UNIVERSITY OF CALIFORNIA
RIVERSIDE

# Statistical Performance Characterization and Analysis of Nano-Scale VLSI Circuits

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Electrical Engineering

by

Ruijing Shen

December 2011

ACKNOWLEDGEMENTS

My thanks and appreciation goes to my thesis committee members, Dr Jianlin Liu and Dr Albert Wang for their direction, dedication, and invaluable advice. I would like to express my deepest gratitude to my advisor, Dr Sheldon X.-D. Tan, for his help, trust and guidance. There exist wonders as well as frustrations in academic research. His kindness, insight and suggestions always let me to the right way. A special word of thanks for all the students in the Mixed-Signal Nanometer VLSI Research Lab (Dr Wei Wu, Dr Duo Li, Dr Ning Mi, Dr Boyuan Yan, Dr Pu Liu, Jian Cui, Hai Wang, Xuexin Liu, etc.) for the collaborative research works and help, which lead to some of the presented works in this dissertation.

This dissertation will not be finished without the financial supports for the author. I highly appreciate the consistent supports of of National Science Foundation and National Natural Science Foundation of China (NSFC). The research in this dissertation is funded in part by NSF grant under, in part by NSF grants under No. OISE-0623038 and OISE-0929699, in part by NSF grant under No. CCF-0448534, in part by National Natural Science Foundation of China (NSFC) grant under No. 60828008.

Last but not lease, I would like to thank to my parents, my husband Boyuan Yan, and all my friends for their constant support and encouragement during the writing of this dissertation.

ABSTRACT OF THE DISSERTATION

Statistical Performance Characterization and Analysis of Nano-Scale VLSI Circuits

by

Ruijing Shen

Doctor of Philosophy, Graduate Program in Electrical Engineering
University of California, Riverside, December 2011
Dr. Sheldon X.-D. Tan, Chairperson

The performance of integrated circuits (IC) is becoming less predictable as technology scales to the sub-90-num regime. Disparate sources of variations stem from the manufacturing process and ultimately translate to a parametric yield loss. To improve parametric yield, efficient algorithms are required to accurately predict the performance of a circuit at the design stage. However, given the high complexity of design and the presence of a large number of correlated parameters that exhibit significant variations, traditional Monte Carlo (MC) method becomes inefficient as a large number of sampling points are required for an accurate statistical description of the circuit response.

To mitigate this problem, a novel methodology for statistical performance analysis is proposed by representing statistical processes in a deterministic way to determine the key characteristics of statistical distributions. The proposed methodology has been successfully applied to statistical full-chip leakage analysis and capacitance extraction. For statistical full-chip leakage analysis, a general framework has been provided to derive the full-chip leakage currents or powers as closed form functions of process variation parameters. To the best knowledge of the author, this is the first full-chip statistical leakage analysis algorithm considering all types of spatial correlations with only linear time complexity $O(N)$. Furthermore, it is extendable to incremental analysis for even more promising computing speed for larger problem sizes. For statistical capacitance extraction, a 3D statistical capacitance extraction method called *StatCap* is proposed, in which orthogonal polynomials are used to represent the statistical processes and the analytic second-order orthogonal polynomials are derived from the capacitance integrated equations to give more accurate results without loss of efficiency compared to the linear models. Experimental results show that *StatCap* is two orders of magnitude faster than the recently proposed statistical approach and many orders of magnitude faster than the MC.

To improve parametric yield, not only efficient algorithms are required to accurately predict the performance of a circuit, but also efficient techniques are highly desirable for chip design. Toward this direction, a novel voltage binning technique is proposed in the last part of the dissertation. The proposed method makes it possible to predict maximum bin numbers required under the uniform binning scheme, and model the optimal binning scheme as a set-cover problem. To achieve the same yield as the uniform approach, the proposed method can significantly save the number of bins and only takes very small CPU time cost.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

## 1.1 Motivation

### 1.1.1 Nanometer chip design in an uncertain world

As VLSI technology scales into the nanometer regime, chip design engineering faces several challenges in maintaining historical rates of performance improvement and capacity increase with CMOS technologies. One profound change in the chip design business is that engineers can't put the design precisely into the silicon chips. Chip performance, manufacture yield and lifetime become unpredictable at the design stage. Chip performance, manufacture yield and lifetime can't be determined accurately at the design stage. The main culprit is that many chip parameters – such as oxide thickness due to chemical and mechanical polish (CMP) and impurity density from doping fluctuations – can't be determined precisely, and thus are unpredictable. The so-called manufacture process variations start to play a big role and their influence on the chip's performance, yield and reliability becomes significant [12, 30, 51, 69, 52].

As a result, it is imperative to develop new design methodologies to consider the impacts of various process and environmental uncertainties and elevated temperature on chip performance. Variational impacts have to be incorporated into every steps of design process to ensure the reliable chips and profitable manufacture yields. The design methodologies and design tools from system level down to the physical levels have to consider variability and thermal impacts on the chip performance, which calls for new statistical and thermal-aware optimization approaches for designing nanometer VLSI systems.

Performance modeling and analysis of nanometer VLSI systems in the presence of process-induced variation and uncertainness is the one of the grand challenging problems facing IC chip designers and design tool developers. How to efficiently and accurately assess the impacts of the process variations on circuit performances in the various physical design steps are critical for fast design closure, yield improvement, cost reduction of VLSI design and fabrication processes. The design methodologies and design tools from system level down to the physical levels have to embrace variability impacts on the nanometer VLSI chips, which calls for statistical/stochastic based approaches for designing 90nm and beyond VLSI systems.

Process variation can occur at different levels: wafer level, inter-die level, and intra-die level. Furthermore, they are caused by different sources such as lithography, materials, aging, etc [7, 69]. Some of the variations are systematic, i.e., those caused by the lithography process [18, 55]. Some are purely random, i.e., the doping density of impurities and edge roughness [7].

### 1.1.2 Process induced variations in a nutshell

During the chip fabrication process, there exists thickness and width variation, which is referred by process induced variation. Inaccurate model of process variations will cause very high potential for silicon failure. Due to the limitations of manufacturing process, there are many causes for process variation. Take chemical mechanical polishing (CMP) for instance, which is a technique used to manufacture copper interconnect. In CMP, a slurry is used to grind down the copper. As a result, wider conductors will have more copper loss compared to conductors that are skinnier. Therefore, CMP changes the thickness of interconnect wires, which results in shift of the parasitic resistance, capacitance and inductance. Another cause of the process variation is the limit of the wavelength of light used in photolithography. Optical proximity correction (OPC) [62] is a technique to manufacture structures with dimensions less than the wavelength of the light used to illuminate the wafer. However, the manufactured dimensions will vary from the drawn dimensions even with the use of OPC.

In general, process variations can be classified into the following categories [12]: inter-die and intra-die. Inter-die variations are the variations from die to die, while intra-die variations correspond to variability within a single chip. As a result, inter-die variations are global variables, and affect all the devices on a chip in the same way, i.e., make the transistor gate channel lengths of all the devices on the same chip smaller. Intra-die variations may affect different devices differently on the same chip, i.e., make some devices have smaller gate oxide thicknesses and others have larger transistor gate oxide thicknesses. In addition, intra-die variations may exhibit spatial correlation, i.e., it is more likely for devices located close to each other to have similar characteristics than those placed far away. Under inter-die variation, if the circuit performance metrics such as (power, timing, noises) of all gates or devices are sensitive to the process parameters in similar ways, then the circuit performance can be analyzed at multiple process corners using deterministic analysis methods. However, statistical methods must be used to correctly predict the performances if intra-die variations are involved. As some circuit performance metrics such as leakage current varies exponentially with these parameters, simple use of worst-case values for all parameters can result in exponentially larger leakage estimates than the actual values, and is too inaccurate to be used in practical cases.

### 1.1.3 A simple look at process variations modelling

In this book, we can model parameter variation as follows,

$$\delta_{total} = \delta_{inter} + \delta_{intra}, \tag{1.1}$$

where $\delta_{inter}$ and $\delta_{intra}$ represent the inter-die variation and intra-die variation, respectively. In some works such as in [9, 69, 35], $\delta_{inter}$ and $\delta_{intra}$ are both modeled as Gaussian random variables. In general, we will consider both the Gaussian and non-Gaussian cases.

For $\delta_{intra}$, the value of parameter $p$ located at $(x, y)$ can be modeled as a location-dependent normally distributed random variable [40]:

$$p = \mu_p + \delta_x + \delta_y + \epsilon, \tag{1.2}$$

where $\mu_p$ is the mean value (nominal design parameter value) at $(0, 0)$, and $\delta_x$ and $\delta_y$ stand for the gradients of the parameter indicating the spatial variations of $p$ along the $x$ and $y$ directions, respectively. $\epsilon$ represents the random intra-chip variation. Due to spatial correlations in the intra-die variation [77], the vector of all random components across the chip $\vec{\epsilon}$ has a correlated multivariate normal distribution, $\vec{\epsilon} \sim N(0, \Sigma)$, where $\Sigma$ is the covariance matrix of the spatially correlated parameters. We will have a more comprehensive coverage of variation modeling in Chapter 2.

## 1.2 Objectives and results of this thesis

The main objective of this dissertation is to develop new theories and methodologies for efficient and accurate statistical analysis of very large scale integrated circuits. The key contribution of this research is the introduction and the exploration of several novel technique for statistical analysis and characterization of nano-scale VLSI designs. The major achievements accomplished in this dissertation are as follows:

- A novel fast and accurate method for full-chip statistical analysis of leakage power is proposed. The method is capable to consider both intra-die and inter-die variations with strong spatial correlations without any limitation of static leakage models. Unlike many existing approaches, no grid-based partitioning and approximation are required. Instead, the spatial correlations are naturally handled by orthogonal decompositions. The proposed method is very efficient and it becomes linear in the presence of strong spatial correlations. Experimental results show that the proposed method is about 16× faster than the recently proposed method [9] with constant better accuracy.

- A new statistical leakage characterization in standard cell library (SCL) is put forward for fast full-chip statistical leakage estimation. This novel characterization in SCL works in the presence of any spatial correlation condition (strong or weak), and can guarantee $O(N)$ time complexity, where $N$ is the number of grids on chip. The numerical examples in 45nm CMOS process demonstrate the proposed algorithm is 1000X faster than a recently proposed grid-based method [9] with similar accuracy and many orders of magnitude times speedup over the Monte Carlo method.

- Further more, an incremental analysis algorithm is proposed to update the chip-level statistical leakage information efficiently after a few changes are made. The incremental analysis provides about 10X further speedup, which is 10,000X compared to [9]. We expect the incremental analysis could achieve more speedup over the full leakage analysis for larger problem sizes.

- An efficient non-linear 3-D statistical capacitance extraction method is developed. The new method, called *StatCap*, uses orthogonal polynomials to represent the non-linear statistical processes. Experimental results show that *StatCap* is two orders of magnitude faster than the recently proposed statistical capacitance extraction method based on the spectral stochastic collocation approach [86] and many orders of magnitude faster than the Monte Carlo method for several practical conductor structures.

- A new concept called "valid voltage segment" is proposed for yield optimization technique using voltage binning method to improve yield of chips. The new concept of valid voltage segment enables a series of intuitionistic and efficient technique for yield analysis and improvement. For instance, a formulation to predict the maximum number of bins required under the uniform binning scheme is proposed. Also, an optimal binning scheme can be modeled as a set-cover problem. A greedy algorithm is developed to solve the set-cover problem in an incremental way. The proposed method is also extendable to deal with a range of working supply voltages for dynamic voltage scaling under different operation modes (like lower power and high performance modes).

## 1.3 Organization

The rest of this dissertation is organized as follows. Chapter 2 starts with providing background knowledge about some fundamental statistical and stochastic mathematic models, concepts and algorithms which we will used in latter part of the dissertation..

Chapter 3 to Chapter 5 focus on the new techniques for statistical full-chip leakage power consumption analysis considering process variations. After that, Chapter 6 introduces a statistical capacitance extraction method for interconnect conductors considering process variations. Chapter 7 presents a yield optimization technique using voltage binning method to improve yield of chips. Finally, Chapter 8 concludes the dissertation.

# Chapter 2

# Background of Statistical Analysis in VLSI

## 2.1 Multiple random variables for VLSI design

### 2.1.1 Components of covariance in process variation

In general, process variation can be classified into the two categories [9]: inter-die and intra-die. Inter-die variations are the variations from die to die, while intra-die variations correspond to variability within a single chip. As a result, inter-die variations are global variables, and affect all the devices on a chip in the same way, i.e., make the transistor gate channel lengths of all the devices on the same chip smaller. Intra-die variations may affect different devices differently on the same chip, i.e., make some devices have smaller gate oxide thicknesses and others have larger transistor gate oxide thicknesses. In addition, intra-die variations may exhibit spatial correlation, i.e., it is more likely for devices located close to each other to have similar characteristics than those placed far away.

