as a function of the number of process/device parameters. In this section, the algorithm scalability as a function of the circuit size is analyzed.

For the non-transient problem as discussed in section 4.3.2, the scalability is determined by the scalability of obtaining the gradient and the Hessian of the performance function. If the adjoint network approach is used, and assuming that the number of device/parameters increases as a linear function of the circuit size, then the runtime is given by O(s^4) where *s* is the circuit size.

For the transient problem as discussed in section 4.3.3, since analytical expressions can be pre-computed for the gradient and Hessian of the performance function, then the scalability of the runtime is determined by the formulation of moment-matching equations and by the nonlinear system of equation solver. Algorithms that can efficiently solve large-scaled nonlinear systems of equations have been investigated and developed in various works [4.18]-[4.20]. The readers can refer to these works for details on the scalability of solving the nonlinear systems in question. An analysis on the scalability of the formulation of the moment-matching equations is given below.

In general as the circuit sizes increase, the number of device/process parameters also increases linearly as a function of the circuit size. In addition, the number of noise variables required to characterize these process/device parameters increases as a linear function of the number of device/process parameters and increases as a linear function of the number of Gaussian components. Therefore, if the circuit size is *s*, the number of Gaussian component is *C*, then the runtime of the formulation of the moment-matching equation is give by O(C^2s^3)

4.4 Circuit Example

To test our algorithm we used a circuit path of 100 gates containing inverters, NAND gates, and NOR gates. Our selection of the type of gates is limited by the process variability data in our disposal.

4.4.1 **Cell parameter distribution extraction**

We obtained context-induced delay variation data for various type of inverters, NAND gates and NOR gates. For our algorithm, we need to know the distributions of the device parameters rather than of the delay, thus we back-calculated plausible distributions by simulations and approximations.

The BSIM device model parameters gate length (*L*), the correction factor on threshold voltage (*delvt0*), and the multiplier on mobility (*mulu0*) are selected for

capturing the context-induced variability, consistent with the context-induced variability models used in industry [4.8].

Since gate lengths are largely affected by photolithographical context, we performed lithography simulations in Calibre by randomly placing the cell of interest in 50 different contexts and by simulating the resulting printed gate length. We use this simulation result in order to estimate the distribution of the gate length. Figure 4.3 shows the pair-wise scatter plot of the gate length distributions for the four transistors in a NOR2X1 gate. We found that in all the cells we simulated, the gate length distributions can be modeled as a single Gaussian distribution and that they are independent among the transistors in a gate.

We simulated the logic gates using SPICE given the distributions of the gate length, and obtained changes in delay due to gate length alone. We then subtracted the delay data given by our simulated results. The residual is then assumed to be purely due to changes in *delvt0* and *mulu0*.

SPICE simulation was used to obtain the sensitivity of delay due to *delvt0* and *mulu0*. In addition, we assume that *delvt0* and *mulu0* each is responsible for 50% of the changes in cell delay. From here, we back-calculated the values for *delvt0* and *mulu0* given the context-induced delay data. We found that the distributions are non-Gaussian.

To cast the distribution of *delvt0* and *mulu0* into MOG representation, we performed a multivariate mixture of Gaussian fitting on the data using the Expectation-Maximization algorithm [4.4]. The number of Gaussian components was chosen such that the Akaike's information criterion(AIC) is minimized. In the cells we examined, the number of components ranges from 4 to 7. Figure 4.4 and Figure 4.5 show sample histograms of *delvt0* and *mulu0* for a transistor in a NOR2X1 gate obtained using this approach and the marginal distribution was fitted using MOG.