In general, we can model parameter variation as follows,

$$\delta_{total} = \delta_{inter} + \delta_{intra}, \tag{2.1}$$

where $\delta_{inter}$ and $\delta_{intra}$ represent the inter-die variation and intra-die variation, respectively. In some works such as in [9, 68, 35], $\delta_{inter}$ and $\delta_{intra}$ are both modeled as Gaussian random variables. In this chapter, we will discuss both the Gaussian and non-Gaussian cases. Due to the global effects of inter-die variation, we will use a single random variable $\delta_{inter}$ for all gates/grids in a chip.

For $\delta_{intra}$, the value of parameter $p$ located at $(x, y)$ can be modeled as a location-dependent normally distributed random variable [40]:

$$p = \mu_p + \delta_x + \delta_y + \epsilon, \tag{2.2}$$

where $\mu_p$ is the mean value (nominal design parameter value) at $(0, 0)$, and $\delta_x$ and $\delta_y$ stand for the gradients of the parameter indicating the spatial variations of $p$ along the $x$ and $y$ directions, respectively. $\epsilon$ represents the random intra-chip variation. Due to spatial correlations in the intra-chip variation, the vector of all random components across the chip $\vec{\epsilon}$ has a correlated multivariate normal distribution, $\vec{\epsilon} \sim N(0, \Sigma)$, where $\Sigma$ is the covariance matrix of the spatially correlated parameters.

A grid-based method is used in works such as [9] to consider speed-up. In the grid-based method, the intra-die spatial correlations of parameters are modeled by partitioning the die region into $\sqrt{n}$ row $\times$ $\sqrt{n}$ col $= n$ grids. Since devices close to each other are more likely to have similar characteristics than those placed far away, grid-based methods assume perfect correlations among the devices in the same grid, high correlations among those in close grids and low to zero correlations in far-away grids.

For example, in Fig. 2.1: $Gate_1$ and $Gate_2$ (whose sizes are shown to be exaggeratedly large) are located in the same grid square, so their parameter variations (such as the variations of their gate channel length) are assumed to be always identical. $Gate_1$ and $Gate_3$ lie in neighboring grids, so their parameter variations are not identical but highly correlated due to their spatial proximity (for example, when $Gate_1$ has a larger than nominal gate channel length, $Gate_3$ is more likely to have a larger than nominal gate channel length). On the other hand, $Gate_1$ and $Gate_4$ are far away from each other, their parameters can be assumed as weakly correlated or uncorrelated (i.e. when $Gate_1$ has a larger than nominal gate channel length, the gate channel length for $Gate_4$ may be either larger or smaller than nominal).

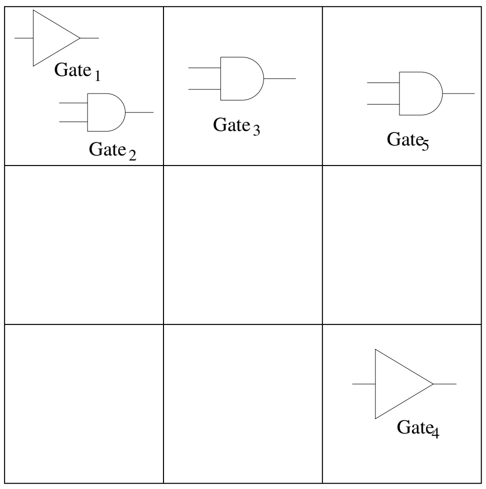

Figure 2.1: Grid based model for spatial correlations

With the grid-based model, we can use a single random variable $p(x, y)$ to model a parameter variation in a single grid at location $(x, y)$. As a result, $n$ random variables are needed for each type of parameter, where each represents the value of a parameter in one of the $n$ grids. In addition, we assume that correlation only exists among the same type of parameters in different grids (this assumption is not critical and can easily be removed). For example, gate length $L$ for transistors in the $i$-th grid are correlated with those in nearby grids, but are uncorrelated with other parameters such as gate oxide thickness $T_{ox}$ in any grid including the $i$-th grid itself. For each type of parameter, a correlation matrix $\Sigma$ of size $n \times n$ represents the spatial correlations of this parameter. Notice that the number of grid partitions needed is determined by the process, not the circuit. So we can apply the same correlation model to different designs under the same process.

## 2.1.2 Variable decoupling and reduction

Due to the large number of random variables involved in VLSI design, we should reduce the number of variables by exploiting the spatial correlations of the given process variations.

Also, the spectral stochastic method in 2.2, starts with independent random variables as the input of the spectral stochastic method. Since the random variables are correlated, this correlation should be removed before using the spectral stochastic method. We first present following result as our theoretical basis for decoupling the correlation of those variables [84].

**Proposition 2.1.1.** *For a set of zero-mean Gaussian distributed variables $\xi^*$ whose covariance matrix is $\Omega$, if there is a matrix $L$ satisfying $\Omega = LL^T$ , then $\xi^*$ can be represented by a set of independent standard normal distributed variables $\xi$ as $\xi^* = L\xi$.*

*Proof.* According to the characteristics of normal distribution, linear transformation does not impact on the zero mean of the variables, and yield another normal distribution. Thus we only need to prove the covariance matrix remains unchanged during the transformation. According to the definition of covariance,

$$\mathrm{cov}(L\xi) = E(L\xi(L\xi)^T) = LE(\xi\xi^T)L^T, \tag{2.3}$$

where $E(x)$ means the mean value of $x$. Since $\xi$ is subject to standard normal distribution,

$$LE(\xi\xi^T)L^T = LL^T = \Omega \tag{2.4}$$

The techniques for decoupling and reducing the number of variables consists of two different types: principal factor analysis (PFA) and principal component analysis (PCA). Which will be introduced in the following part.

### 2.1.3 Principle factor analysis technique

Note that the solution for decoupling is not unique. For example, Cholesky decomposition can be used to seek $L$ since the covariance matrix $\Omega$ is always a semi-positive definite matrix. However Cholesky decomposition cannot reduce the number of variables. Principle factor analysis (PFA) [28] can substitute Cholesky decomposition when variable reduction is needed. Eigen-decomposition on the covariance matrix yields:

$$\Omega = LL^T,\ L = (\sqrt{\lambda_1}e_1, ..., \sqrt{\lambda_n}e_n), \tag{2.5}$$

where $\{\lambda_i\}$ are eigenvalues in order of descending magnitude, and $\{e_i\}$ are corresponding eigenvectors. PFA reduces the number of components in $\xi$ by truncating $L$ using the first $k$ items.

The error of PFA can be controlled by $k$:

$$err = \frac{\sum\limits_{i=k+1}^{n} \lambda_i}{\sum\limits_{i=1}^{n} \lambda_i}, \tag{2.6}$$

where bigger $k$ leads to a more accurate result. PFA is efficient, especially when the correlation length is large. In our experiments, we set the correlation length being 8 times of width of wires. As a result, PFA can reduce the number of variables from 40 to 14 with an error of about 1% in an example with 20 parallel wires.

### 2.1.4 Weighted PFA technique

Principle factor analysis is another variable reduction technique, which can handle the spatial correlation. One idea is to consider the importance of the outputs during the reduction process. The recently proposed weighted PFA (wPFA) technique [83] is followed here to seek better variable reduction efficiency.

If a weight is defined for each physical variable $\xi_i$, to reflect its impact on the output, then a set of new variables $\xi^*$ are formed:

$$\xi^* = W\xi \tag{2.7}$$

where $W = diag(w_1, w_2, ..., w_n)$ is a diagonal matrix of weights. As a result, the covariance matrix of $\xi^*$, $\Omega(\xi^*)$, now contains the weight information and performing PFA on $\Omega(\xi^*)$ leads to the weighted variable reduction. Specifically, we have

$$\Omega(\xi^*) = E(W\xi(W\xi)^T) = W\Omega(\xi)W^T \tag{2.8}$$

and denote its eigenvalues and eigenvectors by $\lambda_i^*$ and $e_i^*$. Then, the variables $\xi$ can be approximated by the linear combination of a set of independent dominant variables $\zeta^*$:

$$\xi = W^{-1}\xi^* \approx W^{-1}\sum_{i=1}^{k}\sqrt{\lambda_i^*}e_i^*\zeta_i^*. \tag{2.9}$$

The error controlling process is similar to (2.6), but using the weighted eigenvalues $\lambda_i^*$.

### 2.1.5 Principal component analysis technique

We first briefly review the concept of principal component analysis (PCA), which is used here to transform the random variables with correlation to uncorrelated random variables [29]. Then the difference between PFA and PCA will be discussed.

Suppose that $x$ is a vector of $n$ random variables, $x = [x_1, x_2, ..., x_n]^T$, with covariance matrix $\Omega$ and mean vector $\mu_x = [\mu_{x_1}, \mu_{x_2}, ..., \mu_{x_n}]$. To find the orthogonal random variables, we first calculate the eigenvalue and corresponding eigenvector. Then, by ordering the eigenvectors in descending order eigenvalues, the orthogonal matrix $A$ will be obtained. Here, $A$ is expressed as

$$A = [e_1^T, e_2^T, ..., e_n^T]^T \tag{2.10}$$

where $e_i$ is the corresponding eigenvector to eigenvalue $\lambda_i$, which satisfies

$$\lambda_i e_i = \Omega e_i, i = 1, 2, ..., n \tag{2.11}$$

and

$$\lambda_i < \lambda_{i-1},\ \ i = 2, 3, ..., n \tag{2.12}$$

With $A$, we can perform the transformation to get orthogonal random variables $y$, $y = [y_1, y_2, ..., y_n]^T$ by using

$$y = A(x - \mu_x) \tag{2.13}$$

where, $y_i$ is a random variable with Gaussian distribution. The mean, $\mu_{y_i}$, is 0 and the standard deviation, $\sigma_{y_i}$, is $\sqrt{\lambda_i}$ on the condition that [29]

$$e_i^T e_i = 1, i = 1, 2, ..., n \tag{2.14}$$

Here, because of the orthogonal property of matrix A

$$A^{-1} = A^T \tag{2.15}$$

To reconstruct the original random variables, we use the following equation:

$$x = A^T y + \mu_x \tag{2.16}$$

PCA and PFA has two major difference. Firstly, the diagonal entries of correlation matrix in PCA are unities, which means that the variance of every variable involving in this analysis is one. Therefore, the "factors" defined in PCA are called "components", since the variables are already normalized. By contract, the diagonal elements of the original correlation matrix are estimated communalities in PFA. Second, the process of factor extraction is different. In PCA, the factor loadings are extracted form the ones in the diagonals. On the other hand, PFA utilizes an iterative method to refine estimates of the communalities to some predefined level of accuracy. After each step, these estimated values are placed into the diagonal of correlation matrix. In most cases, these two methods usually yield very similar results. However, PFA is often preferred as a classification method to detect structure, while PCA is often preferred as a method when the goal of analysis is to reduce the number of variables. More detailed discussion can be found in [23].

## 2.2 Sum of random variables

### 2.2.1 Monte Carlo method

Monte Carlo (MC) techniques [17] can be used to estimate the value of a definite, finite-dimensional integral of the form

$$G = \int_{\mathbb{S}} g(X) f(X) dX, \tag{2.17}$$

where $\mathbb{S}$ is a finite domain and $f(X)$ is a probability density function (PDF) over $X$, i.e., $f(X) \geq 0$ for all $X$ and $\int_{\mathbb{S}} f(X)dX = 1$. We can accomplish the Monte Carlo estimation for the value of $G$ by drawing a set of independent samples $X_1, X_2, ..., X_{MC}$ from $f(X)$ and by applying

$$G_{MC} = (1/MC) \sum_{i=1}^{MC} g(X_i). \tag{2.18}$$

The estimator $G_{MC}$ above is a random variable. Its mean value is the integral $G$ we are trying to estimate, i.e., $E(G_{MC}) = G$, making it an unbiased estimator. The variance of $G_{MC}$ is $Var(G_{MC}) = \sigma^2/MC$, where $\sigma^2$ is the variance of the random variable $g(X)$ given by

$$\sigma^2 = \int_{\mathbb{S}} g^2(X) f(X) dX - G^2. \tag{2.19}$$

We can use the standard deviation of $G_{MC}$ to assess its accuracy in estimating $G$. If the sample number $MC$ is sufficiently large, then by the Central Limit Theorem, $\frac{G_{MC}-G}{\sigma/\sqrt{MC}}$ has an approximate standard normal distribution($N(0,1)$). Hence,

$$P\left(G - 1.96\frac{\sigma}{\sqrt{MC}} \leq G_{MC} \leq G + 1.96\frac{\sigma}{\sqrt{MC}}\right) \approx 0.95, \tag{2.20}$$

where $P$ is the probability measure. (2.20) shows that $G_{MC}$ will be in the interval $\left[G - 1.96\frac{\sigma}{\sqrt{MC}}, G + 1.96\frac{\sigma}{\sqrt{MC}}\right]$ with 95% confidence. Thus, one can use the error measure

$$|Error| \approx \frac{2\sigma}{\sqrt{MC}} \tag{2.21}$$

in order to assess the accuracy of the estimator.

### 2.2.2 Spectral stochastic method using stochastic orthogonal polynomial chaos

One recent advance in stochastic analysis is to apply stochastic orthogonal polynomial chaos (PC) [74] to the nanometer scale integrated circuit analysis. Based on the *Askey scheme* [78], any stochastic random variable can be represented by orthogonal polynomial chaos, and the random variable with different probability distribution type is associated with different type of orthogonal polynomials.

Hermite polynomial chaos (Hermite PC or HPC) utilizes a series of orthogonal polynomials (with respect to the Gaussian distribution) to facilitate stochastic analysis [79]. These polynomials are used as the orthogonal base to decompose a random process in a similar way that sine and cosine functions are used to decompose a periodic signal in a Fourier series expansion. Note that for the Gaussian and log-normal distributions, Hermite polynomial is the best choice as they lead to exponential convergence rate [20]. For non Gaussian and non log-normal distributions, there are other orthogonal polynomials such as Legendre for uniform distribution, Charlier for Poisson distribution and Krawtchouk for Binomial distribution etc [19, 74].

For a random variable $y(\xi)$ with limited variance, where $\xi = [\xi_1, \xi_2, ...\xi_n]$ is a vector of zero mean orthogonal Gaussian random variables. The random variable can be approximated by truncated Hermite PC expansion as follows [20]:

$$y(\xi) = \sum_{k=0}^{P} a_k H_k^n(\xi) \tag{2.22}$$

where $n$ is the number of independent random variables, $H_k^n(\xi)$ is $n$-dimensional Hermite polynomials and $a_k$ are the deterministic coefficients. The number of terms $P$ is given

$$P = \sum_{k=0}^{p} \frac{(n-1+k)!}{k!(n-1)!} \tag{2.23}$$

where $p$ is the order of the Hermite PC.

Similarly, a random process $v(t,\xi)$ with limited variance can be approximated as

$$v(t,\xi) = \sum_{k=0}^{P} a_k H_k^n(\xi) \tag{2.24}$$

If only one random variable/process is considered, the one-dimensional Hermite polynomials are expressed as follows:

$$H_0^1(\xi) = 1, H_1^1(\xi) = \xi, H_2^1(\xi) = \xi^2 - 1, H_3^1(\xi) = \xi^3 - 3\xi, ... \tag{2.25}$$

Hermite polynomials are orthogonal with respect to Gaussian weighted expectation (the superscript $n$ is dropped for simple notation):

$$< H_i(\xi), H_j(\xi) >=< H_i^2(\xi) > \delta_{ij} \tag{2.26}$$

where $\delta_{ij}$ is the Kronecker delta and $< *, * >$ denotes an inner product defined as follow:

$$< f(\xi), g(\xi) >= \frac{1}{\sqrt{(2\pi)^n}} \int f(\xi) g(\xi) e^{-\frac{1}{2}\xi^T\xi} d\xi \tag{2.27}$$

Like Fourier series, the coefficient $a_k$ for random variable $y$ and $a_k(t)$ for random process $v(t)$ can be found by a projection operation onto the HPC basis:

$$\begin{aligned} a_k &= \frac{< y(\xi), H_k(\xi) >}{< H_k^2(\xi) >}, && (2.28) \\ a_k(t) &= \frac{< v(t,\xi), H_k(\xi) >}{< H_k^2(\xi) >}, \ \forall k \in \{0, ..., P\}. && (2.29) \end{aligned}$$

Once we obtain the Hermite PC, we can obtain the mean and variance of random variable $y(\xi)$ trivially as (one Gaussian variable case):

$$\begin{aligned} E(y(\xi)) &= y_0 \\ Var(y(\xi)) &= y_1^2 Var(\xi_1) + y_2^2(t) Var(\xi_1^2 - 1) \\ &= y_1^2 + 2y_2^2 && (2.30) \end{aligned}$$

Similarly, for random process $v(t, \xi)$ (one Gaussian variable case), the mean and variance are as follows

$$\begin{aligned} E(v(t,\xi)) &= v_0(t) \\ Var(v(t,\xi)) &= v_1^2(t) Var(\xi_1) + v_2^2(t) Var(\xi_1^2 - 1) \\ &= v_1^2(t) + 2v_2^2(t) && (2.31) \end{aligned}$$

One critical problem remains so far is how to obtain the coefficients of Hermite PC in (2.28) and (2.29) efficiently. There are two group of techniques to calculate the coefficients of Hermite PC in (2.28) and (2.29) efficiently, which are collocation-based spectral stochastic method and Galerkin-based spectral stochastic method, or collocation-based and Galerkin-based methods in short.

### 2.2.3 Collocation-based spectral stochastic method

The Gaussian quadrature method is an efficient numerical method for computing the definite integral of a function [26]. Using this method, we can compute the coefficients $a_k$ and $a_k(t)$ in (2.28) and (2.29), respectively. Next, we will review this method, which uses the Hermite polynomial shown below.

Our goal is to determine the numerical solution to the integral equation $< y(\xi), H_j(\xi) >$ ($x$ can be a random variable or random process). In our problem, this is a one-dimensional numerical quadrature problem based on Hermite polynomials [26]. Thus, we have

$$\begin{aligned} < y(\xi), H_k(\xi) > &= \frac{1}{\sqrt{(2\pi)}} \int y(\xi) H_k(\xi) e^{-\frac{1}{2}\xi^2} d\xi \\ &\approx \sum_{i=0}^{P} y(\xi_i) H_i(\xi_i) w_i \end{aligned} \tag{2.32}$$

Here we have only a single random variable $\xi$. $\xi_i$ and $w_i$ are Gaussian-Hermite quadrature abscissas (quadrature points) and weights.

The Quadrature rule states that if we select the roots of the $P^{th}$ Hermite Polynomial as the quadrature points, the quadrature is exact for all polynomials of degree $2P-1$ or less for (2.32). This is called ($P$-1)-level accuracy of the Gaussian-Hermite quadrature.

For multiple random variables, a multi-dimensional quadrature is required. The traditional way of computing a multi-dimensional quadrature is to use a direct tensor product based on one dimensional Gaussian Hermite quadrature abscissas and weights [54]. With this method, the number of quadrature points needed for $n$-dimensions at level $P$ is about $(P+1)^n$, which is well known as the curse-of-dimensionality.

Smolyak quadrature [54], also known as sparse grid quadrature, is used as an efficient method to reduce the number of quadrature points. Let us define a one-dimensional sparse grid quadrature point set $\Theta_1^P = \{\gamma_i, \gamma_2, ..., \gamma_P\}$, which uses $P+1$ points to achieve degree $2P+1$ of exactness. The sparse grid for an $n$-dimensional quadrature at degree $P$ chooses points from the following set:

$$\Theta_n^P = \bigcup_{P+1 \le |\vec{i}| \le P+n} (\Theta_1^{i_1} \times ... \times \Theta_1^{i_n}) \tag{2.33}$$

where $|\vec{i}| = \sum_{j=1}^{n} i_j$. The corresponding weight is:

$$w_{j_{i_1} \cdots j_{i_n}}^{i_1 \cdots i_n} = (-1)^{P+n-|\vec{i}|} \begin{pmatrix} n-1 \\ n+P-|\vec{i}| \end{pmatrix} \prod_m w_{j_{i_m}}^{i_m} \tag{2.34}$$

where $\begin{pmatrix} n-1 \\ n+P-|\vec{i}| \end{pmatrix}$ is the combinatorial number and $w$ is the weight for the corresponding quadrature points. It has been shown that interpolation on a Smolyak grid ensures a bound for the mean-square error [54]

$$|E_P| = O(N_P^r (logN_P)^{(r+1)(n-1)}),$$

where $N_P$ is the number of quadrature points and $k$ is the order of the maximum derivative that exist for the delay function. The number of quadrature points increases as $O(\frac{n^P}{(P)!})$.

It can be shown that a sparse grid of at least level $P$ is required for an order $P$ representation. The reason is that the approximation contains order $P$ polynomials for both $y(\xi)$ and $H_j(\xi)$. Thus, there exists $y(\xi)H_j(\xi)$ with order $2P$, which requires a sparse grid of at least level $P$ with an exactness degree of $2P+1$.

Therefore, level 1 and level 2 sparse grids are required for linear and quadratic models, respectively. The number of quadrature points is about $2n$ for the linear model, and $2n^2$ for the quadratic model. The time cost is about the same as the Taylor-conversion method, while keeping the accuracy of homogenous chaos expansion.

In addition to the sparse grid technique, we also employ several accelerating techniques. Firstly, when $n$ is too small, the number of quadrature points for sparse grid may be larger than that of direct tensor product of a Gaussian quadrature. For example, if there are only 2 variables, the number is 5 and 15 for level 1 and 2 sparse grid, compared to 4 and 9 for direct tensor product. In this case, the sparse grid will not be used. Secondly, The set of quadrature points (2.33) may contain the same points with different weights. For example, the level 2 sparse grid for 3 variables contain 4 instances of the point (0,0,0). Combining these points by summing the weights reduces the computational cost of $y(\vec{\gamma}_i)$.

### 2.2.4 Galerkin-based spectral stochastic method

The Galerkin-based method is using the principle of orthogonality that the best approximation of $y(\xi)$ is obtained when the error, $\Delta(\xi)$, defined as

$$\Delta(\xi) = y(\xi) - y \tag{2.35}$$

is orthogonal to the approximation. That is

$$< \Delta(\xi), H_k(\xi) >= 0, \; k = 0, 1, \ldots, P, \tag{2.36}$$

where $H_k(\xi)$ are Hermite polynomials. In this way, we have transformed the stochastic analysis process into a deterministic form, whereas we only need to compute the corresponding coefficients of the Hermite PC.

For the illustration purpose, considering two Gaussian variable $\xi = [\xi_1, \xi_2]$, we assume that the charge vector in panels can be written as a second order ($p = 2$) Hermite PC, we have

$$\begin{aligned} y(\xi) &= y_0 + y_1\xi_1 + y_2\xi_2 + y_3(\xi_1^2 - 1) + \\ &\quad y_4(\xi_2^2 - 1) + y_5(\xi_1\xi_2). \end{aligned} \tag{2.37}$$

which will be solved by (2.36). Once the Hermite PC of $y(\xi)$ is known, the mean and variance of $y(\xi)$ can be evaluated trivially. Given an example, for one random variable, the mean and variance are calculated as:

$$\begin{aligned} E(y(\xi)) &= y_0 \\ Var(y(\xi)) &= y_1^2 Var(\xi) + y_2^2 Var(\xi^2 - 1) \\ &= y_1^2 + 2y_2^2. \end{aligned} \tag{2.38}$$

In consideration of correlations among random variables, we apply principal component analysis (PCA) 2.1.5 to transform the correlated variables into a set of independent variables.

## 2.3 Sum of log-normal random variables

Due to the exponential convergence rate, Hermite PC can be used to represent log-normal variables and the sum of log-normal variables [42].

### 2.3.1 Hermite PC representation of log-normal variables

Let $g(\xi)$ be the Gaussian random variable, and $l(\xi)$ be the random variable obtained by taking the exponential of $g(\xi)$,

$$l(\xi) = e^{g(\xi)}, \;\; g(\xi) = ln(l(\xi)) \tag{2.39}$$

For a log-normal random variable $I_l$, let the mean and the variance of $g(\xi)$ as $\mu_g$ and $\sigma_g^2$, then the mean and variance of $l(\xi)$ are

$$\begin{aligned} \mu_l &= e^{(\mu_g + \frac{\sigma_g^2}{2})} && (2.40) \\ \sigma_l^2 &= e^{(2\mu_g + \sigma_g^2)}[e^{\sigma_g^2} - 1] && (2.41) \end{aligned}$$

respectively.