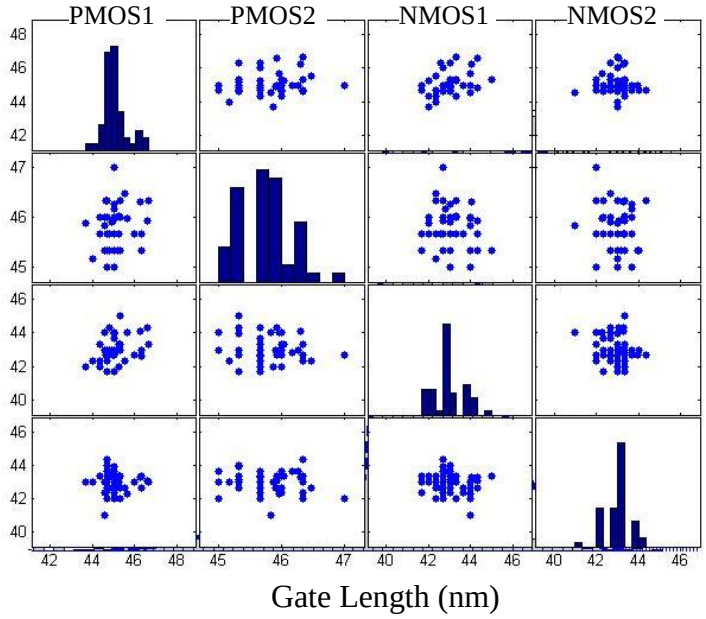

Figure 4.3: pair-wise scatter plot of the gate length distributions for the four transistors in a NOR2X1 gate.

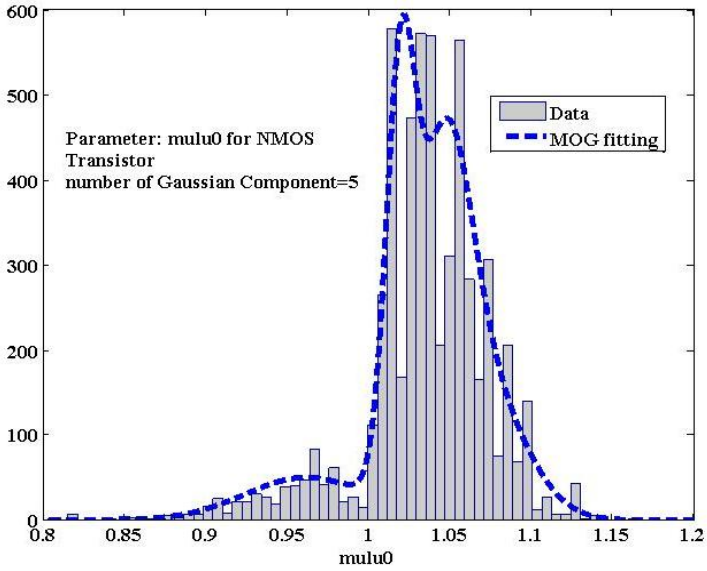

Figure 4.4: histograms and fitted MOG of *mulu0* for NMOS transistor in NOR2X1.

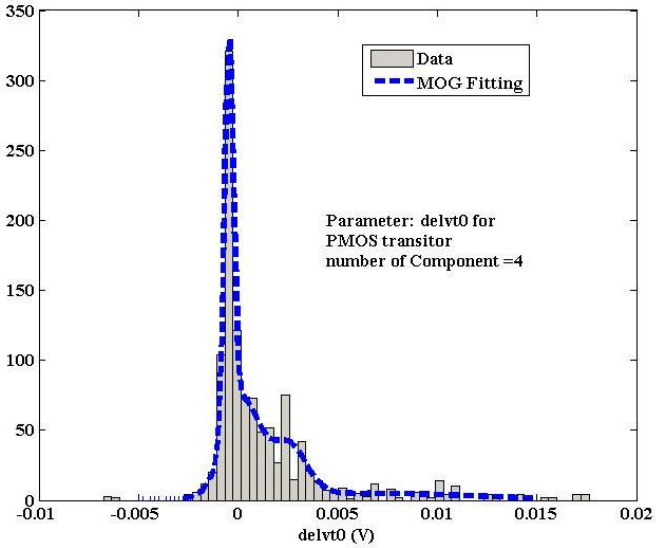

Figure 4.5: histograms and fitted MOG of delvt0 for PMOS transistor in NOR2X1.

4.4.2 **Simulation Setup**

Spatial correlations are imposed along the logic chain such that closely situated cells are perfectly correlated and the correlation coefficients fall off to zero after 10 gates. The total number of parameters (*L, mulu0, delvt0*) in the test circuit is 1140; the number of noise variables is 147.

Monte-Carlo simulations are then performed on the circuit and the circuit delay is measured. We found that the statistics of the delay obtained converge after 10,000 Monte-Carlo simulations and thus we use that sample size as our baseline for comparison purposes. The delay distribution is plotted in Figures 4 and 5, and it fails the normality test at a confidence level >99%.

4.4.3 **Simulation Result**

We next performed an MOG propagation using the algorithm outlined in section 3. We compared the E(delay^l) for l=1,2,…,7 using Monte-Carlo simulation against our algorithm. The result is summarized in Table 4.1. The comparison of the distribution obtained by MOG propagation and Monte-Carlo analysis is shown in Figure 4.6 and Figure 4.7. The runtime of the two methods is compared and summarized in Table 4.2.

Table 4.1: Comparison of the E(delay^l) between Monte-Carlo baseline simulation and the MOG propagation method.

Moment Order(l)	Monte Carlo baseline (sec^l)	MOG propagation (sec^l)	Relative error (%)
1	2.9187E-09	2.9416E-09	0.78
2	8.5392E-18	8.6551E-18	1.35
3	2.5043E-26	2.5444E-26	1.60
4	7.3623E-35	7.4663E-35	1.41
5	2.1696E-43	2.1844E-43	0.68
6	6.4088E-52	6.3651E-52	0.68
7	1.8977E-60	1.8450E-60	2.78
8	5.6327E-69	5.3123E-69	5.69

Table 4.2: Runtime comparison between Monte-Carlo baseline simulation and the MOG propagation method.

Monte Carlo (sec)	MOG Propagation			Speed up
	Gradient and Hessian Calculation (sec)	Other calculations (sec)	Total (sec)	
26669	391	105	496	54X

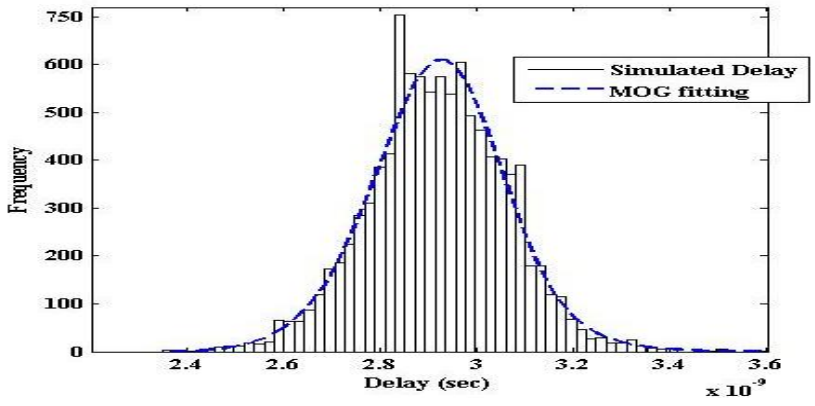

Figure 4.6: Histogram showing the delay distribution from Monte-Carlo simulations and MOG propagation algorithm.

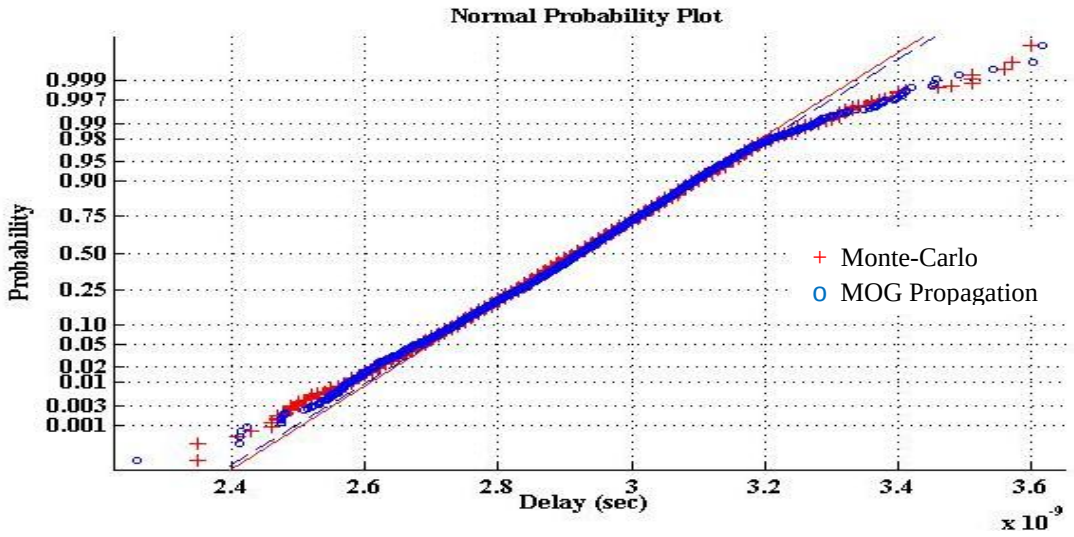

Figure 4.7: Normal plot showing comparing the delay distribution from Monte-Carlo simulations and the MOG propagation algorithm.