A general Gaussian variable $g(\xi)$ can always be represented in the following affine form:

$$g(\xi) = \sum_{i=0}^{n} \xi_i g_i \tag{2.42}$$

where $\xi_i$ are orthogonal Gaussian variables. i.e. $< \xi_i \xi_j >= \delta_{ij}$, $< \xi_i >= 0$ and $\xi_0 = 1$ and $g_i$ is the coefficient of the individual Gaussian variables. Note that such form can always be obtained by using Karhunen-Loeve orthogonal expansion method [20]

In our problem, we need to represent the log-normal random variable $l(\xi)$ by using the Hermite PC expansion form:

$$l(\xi) = \sum_{k=0}^{P} l_k H_k^n(\xi) \tag{2.43}$$

where $l_0 = \exp[\mu_g + \frac{\sigma_g^2}{2}]$. To find the other coefficients, we can apply (2.28) on $l(\xi)$.

Therefore, we have

$$l_k(t) = \frac{< l(t,\xi), H_k(\xi) >}{< H_k^2(\xi) >}, \ \forall k \in \{0, ..., P\}. \tag{2.44}$$

It was shown in [19], $l(\xi)$ can be written as

$$l(\xi) = \frac{< H_k(\xi - \mathbf{g}) >}{< H_i^2(\xi) >} = \exp[\mu_g + \frac{1}{2}\sum_{j=1}^{n} g_j^2] \tag{2.45}$$

where $n$ is the number of independent Gaussian random variables.

The log-normal process can then be written as

$$l(\xi) = l_0(1 + \sum_{i=1}^{n} \xi_i g_i + \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{(\xi_i\xi_j - \delta_{ij})}{< (\xi_i\xi_j - \delta_{ij})^2 >} g_i g_j + ...) \tag{2.46}$$

where $g_i$ is defined in (2.42).

### 2.3.2 Hermite PC representation with one Gaussian variable

In this case, $\xi = [\xi_1]$. For the second order Hermite PC ($P = 2$), following (2.46), we have

$$l(\xi) = l_0(1 + \sigma_g \xi_1 + \frac{1}{2}\sigma_g^2(\xi_1^2 - 1)) \tag{2.47}$$

Hence, the desired Hermite PC coefficients, $I_{0,1,2}$, can be expressed as $l_0, l_0\sigma_g$ and $\frac{1}{2}l_0\sigma_g^2$ respectively.

### 2.3.3 Hermite PC representation of two and more Gaussian variables

For two random variables ($n = 2$), assume that $\xi = [\xi_1, \xi_2]$ is a normalized uncorrelated Gaussian random variable vector that represents random variable $g(\xi)$:

$$g(\xi) = \mu_g + \sigma_1\xi_1 + \sigma_2\xi_2 \tag{2.48}$$

Note that

$$< (\xi_i\xi_j - \delta_{ij})^2 >=< \xi_i^2\xi_j^2 >=< \xi_i^2 >< \xi_j^2 >= 1$$

Therefore, the expansion of the log-normal random variables using second order Hermite PCs can be expressed as

$$\begin{aligned} l(\xi) &= l_0(1 + \sigma_1\xi_1 + \sigma_2\xi_2 + \frac{\sigma_1^2}{2}(\xi_1^2 - 1) + \frac{\sigma_2^2}{2}(\xi_2^2 - 1) + \\ &\quad 2\sigma_1\sigma_2\xi_1\xi_2) \end{aligned} \tag{2.49}$$

where

$$\mu_l = l_0 = \exp(\mu_g + \frac{1}{2}\sigma_1^2 + \frac{1}{2}\sigma_2^2)$$

Hence, the desired Hermite PC coefficients, $I_{0,1,2,3,4,5}$, can be expressed as $l_0$, $l_0\sigma_1$, $l_0\sigma_2$, $\frac{1}{2}l_0\sigma_1^2$, $\frac{1}{2}l_0\sigma_2^2$, and $2l_0\sigma_1\sigma_2$ respectively.

Similarly, for four Gaussian random variables, assume that
$\xi = [\xi_1, \xi_2, \xi_3, \xi_4]$ is a normalized, uncorrelated Gaussian random variable vector. The random variable $g(\xi)$ can be expressed as

$$g = \mu_g + \sum_{i=1}^{4} \sigma_i\xi_i \tag{2.50}$$

As a result, the log-normal random variable $l(\xi)$ can be expressed as

$$l(\xi) = l_0(1 + \sum_{i=1}^{4} \xi_i \sigma_i + \sum_{i=1}^{4} \frac{1}{2}(\xi_i^2 - 1)\sigma_i^2 + \sum_{i=1}^{4}\sum_{j=1}^{4} \xi_i \xi_j \sigma_i \sigma_j + ...) \tag{2.51}$$

where

$$\mu_l = l_0 = \exp(\sigma_0 + \frac{1}{2}\sum_{i=1}^{4} \sigma_i^2)$$

Hence, the desired Hermite PC coefficients can be expressed using the equation (2.51) above.

## 2.4 Summary

To understand statistical analysis and modeling for nanometer VLSI design, some preliminary concepts in probability theory are necessary for the better comprehension of the following chapters. In this chapter, we have introduced the fundamentals of statistical analysis involved in VLSI design. First we have presented the basic concepts and components of covariance in process variation in VLSI design. After that, we reviewed the techniques for variable decoupling and reduction, following by different methods to estimate the sum of random variables. In leakage analysis of VLSI, log-normal distribution has very wide application. Therefore, we discussed the sum of log-normal random variables in detail at the end of this chapter.

# Chapter 3

# Statistical Leakage Power Analysis – Modeling and Previous Works

## 3.1 Introduction

As VLSI technology scales down into the nanometer regime, power consumption and corresponding thermal effects have become the first tier limiting factor threatening the continuous integration and performance improvement of integrated systems. Dynamic and leakage power are the two main sources of power consumption in VLSI circuits. In many high-performance designs, leakage component now is comparable to the dynamic component. Reports indicate that over 40% of total power consumption in 90-nm process technology comes from leakage [58]. This situation will become worse as technology scales and the impacts of the uncertainties from process variability and temperature environments on transistor leakage currents become more significant [1].

Process-induced variability has a huge impact on circuit performance in the sub-90nm VLSI technologies [50]. This is the particular case for leakage power. As a result, leakage variations become significant, and traditional worst case-based approach will lead to extremely pessimistic and expensive design solutions. Statistical estimation and analysis of leakage power considering process variability are critical in various chip design steps to improve design yield and robustness. A block diagram of the Statistical Leakage Analysis (SLA) flow is shown in Fig. 3.1. In the leakage estimation model, we can obtain the chip-level leakage statistics such as the mean value and standard deviation from 1) process information, 2) library information and 3) design information. To reduce leakage, an optimization approach has been proposed for standard cell based designs [63]. In this chapter, we also use a regular standard cell library for experimental results.

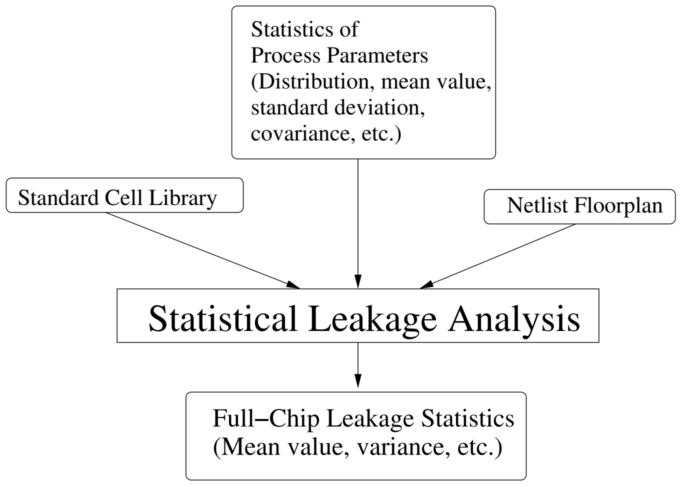
Statistics of Process Parameters (Distribution, mean value, standard deviation, covariance, etc.)
Standard Cell Library
Netlist Floorplan
Statistical Leakage Analysis
Full−Chip Leakage Statistics (Mean value, variance, etc.)

Figure 3.1: Chip-level statistical leakage analysis flow.

Many methods have been proposed for the statistical model of chip-level leakage current. Early work in [68] gives the analytic expressions of mean value and variance of leakage currents of CMOS gates considering only subthreshold leakage. The method in [49] provides simple analytic expressions of leakage currents of the whole chip considering global variations only. The method in [75] uses third order Hermite polynomials without considering spatial correlations, and only calculates the mean value of full-chip leakage current.

In [46], reverse biased source/drain junction band-to-band tunneling (BTBT) leakage current is considered, in addition to the subthreshold leakage currents, for estimating the mean values and variances of the leakage currents of gates only. In [60], the probability density function (PDF) of stacked CMOS gates and the entire chip are derived considering both inter-die and intra-die variations. In [10], a hardware-based statistical model of dynamic switching power and static leakage power was presented, which was extracted from experiments in a pre-determined process window.

Recently, a full-chip SLA method considering spatial correlations in the intra-die and inter-die variations was proposed [9]. This method introduces a grid-based partitioning of the circuits to reduce the number of variables at a loss of accuracy. A projection based approach has been proposed in [35] to speed up the leakage analysis, where Krylov subspace-based reduction has been performed on the coefficient matrices of second order expressions. This method assumes independent random variables after a pre-processing step such as principal components analysis (PCA). However, owing to the large number of random variables involved ($10^3$ to $10^6$), the PCA based pre-process can be very expensive. Work in [22] proposes a linear time complexity method to compute the mean and variance of full-chip leakage currents by exploiting the symmetric property of one existing exponential spatial correlation formula. The method only considers subthreshold leakage and it requires the chip cells and modules to be partitioned into a regular grid with similar uniform fitting functions, which is typically impractical.

Recent work in [6] presents a unified approach for statistical timing and leakage current analysis using quadratic polynomials. However, this method only considers the long-channel effects and ignores the short-channel effects (ignoring channel length variables) for the gate leakage models. The coefficients of the orthogonal polynomials at gate-level are computed directly by the inter-production via the efficient Smolyak quadrature method. The method also tries to reduce the number of variables via the moment matching method, which further speeds up the quadrature process at the cost of more errors.

In [64], Shen et al. propose a general full-chip leakage modeling and analysis approach, which is a gate-based spectral method and uses PCA to reduce the number of variables with much less accuracy loss (assuming that the geometrical variables are Gaussian. For non-Gaussian variables, independent component analysis (ICA) [24] can be used). This method considers both inter-die and intra-die variations and it can work with various spatial correlations. The proposed method becomes linear under strong spatial correlations. Unlike the existing approaches as presented in [9, 22], [64] does not make any assumptions about the distributions of final total leakage currents for both gates and chips, and does not require any grid-based partitioning of the chip.

Chip-level SLA methods can be classified into different categories based on different criteria as shown in Table. 3.1. Our classification and survey may not be complete as this is still an active research field and more efficient methods will be developed in the future. We will present in detail some recent important developments in the section such as Monte Carlo method and the traditional grid-based method [9]. The gate-based spectral stochastic method [64] and the virtual grid-based method will be introduced in Chapter 4 and Chapter 5, respectively. We remark that our limited coverage of the other methods, which are presented in minimal detail, does not diminish the value of their contributions.

Table 3.1: Different methods for full-chip SLA.

| Criteria | Categories | | | |
|---|---|---|---|---|
| Process variation | Inter-die | Intra-die, spatial correlated and non-correlated | | |
| Leakage distribution | Log-normal | Non-log-normal | | |
| Speed-up method | Monte Carlo | Grid-based | Gate-based | Projection based |
| Leakage component | $I_{sub}$ | $I_{gate}$ | | |
| Static leakage model | Gate-based | MOSFET-based | | |

This chapter is structured as follows. In Section 3.2, we discuss the leakage model for one gate without consideration of variation in process parameters first, then Section 3.3 gives the process variation models for computing statistical information of full-chip leakage current. Afterwards, Section 3.4 presents the recently proposed chip-level statistical leakage modeling and analysis work. The chapter concludes with a summary and brief discussion of potential open research problems.

## 3.2 Static leakage modeling

There are several different leakage mechanisms that contribute to the total leakage in a device. Among them, the three major factors can be identified as: subthreshold leakage, gate oxide leakage and junction tunneling leakage [45]. Here we describe the empirical curve-fitting models for them, which will be used for leakage current under process variations in the next section. The leakage current can be modeled based on gates or devices. In this section, we will show both kinds of leakage modeling methods.

### 3.2.1 Subthreshold and gate oxide leakage modeling

The subthreshold leakage current, $I_{sub}$, is exponentially dependent on the threshold voltage, $V_{th}$. $V_{th}$ is observed to be most sensitive to gate oxide thickness $T_{ox}$ and effective gate channel length $L$ due to short-channel effects. When the change in $L$ or $T_{ox}$ is small, the precise relationship shows an exponential dependent effect on $I_{sub}$, with the effect of $T_{ox}$ being relatively weak. For the gate oxide leakage current, both channel length and oxide thickness have strong impacts on the leakage currents, which are exponential functions of the two variables.

a) $I_{sub}$ and $I_{gate}$ model of one gate

The leakage model is based on gates, as in [9] and [64]. They follow the analytical expression estimating the subthreshold leakage currents as follows:

$$I_{sub} = e^{a_1 + a_2 L + a_3 L^2 + a_4 T_{ox}^{-1} + a_5 T_{ox}}, \tag{3.1}$$

$$I_{gate} = e^{a_1 + a_2 L + a_3 L^2 + a_4 T_{ox} + a_5 T_{ox}^2}, \tag{3.2}$$

where $a_1$ through $a_5$ are the fitting coefficients for each unique input combination of a gate. A look-up table (LUT) can be used to store the fitting coefficients. For a $k$-input gate, the size of the LUT is $2^k$ as we have two equations for each input combination. In [9], The authors only keep dominant states for the leakage current, i.e., only one "off" transistor in a series transistor stack. However, this is impractical with technology downscaling to 45nm. The $I_{sub}$ based on the model in (3.1) still has a large error compared to the simulation results. Hence, the authors in [64] keep all the states.

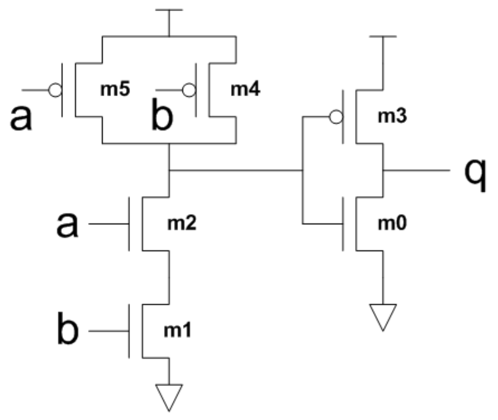

Figure 3.2: Schematic of the AND2 gate

Take the AND2 gate in Fig. 3.2 as an example. There are four different input patterns for the subthreshold leakage current, as shown in Fig. 3.3. In [9], the authors mentioned the "Dominant States", and assumed that only the leakage for dominant input patterns need to be considered. However, the AND2 gate is comprised of two sub-circuits, one NAND gate and one inverter, which they have different dominant input patterns. Actually, there is no significant difference between the four kinds of input patterns. Also, when the inputs are “01” or “10”, the leakage sources are NMOS gates. But when the input is “11”, the main contribution comes from the PMOS gates (as shown in Fig. 3.3). Therefore, in nano-scale technology, we need to measure the leakage currents for all input patterns. After choosing sampling points for $L$ and $T_{ox}$ in their $3\sigma$ regions linearly, and then conducting SPICE simulation at each point, the subthreshold leakage current is stored as the original curve. We can then perform the curve fitting process. Fig. 3.4 and Fig. 3.5 show the curve fitting results of $I_{sub}$ and $I_{gate}$ for four input patterns in the AND2 gate. Here, 100 points are chosen linearly in the $3\sigma$ regions for $L$ and $T_{ox}$. These figures show that the curves fit the SPICE results very well, and the currents in the four cases are comparable with each other. Since there is no “Dominant State”, all of them need to be considered.

After we obtain the analytic expression for each input combination, we take the average of the leakage currents of all the input combinations to arrive at a final analytic expression for each gate in lieu of the dominant states used in [9]. Based on this model, the leakage current of one gate under process variation can be estimated by lognormal distributions. The average leakage of a gate can be computed as a weighted sum of leakage under different input states,

$$I_{sub}^{avg} = \sum_{j \in input\ \ states} P_j I_{sub,j}, \tag{3.3}$$

$$I_{gate}^{avg} = \sum_{j \in input\ \ states} P_j I_{gate,j}, \tag{3.4}$$

$$I_{leak,chip} = \sum_{\forall gates\ \ i=1,\ldots,N} I_{sub,i}^{avg} + I_{gate,i}^{avg}, \tag{3.5}$$

where $P_j$ is the probability of input state $j$; $I_{sub,j}$ and $I_{gate,j}$ are the subthreshold leakage value and the gate oxide leakage value at input state $j$, respectively and $N$ is the total number of gates in the circuit. We ensure that the interaction between these two leakage mechanisms is included in the total leakage current estimation.

Since all the leakage components can be approximated as a lognormal distribution, we can simply sum up the distributions of the lognormals for all gates to get the full-chip leakage distribution. Note that there exist spatial correlations, and the leakage distributions of any two gates may be correlated. Therefore, the full-chip leakage current is calculated by a sum of correlated lognormals:

$$S = \sum_{i=1}^{p} e^{Y_i}, \tag{3.6}$$

where $p$ is the total number of lognormals to sum, $Y_i$ is Gaussian random variable, and $Y = [Y_1, Y_2, \ldots, Y_p]$ forms a multivariate normal distribution with covariance matrix $\Sigma_Y$. The vector $Y$ is a function of $L$ and $T_{ox}$.

If further speed-up is needed, we can consider only the dominant input states here, but it will lose some accuracy, especially for nano-scale technology.

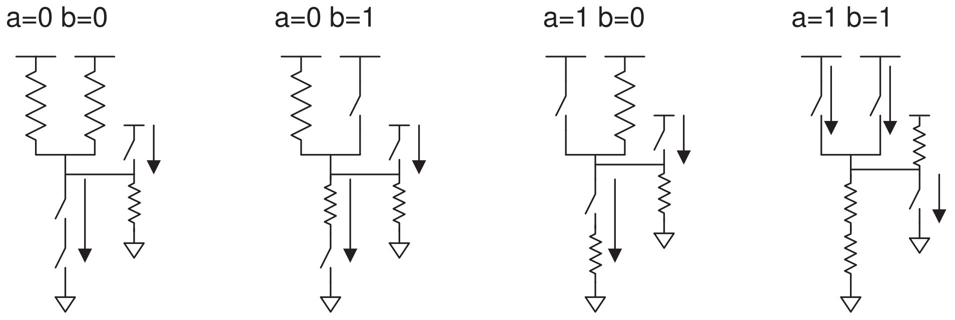

Figure 3.3: Four leakage cases (ab = "00" – Pattern 0, ab = "01" – Pattern 1, ab = "10" – Pattern 2, ab = "11" – Partern 3).

b) $I_{sub}$ and $I_{gate}$ model of a MOSFET

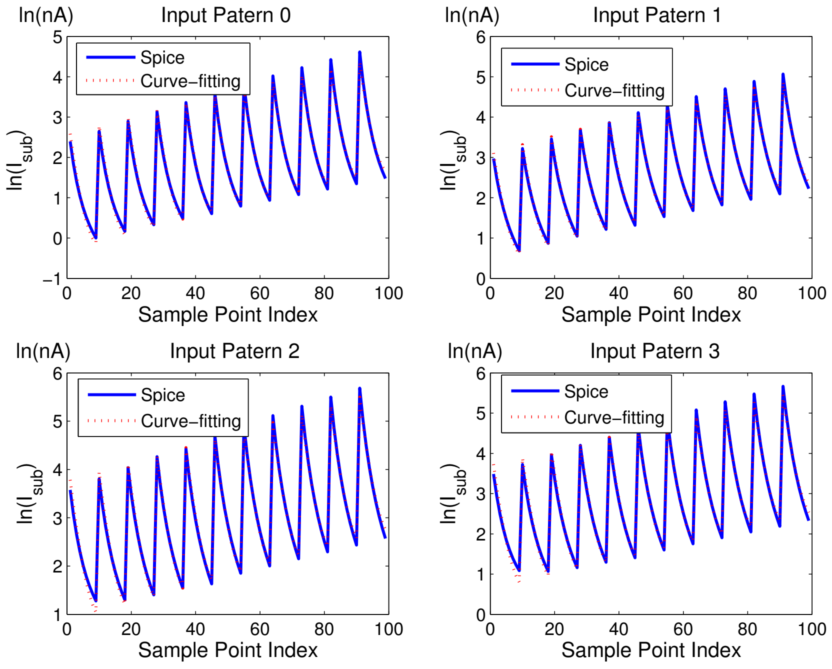

Figure 3.4: Subthreshold leakage currents for four different input patterns in AND2 gate under 45nm technology.

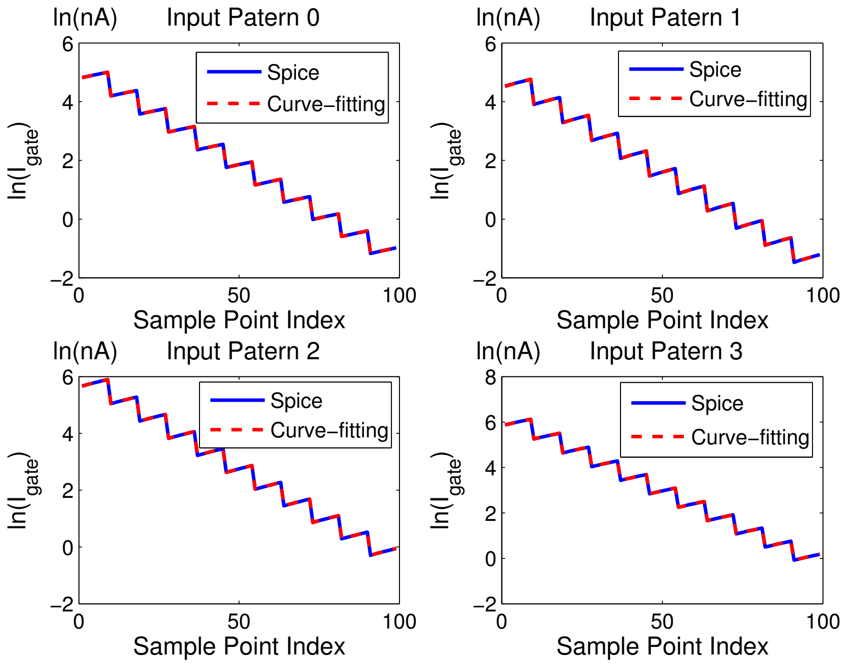

Figure 3.5: Gate oxide leakage currents for four different input patterns in AND2 gate under 45nm technology.

In [36], the authors formulated the statistical model for the subthreshold leakage current in a MOSFET. Here, we only discuss the formulation method developed for NMOS transistors, then, the method can be easily extended to PMOS transistors. In this model, we first set up the $L_{eff}$ (effective gate channel length) model. Afterwards, the $I_{sub}$ of one MOSFET can be formulated.

In nano-scale technology, the minimum feature size of a device is much smaller than the optical wavelength, which causes a severe distortion of the rectangular gate. Due to this, the leakage current ($I_{sub}$) increases significantly, while the current in the strong inversion region ($I_{on}$) is weakly affected [67].

In this part, let us translate an irregular gate structure into a single transistor of effective gate channel length $L_{eff}$ first. Fig. 3.6 shows a non-rectilinear gate. We can divide the non-rectilinear gate into slices of different lengths and same characteristic width $W_0$ along the width direction. In this way, the leakage current of the $i$-th non-rectilinear gate $I_{G,i}$ can be approximated as the sum of the leakage currents of all the slices along the width direction,

$$I_{G,i} = \sum_{j=1}^{M} I_j(L_j, W_0) = I(L_{eff}, W), \tag{3.7}$$

where $W$ is the width of the gate, and each slice can be considered as a regular gate. Table. 3.2 shows the expressions for modeling $L_{eff}$ , which can be used to include the distortion of a rectangular gate in any circuit simulation tool.