4.5 Conclusions

In this chapter, a mixture of Gaussian (MOG) propagation algorithm for statistical circuit simulation has been formulated. The interval representation is used as the tool to capture the distribution which allows for significant runtime reduction when the circuit parameters are correlated. The algorithm has been tested with circuit paths involving thousands of parameters and found that using our MOG propagation results in a 54X speed up as compared to Monte-Carlo simulation, while maintaining very good accuracy.

References for Chapter 4

[4.1] L. H. de Figueiredo and J. Stolfi, "Affine arithmetic: concepts and applications." *Numerical Algorithms* **37** (1–4), 147–158. 2004.

[4.2] B. Sturmfels, "Solving systems of polynomial equations*"*, Regional conference series in mathematics, No 97, American Mathematical Society, October 1, 2002.

[4.3] Marc Giustia, Grégoire Lecerfa and Bruno Salvyb, "A Gröbner Free Alternative for Polynomial System Solving", *Journal of Complexity*, Vol 17, Issue 1, March 2001, Pages 154-211.

[4.4] S.Borman, "The Expectation Maximization Algorithm: A short tutorial", available online at http://www.seanborman.com/publications/EM_algorithm.pdf, 2004.

[4.5] S. W. Director and R. A. Rohrer. "The generalized adjoint network and network sensitivities.", IEEE Trans. Circuit Theory, CT-16:318-23, 1969.

[4.6] A.K. Setha and P.H. Roea, "An efficient method for the computation of second order network sensitivity functions: A comment", Computer-Aided Design, Vol. 5, Issue 2, April 1973, Page 105.

[4.7] J. Duros, "An efficient method for the computation of second-order network sensitivity functions Computer-Aided Design, Volume 2, Issue 1, Autumn 1969, Pages 37-42

[4.8] V. Moroz et. Al., "Stress aware Design Methodology", in the proceedings of ISQED, pp. 807-812. 2006.

[4.9] Liang-Teck Pang, Kun Qian, Costas J. Spanos and Borivoje Nikolic, "Measurement and Analysis of Variability in 45 nm Strained-Si CMOS Technology", IEEE Journal of Solid-State Circuits, Volume 44, Issue 8, Aug. 2009 Page(s):2233 - 2243

[4.10] Lerong Cheng, Puneet Gupta, Costas Spanos, Kun Qian, and Lei He, "Physically Justifiable Die-Level Modeling of Spatial Variation in View of Systematic Across Wafer Variability", Design Automation Conference 2009.

[4.11] Fukunaga, K. and Flick, T., "Estimation of the parameters of a Gaussian Mixture Using the method of Moments", IEEE Trans. On Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No. 5, July 1983. Pp 410-416.

[4.12] Byrnes, J., "Estimation of Gaussian Mixtures from Moments", In the Proceedings of 2008 Seventh International Conference on Machine Learning and Applications, pp 421-425.

[4.13] R. Redner and H. Walker. "Mixture densities, maximum likelihood and the EM algorithm", SIAM Review, 26(2), 1984.

[4.14] C.G. Broyden, "A class of methods for solving nonlinear simultaneous equations", Mathematics of Computation, 19, 577-593, 1965.

[4.15] J.E. Denis, M. ElAlem, K. Williamson, "A Trust-Region Algorithm for Least-Squares Solutions of Nonlinear Systems of Equalities and Inequalities", SIAM Journal on Optimization 9(2), 291-315, 1999.

[4.16] W. Gragg, G. Stewart, "A stable variant of the secant method for solving nonlinear equations", SIAM Journal of Numerical Analysis, 13, 889-903, 1976.

[4.17] J. E. Denis, "On Newton's method and nonlinear simultaneous replacements," SIAM J. Numer. Anal., vol. 4, pp. 103–108, 1967.