After we set up the $L_{eff}$ model, a compact model for $I_{sub}$ can be developed based on it. The leakage current is expressed as a function of $L_{eff}$ [22]. Layout parameters $SA$ and $SB$ are also employed to account for the stress effects of $I_{sub}$ (Fig. 3.7). By evaluating different values for these two parameters, asymmetrical stress effects of $I_{sub}$ are included. This method uses the stress liner technique [36], and could be

Table 3.2: Summary of the effective gate channel length model.

| Model Parameters | Model Expressions |
|---|---|
| M | Total number of slices along the width direction |
| $\mu$ | $\mu = \frac{\sum_{j=1}^{M} L_j}{M}$ |
| $\sigma$ | $\sigma = \frac{\sqrt{\sum_{j=1}^{M}(L_j-\mu)^2}}{M}$ |
| $\alpha$ | Fitting parameter |
| $L_{eff} = L_{min} + \alpha ln\left(\frac{\sigma W}{W_0}\right)$ | |

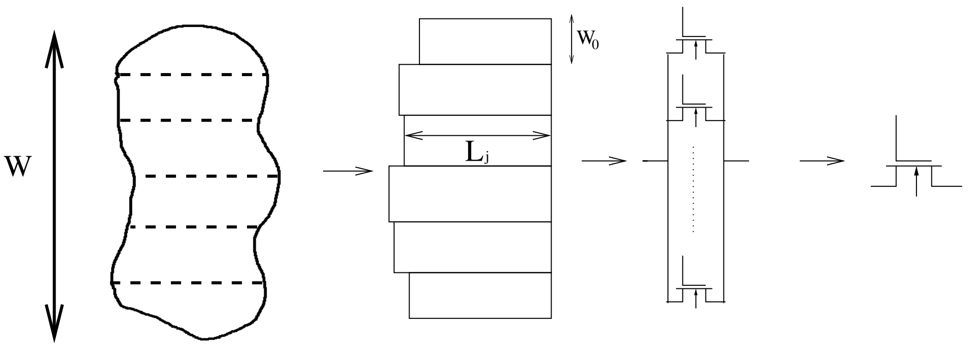

Figure 3.6: Procedure to derive the effective gate channel length model.

extended to include the stress effects induced by other techniques (e.g. SiGe, STI). The curve-fitted leakage model considering narrow width effect is shown in (3.8),

$$
\begin{aligned}
I_{sub} \quad = \quad & \frac{\alpha_{sub}\sqrt{q\epsilon_{si}N_{cheff}(W^2+\alpha_W W)}}{(V_{ds}^2+\alpha_{ds1}V_{ds}+\alpha_{ds2})exp(\alpha_{L1}L_{eff}^2+\alpha_{L2}L_{eff})} \\
& \times\left(1-exp\left(-\frac{V_{ds}}{V_T}\right)\right)\left(2-exp\left(-\frac{SA}{SA_0}\right)-exp\left(-\frac{SB}{SB_0}\right)\right) \\
& \times exp\left(\frac{V_{gs}-V_{thlin}}{nV_T}\right), \qquad (3.8)
\end{aligned}
$$

where all $\alpha$s are fitting parameters, $\epsilon_{si}$ is the dielectric constant of Si, and $N_{cheff}$ is the effective channel doping concentration.

In [36], the authors point out that $I_{gate}$ is less important than the other two components. Because high-k techniques are used to better insulate the gate from the channel for sub-65nm technologies, gate-oxide tunneling effect has been moderated

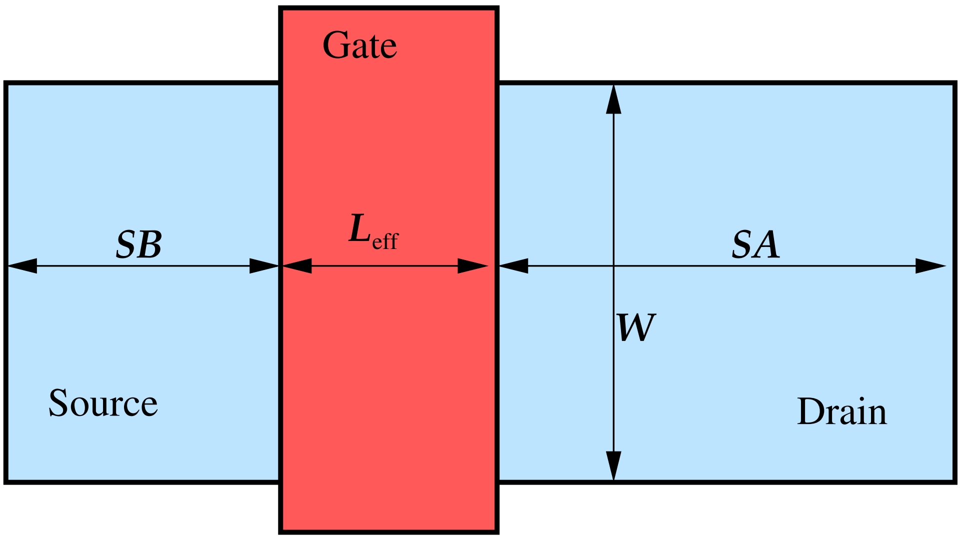

Figure 3.7: Typical layout of a MOSFET.

and controlled.

### 3.2.2 Junction tunneling leakage modeling

In nano-scale CMOS devices, Junction Tunneling Leakage (JTL) becomes an extremely important component in leakage current [45]. In [45] a JTL model requiring "rectangular junction" approximation has been proposed as a simplifying assumption, while the JTL model in [36] does not have that limit. Here, we give more details about the non-rectangular JTL model in [36]. This model includes both band-to-band tunneling (BBT) and trap-assistant tunneling (TAT) effects. Since the drain and source junctions have the same expression of JTL for symmetrical MOSFET, we only describe the drain junction model.

It is known that the tunneling of electrons through the bandgap contributes to the carrier transport in the PN junction in high electric fields. This electron-hole pair generation/recombination comes through direct or indirect BBT, and also through TAT. We can obtained the JTL current by an integration of the generation rates over the structure:

$$I_{junc} = q \iint (R_{BBT} + R_{TAT}) dx dy, \tag{3.9}$$

where $R_{BBT}$ is the generation rate from BBT contribution and $R_{TAT}$ is from TAT. By approximating the complex junction edge as a rectangular one [45], we can calculate the integral in (3.9) by using the average tunneling current density ($J_{junc}$). $J_{junc}$ is determined by the average electric field ($E_{junc}$) across the junction. Note that this approximation works only when the current density is almost the same everywhere along the PN junction. However, this is not the real case. The peak of current density changes greatly in nano-scale technologies, since we use the "halo" profile to reduce the depletion region width of the source-substrate and drain-substrate junctions. Many two-dimensional device simulations have been conducted in detail to describe the mechanism. It is shown that the JTL current density is dramatically changed when "halo" is applied, and the density peak occurs where the high doping region is implanted. There is nearly no current leaking through the bottom side of the junction.

The traditional junction leakage model cannot be used in SLA for another reason: $J_{junc}$ is very hard to express. In a previous work [45], $J_{junc}$ is modeled as a function of $N_{dside}$ and $N_{aside}$, where $N_{dside}$ and $N_{aside}$ represent the donor and acceptor doping concentration, respectively. In the real case, these two non-lithographic variation sources are very hard to characterize, since the actual dopant profile can not be measured directly. As a result, we need to employ new parameters.

In the modeling method in [36], the authors introduced two new parameters, $V_{thlin}$ and $V_{thsat}$ to express JTL and its variations. $V_{thlin}$ represents the channel doping effect, which is the subthreshold voltage measured when $V_{ds} = 0.1V$; and $V_{thsat}$ represents the doping effects in drain, which is the subthreshold voltage measured when $V_{ds} = V_{dd}$.

If the variation characterization procedure is feasible, these two parameters are easily measurable. Observing that by characterizing the fluctuations of $V_{thlin}$ and $V_{thsat}$, most of the JTL variations can be included, and the junction tunneling leakage model can be expressed as follows:

$$I_{junc} = W\alpha_{halo}(R_{BBT} + R_{TAT})\frac{V_{ds}^{3/2}}{E_g^{1/2}} \times exp\left(\frac{E_g^{3/2}}{\sqrt{V_{ds}}} + \frac{V_{thlin}}{\alpha_{thlin}} + \frac{V_{thsat}}{\alpha_{thsat}}\right), \qquad (3.10)$$

where $E_g$ is the band-gap, $\alpha_{halo}$ stands for the tunneling length induced by "halo" $\alpha_{thlin}$ and $\alpha_{thsat}$ are empirical parameters which can be easily extracted from the measured data. The reasons are: 1) when $V_{thsat}$ is a constant, $I_{junc}$ mostly depends on $V_{thlin}$; 2) when $V_{thlin}$ is a constant, $I_{junc}$ mostly depends on $V_{thsat}$.

## 3.3 Process variational models for leakage analysis

In this section, we present the process variation for computing variational leakage currents. Process variation occurs at different levels: wafer level, inter-die level, and intra-die level. Furthermore, they are caused by different sources such as lithography, materials, aging, etc [7]. Some of the variations are systematic, i.e., those caused by the lithography process [18, 55]. Some are purely random, i.e., the doping density of impurities and edge roughness [7]. In this section, we introduce different kinds of process variations first, and then the process variational model for leakage analysis.

The main process parameter to have a big impact on leakage current is the transistor threshold voltage $V_{th}$. $V_{th}$ is observed to be the most sensitive to the effective gate channel length $L$ and gate oxide thickness $T_{ox}$. The ITRS'08 [1] indicates that the gate channel length variation is a primary factor for device parameter variation, and the number of dopants in channel results in an unacceptably large statistical variation of the threshold voltage. Therefore, we must consider the variations in $L$ and $T_{ox}$, since leakage current is most sensitive to these parameters [9]. To reflect reality, we model spatial correlations in the gate channel length, while the gate oxide thickness values for different gates are taken to be uncorrelated.

Here we list an example of detailed parameters for gate channel length and gate oxide thickness variations for under 45nm technology in Table 3.3. As indicated in the second column, we can decompose each parameter variation into "inter-die" and "intra-die" variations. For intra-die variation, we further decompose it into with and without spatial correlation. In most cases, these variations can be modeled by Gaussian distributions ([71], [14]). The total variance ($\sigma^2$) is computed by summing up the variances of all components, since the sum of Gaussian distributions is still a Gaussian distribution.

Table 3.3: Process variation parameter breakdown for 45nm technology.

| | $\sigma^2$ Distribution | | $(\sigma)$ |
|---|---|---|---|
| Gate Length(L) | Inter-die | 20% | $4\% \times 18nm$ |
| | Intra-die | | |
| | $*$ Spatial Correlated | 80% | |
| Gate Oxide Thickness($T_{ox}$) | Inter-die | 20% | $4\% \times 1.8nm$ |
| | Intra-die | | |
| | $*$ Non-Correlated | 80% | |

Electrical measurements of a full wafer shows that the intra-die gate channel length variation has strong spatial correlation [18]. This implies that devices that are physically close to each other are more likely to be similar than those that are far apart. Therefore, the intra-die variation of gate channel lengths is modeled based on such kind of correlation. There are several different models that can represent this kind of spatial correlations. Take the exponential model [77] for instance,

$$\gamma(dis) = e^{-dis^2/\eta^2}, \tag{3.11}$$

where $dis$ is the distance between two panel centers and $\eta$ is the correlation length. The strong spatial correlation suggested by (3.11) can be used to speed up the calculation by grid based method or principal component analysis (for Gaussian distributions) or independent component analysis (for non-Gaussian distributions). Details will be given in the next section. For gate oxide thickness, $T_{ox}$, strong spatial correlation does not exist, therefore, we assume $T_{ox}$ of different gates are uncorrelated.

The last column of Table 3.3 shows the standard deviation ($\sigma$) of each variation. According to statistical theory regarding Gaussian distributions, 99% of the samples should fall in the range of $\pm 3\sigma$. According to [1], the physical gate channel length for high performance logic in 45nm technology will be 18nm, and the physical variation should be controlled within +/-12%. Therefore, we let $3\sigma$ be 12%, and a similar analysis can be done for $T_{ox}$.

For a gate/module in a chip with gate channel length $L$, and process variation $\Delta L$ using our model parameters in Table 3.3, we have

$$L = \mu_L + \Delta L, \Delta L = \Delta L_{inter} + \Delta L_{intra\_corr}, \tag{3.12}$$

where $\mu_L$ is the nominal design parameter value, and $\Delta L_{inter}$ is constant for all gates in all grids since it is a global factor that applies to the entire chip. For one chip sample, we only need to generate it once. $\Delta L_{intra\_corr}$ is different between each gate or each grid, and has spatial correlation. Therefore, we generate one value for each gate/grid, and the spatial correlation is regarded as an exponential model in (3.11), so that the correlation coefficient value diminishes with the distance between any two gates/grids.