[4.18] C. Grosan, A. Abraham, "A New Approach for Solving Nonlinear Equations Systems", *IEEE trans. on systems, man and cybernetics*, vol. 38, No. 3, pp. 698-714, 2008.

[4.19] B. W. Bader, "Tensor–Krylov methods for solving large-scale systems of nonlinear equations," SIAM J. Numer. Anal., vol. 43, no. 3, pp. 1321–1347, 2005.

[4.20] J. M. Martinez, "Algorithms for solving nonlinear systems of equations," *Continuous Optimization: The State of the Art*, E. Spedicato, Ed. Norwell, MA: Kluwer, 1994, pp. 81–108.

Chapter 5 Conclusions

In this dissertation, the problem of uncertainty propagation in statistical circuit simulation is addressed. Uncertainty in circuit process and device parameters arises due to manufacturing variability. Since electrical circuits are, in general, complex and nonlinear systems, the problem of estimating the circuit performance efficiently and accurately due to the variations in process/device parameters is very challenging.

Conventional methods to estimate variations in circuit performance include worst case corner analysis and Monte-Carlo simulation. The worst case corner analysis method has become increasingly inadequate to produce realistic estimations due to the increase in both magnitude and complexity of process variations [5.3]. Monte-Carlo simulations can produce very accurate estimates of the circuit performance distributions. However, the simulation time needed to perform Monte-Carlo type analysis can be very long, making it practically intractable in complex circuit design.

A novel interval based circuit simulation algorithm is developed in order to solve this problem. Intervals are originally used to represent the upper and lower bounds of a quantity with uncertainty. [5.1] introduces the idea of representing intervals with a linear combination of noise variables distributed with uniform distributions to obtain tighter bounds for uncertainty quantification. In this work, a similar formulation as in [5.1] is used to define intervals, but the noise variables are modified to follow Gaussian distributions. With this extension, the intervals can be viewed equivalently to any Gaussian distribution. Algebraic operations, such as addition, subtraction and multiplication, between these intervals are defined using a novel moment-preserving approximation, such that the result of an interval operation is also represented in an interval form that best approximates the distribution of the true result.

When the uncertainty in the circuit process and device parameters can be captured by correlated Gaussian distributions, the process/device parameters are first represented by appropriate interval representations. An interval-valued SPICE simulator, in which all real number operations are replaced by interval operations, is used to simulate the circuit. The simulation results are therefore interval-values that can be used to extract

performance statistics. In this approach, only one circuit simulation is required to obtain the best Gaussian distribution approximation for any circuit performance.

The algorithm is tested on RC circuits and transistor circuits with excellent simulation accuracy as compared to Monte Carlo simulation results. It is shown analytically that the runtime of the interval valued circuit simulation is on the order of $O(n+m)O(c^3)$ where n is the average number of noise variables per interval operation, m is the average number of noise variables shared between any two interval quantities and c is the number of nodes in the circuit. Experimental data shows that the runtime scales linearly with the number of process/device parameters subjected to process variations and also scales linearly with the complexity of the variability model used for the process/device parameters.

In the case when the process/device parameters cannot be modeled with Gaussian distributions, a novel extension to the interval representation is proposed. Non-Gaussian distribution of process/device parameters can result from the combining effects of random and systematic variations [5.2] in state of the art IC technologies. In the proposed algorithm, a Mixture of Gaussian (MOG) distribution is used to approximate all non-Gaussian distributions.

Each Gaussian component in a mixture of Gaussian distribution is reduced to an interval representation. The sets of all interval representations for all the Gaussian components are organized in a vector, and pre-multiplied by the transpose of a random vector with a discrete probability density function properly assigned to ensure that the expression correctly represents the original MOG distribution.

A statistical circuit simulation procedure is developed using these interval MOG representations. Similar to the interval circuit simulation for Gaussian distributions, at each step during the simulation algorithm, all quantities are represented by interval MOG representations. The algorithm has been tested on circuit paths of 100 stages containing inverters, NAND gates and NOR gates. The simulation result of the proposed algorithm agrees very well with Monte-Carlo simulation. In addition, the runtime of the proposed algorithm shows a 54X speed up compared to Monte-Carlo simulation.