As for the gate oxide thickness $T_{ox}$, using model parameters in Table 3.3, we have

$$T_{ox} = \mu_{ox} + \Delta T_{ox}, \Delta T_{ox} = \Delta T_{ox,inter} + \Delta T_{ox,intra\_uncorr}, \tag{3.13}$$

where $\mu_{ox}$ is the nominal design parameter value. Similarly to $\Delta L_{inter}$, $\Delta T_{ox,inter}$ is constant for all gates in all grids. $\Delta T_{ox,intra\_uncorr}$ is different between any gates/grids, but does not have spatial correlation.

After the process variations are modeled as correlated distributions, we can apply the Principal Component Analysis (PCA) in 2.1.5 to decompose correlated Gaussian distributions into independent ones. After PCA, the process variations (e.g., $\Delta V_{th}$, $\Delta T_{ox}$ and $\Delta L$) of each gate can be modeled as:

$$\Delta X_{G,i} = V_{G,i} E, \tag{3.14}$$

where the vector $\Delta X_{G,i} = [\Delta x_{G,i,1}, \Delta x_{G,i,2}, \ldots]^T$ stands for the parameter variations of the $i$-th gate. $E = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m]^T$ represents the random variables for modeling both inter-die and intra-die variations of the entire die. Here $\{\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_m\}$ can be extracted by PCA. They are independent and satisfy the standard Gaussian distribution (i.e., zero mean and unit standard deviation). $m$ is the total number of these random variables. For practical industry designs, $m$ is typically large (e.g., $10^3 \sim 10^6$). $V_{G,i}$ captures the correlations among the random variables.

When $m$ is a large number, the size of $V_{G,i}$ can be extremely huge. However, $X_{G,i}$ only depends on the intra-die variations within its neighborhood; so $V_{G,i}$ should be quite sparse. In Section 3.4, the gate-based spectral stochastic method and the projection-based method will use this sparsity property to reduce the computational cost in two different ways.

Gate-based statistical leakage analysis typically starts from the leakage modeling for one gate,

$$I_{G,i} = f(E), \tag{3.15}$$

where $I_{G,i}$ represents the total leakage current (including all the leakage components such as subthreshold leakage, gate oxide leakage and junction tunneling leakage) of the $i$-th gate. Different models can be chosen here to represent the relationship between $E$ and $I_{G,i}$. For example, quadratic models are used to guarantee accuracy,

$$\log(I_{G,i}) = E^T A_{G,i} E + B_{G,i}^T E + C_{G,i}, \tag{3.16}$$

where $A_{G,i} \in R^{m \times m}$, $B_{G,i} \in R^m$ and $C_{G,i} \in R$ are the coefficients. More details will be given in the next section.

Given the leakage models of all the individual gates, the full-chip leakage current is the sum of leakage currents of all the gates on the chip:

$$I_{leak,Chip} = I_{G,1} + I_{G,2} + \cdots + I_{G,N}, \tag{3.17}$$

where $N$ is the total number of gates in a chip. If we choose the quadratic model in (3.16), (3.17) implies that the full-chip leakage current is the sum of many lognormal distributions. As we mentioned before, it can be approximated as a lognormal distribution [9]. Therefore, we can also use a quadratic model to approximate the logarithm of the full-chip leakage:

$$\log(I_{leak,Chip}) = E^T A_{Chip} E + B_{Chip}^T E + C_{Chip}, \tag{3.18}$$

where $A_{Chip} \in R^{m \times m}$, $B_{Chip} \in R^m$ and $C_{Chip} \in R$ are the coefficients. In (3.16) and

(3.18), the quadratic coefficient matrices $A_{Gate_i}$ and $A_{Chip}$ can be extremely large for capturing all the intra-die variations, which makes the quadratic modeling problem extremely expensive in practical applications. Several approaches have been made to reduce the size of the model, with more details shown in the next section.

## 3.4 Previous works for full-chip leakage analysis

As shown in Table. 3.1, full-chip leakage modeling and analysis methods can be classified into different categories based on different criteria. In this section, we will present in detail several important methods such as the grid-based method, the gate-based spectral stochastic method and project-based methods.

### 3.4.1 Monte Carlo methods

Monte Carlo (MC) technique mentioned in Section 2.2.1 can be used to estimate the value of leakage power at gate-level as well as chip-level.

For full-chip leakage current, $I_{leak,Chip}$ is $G$ in (2.17). If the sample number $MC$ is large enough, then we can obtain a sufficiently accurate result. However, for full-chip leakage current analysis, the Monte Carlo estimator is too expensive. A more efficient method with good accuracy is needed.

Several techniques exist for improving the accuracy of Monte Carlo evaluation of finite integrals. In these techniques, the goal is to construct an estimator with a reduced variance for a given, fixed number of samples. In other words, the improved estimator can provide the same accuracy as the standard Monte Carlo estimator, while needing considerably fewer samples. This is desirable because computing the value of $g(X_i)$ is typically costly.

### 3.4.2 Grid-based methods

Since the number of gates on an entire chip is very large and every gate has their own variational parameter, the resulting number of random variables is very large. For greater efficiency, the grid-based method partitions a chip to several grids, and assigns all the gates on one grid with the same parameters.

The work in [9] shows a typical grid-based method considering both inter-die and intra-die variations (including the consideration of spatial correlation). It uses lognormal distribution to approximate the leakage current of each gate and the total chip leakage is determined by summing up the lognormals. In this work, both subthreshold leakage and gate oxide leakage of only dominant input states are considered in (3.4) and (3.5). Here we consider only intra-die variation of parameters. The extension to handling inter-die variation is quite obvious, as shown at the end of this subsection.

As shown in (3.6), the total leakage current of a chip is the sum of correlated leakage components, which can be approximated as a lognormal using Wilkinson's method [4]. A sum of $t$ lognormals, $S = \sum_{i=1}^{t} e^{Y_i}$, is approximated as the lognormal $e^Z$, where $Z = N(\mu_z, \sigma_z)$. In Wilkinson's approach, the mean value and standard deviation of $Z$ are obtained by matching the first two moments, $u_1$ and $u_2$ of $\sum_{i=1}^{t} e^{Y_i}$ as follows:

$$u_1 = E(S) = e^{\mu_z + \sigma_z^2/2} = \sum_{i=1}^{t} e^{\mu_{y_i} + \sigma_{y_i}^2/2}, \tag{3.19}$$

$$\begin{aligned} u_2 &= E(S^2) = e^{2\mu_z + 2\sigma_z^2} = \sum_{i=1}^{t} e^{2\mu_{y_i} + 2\sigma_{y_i}^2} + \\ &\quad 2\sum_{i=1}^{t-1} \sum_{j=i+1}^{t} e^{\mu_{y_i} + \mu_{y_j}} e(\sigma_{y_i}^2 + \sigma_{y_j}^2 + 2r_{ij}\sigma_{y_i}\sigma_{y_j})/2, \end{aligned} \tag{3.20}$$

where $r_{ij}$ is the correlation coefficient of $Y_i$ and $Y_j$ . Solving (3.20) for $\mu_z$ and $\sigma_z$

yields

$$\mu_z = 2\ln u_1 - \frac{1}{2}\ln u_2, \tag{3.21}$$

$$\sigma_z^2 = \ln u_2 - 2\ln u_1. \tag{3.22}$$

From the above formula, we can see that a pair-by-pair computation for all correlated pairs of variables needs to be done, i.e., for all $i$, $j$ such that $r_{ij} = 0$. It will lead to a very expensive computation time cost. First, leakage currents of different gates are correlated because of the spatial correlation of $L$. Secondly, $I_{sub}$ and $I_{gate}$ associated with the same NMOS transistor are correlated. Thirdly, $I_{sub}$ in the same transistor stack are also correlated. If there are $N$ gates in the circuit, the complexity for computing the sum will be $O(N^2)$, which is far from practical for large circuits. Therefore, the grid-based method uses several approximations to reduce the time complexity. In the grid-based method, gates in the same grid have the same parameter values. For example, let $I_{sub,i}$ be the subthreshold leakage currents for $Gate_i$ $(i = 1, \ldots, t)$ under the same input vector, and assume that these gates are all in the same grid $k$. Then

$$I_{sub,i} = \alpha_i e^{Y_i^0 + \beta_0 \cdot dL_k + \beta_1 \cdot dT_{ox,i}}, \tag{3.23}$$

where $\alpha_i$, $\beta_0$ and $\beta_1$ are the fitting coefficients. Since we assume that $L$ is spatially correlated and $T_{ox}$ is uncorrelated, all of the $I_{sub,i}$ in the same grid should use the same variable $dL_k$, and different $dT_{ox}$ values. Then, the sum of the leakage terms $I_{sub,i}$ in grid $k$ is given by:

$$e^{Y_i^0 + \beta_0 \cdot dL_k} \cdot \sum_{i=1}^{t} \alpha_i \cdot e^{\beta_1 \cdot dT_{ox,i}}. \tag{3.24}$$

Note that the second part of the above expression is a sum of independent lognormal variables, which is a special case for the sum of correlated lognormal variables. By using Wilkinson's method, this can be computed in linear time. Therefore, for gates of the same type with the same input state in the same grid, the time complexity is only linear, and we can approximate the sum of leakage of all gates by a lognormal variable which can be superposed in the original expression. Similarly, $I_{gate}$ of different gates in the same grid can be calculated through summation in linear time, and can be approximated by a lognormal variable.

Now, if the chip is divided into $n$ grids, we can reduce the number of correlated leakage components in each grid to a small constant $c$ in their library. As a result, the total number of correlated lognormals to sum is no more than $c \cdot n$. In general, the number of grids is set to be substantially smaller than the number of gates in the chip, which can be regarded as a constant number. Therefore, the complexity required for the sum of lognormals in the grid-based method is reduced from $O(N^2)$ to a substantially smaller constant $O(n^2)$.

As we discussed before, leakage currents of different gates are correlated due to spatially correlated parameters such as transistor gate channel length. Furthermore, $I_{sub}$ and $I_{gate}$ are correlated within the same gate. In addition, leakage currents under different input vectors of the same gate are correlated because they are sensitive to the same parameters of the gate, regardless of whether or not these are spatially correlated. We must carefully predict the distribution of total leakage in the circuit, and the correlations of these leakage currents must be correctly considered when they are summed up.

As we mentioned before, the leakage currents that arise from the same leakage mechanisms in the same grid from the same entry of the look-up table are merged into a single lognormally distributed leakage component to reduce the number of correlated leakage components to sum. Let $I_1^{sum}$ and $I_2^{sum}$ be two merged sums, which correspond to subthreshold leakage and gate oxide leakage components in the same grid, respectively. These can be calculated as

$$I_1^{sum} = e^{Y_1^0+\beta_0\cdot dL}\cdot\sum_{i=1}^{t}\alpha_i\cdot e^{\beta_1\cdot dT_{ox,i}} = e^{Y_1^0+\beta_0 dL}e^{\xi}, \quad (3.25)$$

$$I_2^{sum} = e^{Y_2^0+\beta_0'\cdot dL}\cdot\sum_{i=1}^{t'}\alpha_i'\cdot e^{\beta_1'\cdot dT_{ox,i}'} = e^{Y_2^0+\beta_0 dL}e^{\gamma}, \quad (3.26)$$

where $e^{\xi}$ and $e^{\gamma}$ are the lognormal approximations of the sum of independent lognormals, $\sum_{i=1}^{t}\alpha_i\cdot e^{\beta_1\cdot dT_{ox,i}}$ and $\sum_{i=1}^{t'}\alpha_i\cdot e^{\beta_1'\cdot dT_{ox,i}'}$ in $I_1^{sum}$ and $I_2^{sum}$ respectively, as described in (3.24).