The proposed interval-value based simulation engine for both Gaussian and non-Gaussian process variations can be directly applied to fast and accurate standard cell library characterization, where the distribution of the cell delay and power are calculated. In addition, the distribution information provided by the proposed simulation method can be feed into statistical timing analysis engine for full-chip level timing closure. The proposed simulation engine can also be incorporated into statistical circuit optimizations.

Some limitations apply to the current implementation of the algorithm. First and foremost, the device equations used in the implementation are simple square-law MOSFET device equations that do not taken into account deep-submicron device effects. However, it should be noted that the interval-based algorithms developed in this dissertation do not impose any fundamental constraints on the type of device equations used. Incorporation of more sophisticated device models is merely a matter of implementation, but such implementation could tedious and error prone. One could envision an automated “translator” that could take a scalar model and translate it to interval representation. This type of automating “translation” is conceivable, but its study and implementation are well beyond the scope of the present work.

Secondly, the interval-valued SPICE simulation engine implemented in this work is a simplified version of commercial SPICE simulators and does not include advanced features such as simulation convergence tuning and variable step time size. Due to this limitation, circuits that involving feedback loops cannot be simulated correctly. Also, the interval SPICE simulator implemented in this work is only capable of transient and DC circuit analysis. The interval algorithms developed in this thesis focus only on transient circuit analyses; other types of analyses (such as harmonic, etc) can be part of the future investigation.

Other possibilities for future work include further improvements in the runtime of the interval developed algorithms. For the interval propagation algorithm for Gaussian distributions, the major runtime bottleneck comes from the introduction of extra noise variables after every non-linear operation. Even though a noise variable pruning algorithm (see section 3.4.4 for details) is used at the end of each time step, extra noise variables are still introduced by the intermediate calculations during a time step. However, many noise variables introduced does not carry additional information. For example, in a chain of m interval multiplications, m additional noise variables are introduced and included in the final result. However, since only the first two moments are preserved through this chain of multiplications, the final result can be represented with only one extra noise variable. A final result with only one extra noise variable can be found by directly defining a moment-preserving interval operation for the chain of multiplication. Runtime reduction can be achieved in this way by defining a moment-preserving interval operation directly on the device model equation by viewing the entire equation as one operation.

In the interval MOG propagation algorithm, it has been noted that the final circuit performance distribution resembles a Gaussian distribution except at the tails. It is often the case that the tail of a distribution is more important in statistical circuit analysis, especially in the context of state of the art IC design, where the acceptable rate of failure is very small. In the mixture of Gaussian representation for this kind of performance distribution, one component is used to capture the center part of the distribution density function that follows a Gaussian *pdf*. A few additional Gaussian components are placed at the tails to capture the deviations from a Gaussian *pdf*. Therefore, it may be possible to focus only on those additional Gaussian components used to capture the distribution tail, and to propagate only those components instead. Compared to propagating the entire mixture of Gaussian distributions, runtime can be reduced using this method.

Other future works include evaluating the algorithms on circuits that are more functionally complex (e.g., circuit involving feedback loops) and that are of larger sizes. Such evaluations are not included in this dissertation due to the limitations of the current implementation of the algorithm as noted above, and due to the limitations on the runtime of the current implementation.

As the device dimensions shrink, statistical variations in circuit performances are becoming a more severe issue. Circuit simulation results obtained using corner-based worst case analysis can deviate significantly from silicon data, resulting possible functional failures in sensitive analog and digital designs after chip fabrication. Statistical circuit simulations, on the other hand, can provide results that are more close to the real silicon data. Therefore, there is an increasing need for statistical circuit simulations in

nano-scale designs. Considering the large runtime of Monte-Carlo simulations, a fast and efficient statistical circuit simulation at a SPICE level will be the enabling technology for the transition from transitional corner-based design methods to true statistical design methods.

References for Chapter 5

[5.1] L. H. de Figueiredo and J. Stolfi (2004) "Affine arithmetic: concepts and applications." *Numerical Algorithms* **37** (1–4), 147–158.

[5.2] Kun Qian, Costas J. Spanos, "A Comprehensive Model of Process Variability for Statistical Timing Optimization", in Proceedings of SPIE, San Jose, 2008.

[5.3] Li X. and Pileggi L., "Statistical performance modeling and optimization", Foundations and trends in EDA. Vol 1, No. 4, 331-490(2006).