Note that $\sum_{i=1}^{t}\alpha_i\cdot e^{\beta_1\cdot dT_{ox,i}}$ and $\sum_{i=1}^{t'}\alpha_i\cdot e^{\beta_1'\cdot dT_{ox,i}'}$ may be correlated, since the same gate could have both subthreshold and gate leakage. Therefore, $e^{\xi}$ and $e^{\gamma}$ are correlated, and we need to derived the correlation between $\xi$ and $\gamma$. Since the $T_{ox}$ values are independent in different gates, we can easily compute the correlation, $cov(\sum_{i=1}^{t}\alpha_i\cdot e^{\beta_1\cdot dT_{ox,i}}, \sum_{i=1}^{t'}\alpha_i\cdot e^{\beta_1'\cdot dT_{ox,i}'})$ as

$$\sum\alpha_i\alpha_i' e^{(\beta_i^2+\beta_i'^2)\sigma_{T_{ox,i}'}^2/2}(e^{\beta_i\beta_i'\sigma_{T_{ox,i}}^2}-1). \quad (3.27)$$

The correlation between $e^{\xi}$ and $e^{\gamma}$ is then found as

$$\begin{aligned} cov(e^{\xi},e^{\gamma}) &= E(e^{\xi+\gamma}) - E(e^{\xi})E(e^{\gamma}) \\ &= e^{\mu_\xi+\mu_\gamma+(\sigma_\xi^2+\sigma_\gamma^2)/2}(e^{cov(\xi,\gamma)/2}-1), \quad (3.28)\end{aligned}$$

where $\mu_\xi$ / $\mu_\gamma$ and $\sigma_\xi$ / $\sigma_\gamma$ are the mean value and standard deviation of $\xi$ / $\gamma$, respectively. Solving (3.28) for $cov(\xi, \gamma)$, we have

$$cov(\xi, \gamma) = 2log\left(1 + \frac{cov(e^{\xi}, e^{\gamma})}{e^{m_{\xi}+m_{\gamma}+(\sigma_{\xi}^2+(\sigma_{\gamma}^2)/2}}\right). \tag{3.29}$$

Since $e^{\xi}$ and $e^{\gamma}$ are approximations of $\sum_{i=1}^{t} \alpha_i \cdot e^{\beta_1 \cdot dT_{ox,i}}$ and $\sum_{i=1}^{t'} \alpha_i \cdot e^{\beta_1' \cdot dT'_{ox,i}}$ respectively, it is reasonable to assume that

$$cov(e^{\xi}, e^{\gamma}) = cov\left(\sum_{i=1}^{t} \alpha_i \cdot e^{\beta_i \Delta T_{ox,i}}, \sum_{i=1}^{t'} \alpha_i' \cdot e^{\beta_i' \Delta T'_{ox,i}}\right). \tag{3.30}$$

At the same time, the mean values and standard deviations of $\xi$ and $\gamma$ are already known from the approximations, therefore, the computation of $cov(\xi, \gamma)$ is easily possible.

We can extend the framework for statistical computation of full-chip leakage considering spatial correlations in intra-die variations of parameters to handle inter-die variation. For each type of parameter, a global random variable can be applied to all gates in the circuit to model the inter-die effect. In addition, this framework is general, and can be used to predict the circuit leakage under other parameter variations or other leakage components such as the JTL current discussed in Section 3.2.2. However, if the Gaussian or lognormal assumption does not work, we can not use the grid-based method to estimate full-chip leakage.

### 3.4.3 Projection-based statistical analysis methods

The projection-based method is used to compute the moments of statistical leakages via moment matching techniques, which are well developed in the area of interconnect model order reduction [70]. In the projection-based method, quadratic models in (3.16) and (3.17) are used to guarantee accuracy. Li et al. [38] proposed a projection-based approach (PROBE) to reduce the quadratic modeling cost. In a quadratic model, we need to compute all elements of the quadratic coefficient matrix, which is the main difficulty. Take $A_{chip}$ in (3.18) for example. In most real cases, $A_{chip}$ is rank-deficient. As a result, this full-rank matrix $A_{chip}$ can be approximated by another low-rank matrix $\tilde{A}_{chip}$, if $\|A_{chip} - \tilde{A}_{chip}\|_F$ is minimized. Here, $\| \cdot \|_F$ denotes the Frobenius norm, which is the square root of the sum of the squares of all matrix elements. Li et al. [38] proved that the optimal rank-R approximation is:

$$\tilde{A}_{chip} = \sum_{r=1}^{R} \lambda_{chipr} P_{chipr} P_{chipr}^T, \tag{3.31}$$

where $m$ stands for the total number of random variables, and $\lambda_{chipr} \in R$ and $P_{chipr} \in R^m$ are the $r$-th dominant eigenvalue and eigenvector of the matrix $A_{chip}$, respectively.

The PROBE method proposed in [38] is efficient in handling $10^1 \sim 10^2$ random variables. However, there are $10^3 \sim 10^6$ variables in a full-chip SLA. This led Li et al. [35] to improve the projection-based analysis algorithm by exploring the underlying sparse structure of the leakage analysis problem. Specifically, the improved methodology includes: 1) Two-step iterative algorithm for quadratic SLA modeling; 2) Quadratic model compaction algorithm for leakage distribution estimation; and 3) Incremental analysis algorithm for locally updating the leakage distribution.

The algorithm in [35] estimates the full-chip leakage power with consideration of both inter-die and intra-die process variations, and is not limited to lognormal distributions. It can be applied to either grids or gates in the full chip.

This method has linear computational complexity in relation to circuit size under the assumption that all the given random variables are linearly independent. However, this is not generally true for practical circuits as spatial correlation typically occurs in many layout related parameters. One way to mitigate this problem as suggested by the method is to convert the correlated random variable into independent ones via some decomposition processes such as principal component analysis (PCA). PCA essentially is based on the singular value decomposition (SVD), whose complexity is $(nN^2)$, where $n$ is the number of correlated variables and $N$ is the number of independent variables. As a result, the paper is not linear if PCA is considered.

The projection-based SLA starts from the standard cell library characterization. In this step, the leakage current of each gate can be approximated by a regression model. Typically, modeling the variations in one gate only involves a few (e.g., $5 \sim 10$) random variables. Therefore, we can run SPICE simulations (or utilize measurement models if available) and apply the PROBE [38] method to fit the rank-$K$ model for each gate:

$$\log(I_{G,i}) = \sum_{j=1}^{K} \lambda_{G,i,j} \cdot \left(\tilde{P}_{G,i,j}^T \cdot \Delta X_{G,i}\right)^2 + \tilde{B}_{G,i}^T \cdot \Delta X_{G,i} + C_{G,i}, \tag{3.32}$$

where $X_{G,i}$ is defined in (3.14), and $\lambda_{G,i,j}$, $P_{G,i,j}$, $B_{G,i}$ and $C_{G,i}$ are the coefficients. Substituting (3.14) into (3.32):

$$\log(I_{G,i}) = \sum_{j=1}^{K} \lambda_{G,i,j} \cdot \left(P_{G,i,j}^T E\right)^2 + B_{G,i}^T E + C_{G,i}, \tag{3.33}$$

$$P_{G,i,j} = V_{G,i}^T \tilde{P}_{G,i,j}^T, \quad B_{G,i} = V_{G,i}^T \tilde{B}_{G,i}^T, \tag{3.34}$$

where $P_{G,i,j} \in R^m$ , $B_{G,i} \in R^m$, and $m$ is the total number of random variables for the whole chip. Note that the sizes of $P_{G,i,j}$ and $B_{G,i}$ in (3.33) are much larger than the sizes of $P_{G,i,j}$ and $B_{G,i}$ in (3.32). However, as discussed in Section 3.3, $V_{G,i}$ is a sparse matrix. So both $P_{G,i,j}$ and $B_{G,i}$ are sparse, too. To simplify the notation, the following symbols are defined to represent all gate-based leakage models in a matrix form:

$$\begin{aligned}\log(I_{leak,Gate}) &= \left[\log(I_{G,1}), \log(I_{G,2}), \ldots, \log(I_{G,N})\right]^T, \\ \Lambda_{G,j} &= \left[\lambda_{G,1,j}, \lambda_{G,2,j}, \ldots, \lambda_{G,N,j}\right]^T, \\ P_{G,j} &= \left[P_{G,1,j}, P_{G,2,j}, \ldots, P_{G,N,j}\right]^T, \\ B_G &= \left[B_{G,1}, B_{G,2}, \ldots, B_{G,N}\right], \\ C_G &= \left[C_{G,1}, C_{G,2}, \ldots, C_{G,N}\right]^T. \end{aligned} \tag{3.35}$$

Comparing (3.35) with (3.33), it is easy to verify that:

$$\log(I_{leak,Gate}) = \sum_{j=1}^{K} \Lambda_{G,j} \otimes \left(P_{G,j}^T E\right) \otimes \left(P_{G,j}^T E\right) + B_G^T E + C_G, \tag{3.36}$$

where $\otimes$ stands for the point-wise multiplication, i.e., $[a_1, a_2, \ldots,]^T \otimes [b_1, b_2, \ldots]^T = [a_1 b_1, a_2 b_2, \ldots]^T$.

After the standard cell library characterization, we need to extract the low-rank quadratic model of the full-chip leakage current in an efficient way. As shown in (3.17), full-chip leakage current can be calculated by summing leakage currents of all the gates. Applying the log transform to both sides of (3.17),

$$\log(I_{leak,chip}) = \log\left[e^{\log(I_{G,1})} + e^{\log(I_{G,2})} + \cdots + e^{\log(I_{G,N})}\right]. \tag{3.37}$$

Substitute (3.36) into (3.37), and then apply second order Taylor expansion on it, we will obtain the quadratic model in the form of (3.18) after some mathematical manipulations. In this quadratic model, the coefficients are given by

$$\begin{aligned} C_{chip} &= \log\left(\frac{1}{\alpha}\right), && (3.38) \\ B_{chip} &= \alpha \cdot B_G \cdot \Phi, && (3.39) \\ A_{chip} &= \alpha \cdot \sum_{j=1}^{K} P_{G,j} \cdot diag(\Phi \otimes \Lambda_{G,j}) \cdot P_{G,j}^T \\ &\quad + \frac{\alpha}{2} \cdot B_G \cdot diag(\Phi) \cdot B_G^T - \frac{\alpha^2}{2} \cdot B_G \cdot \Phi\Phi^T \cdot B_G^T, && (3.40) \end{aligned}$$

where $diag([a_1, a_2, \ldots]^T)$ means the diagonal matrix with the elements $\{a_1, a_2, \ldots\}$ and

$$\begin{aligned} \alpha &= \frac{1}{e^{C_{G,1}} + e^{C_{G,2}} + \cdots + e^{C_{G,N}}}, && (3.41) \\ \Phi &= \left[e^{C_{G,1}} e^{C_{G,2}} \cdots e^{C_{G,N}}\right]^T. && (3.42) \end{aligned}$$

Note that computing the values of $\alpha$ and $\Phi$ has linear computational complexity. After getting the values of $\alpha$ and $\Phi$, the coefficients $C_{chip}$ and $B_{chip}$ can be evaluated from (3.39) and (3.40). Since $B_G$ in (3.40) is sparse, we can compute the matrix-vector product $B_G$ with linear computational complexity. Therefore, we can finish the extraction of both $C_{chip}$ in (3.39) and $B_{chip}$ in (3.40) in linear time.

However, the quadratic coefficient matrix $A_{chip}$ in (3.40) is no longer sparse, which is the major difficulty here. We can understand this non-sparsity feature from the last term at the right-hand side of (3.40). Because the vector $\Phi$ is dense, $\Phi\Phi^T$ should be a dense matrix. Therefore, $B_G\Phi\Phi^T B_G^T$ is dense, although $B_G$ is sparse. For this reason, explicitly constructing $A_{chip}$ based on (3.40) becomes extremely expensive. To solve this problem, the projection-based method uses an iterative algorithm, which consists of two steps: Krylov subspace generation and orthogonal iteration. In this process, we try to find the optimal low-rank approximation of $A_{chip}$ instead of the full matrix $A_{chip}$.

a) Krylov Subspace Generation

We can obtain the optimal Rank-$R$ approximation of $A_{chip}$ by the dominant eigenvalues $\{\lambda_{chip1}, \lambda_{chip2}, \ldots, \lambda_{chipR}\}$ and eigenvectors $\{P_{chip1}, P_{chip2}, \ldots, P_{chipR}\}$, as shown in (3.31). All the linear combinations of these dominant eigenvectors generate a subspace called the dominant invariant subspace [21] and is denoted as

$$span\{P_{chip1}, P_{chip2}, \ldots, P_{chipR}\}. \tag{3.43}$$

We can approximate the dominant invariant subspace in (3.43) by the following *Krylov subspace* [21]

$$span\{Q_0, A_{chip}Q_0, A_{chip}^2Q_0, \ldots, A_{chip}^{R-1}Q_0\}, \tag{3.44}$$

where $Q_0 \in R^m$ is a non-zero vector which is not orthogonal to any dominant eigenvectors. First, we need to develop an algorithm to extract the Krylov subspace which is a good approximation of the dominant invariant subspace. Then, the extracted Krylov subspace can be used as the starting point of the orthogonal iteration in next step. By doing this, the orthogonal iteration converges to the dominant invariant subspace in only a few iteration steps. The Arnoldi algorithm used in matrix computations [21] can be applied here to generate the Krylov subspace.

Fig. 3.8 is the flow of a simplified implementation of the Arnoldi algorithm [35]. Here, Step 3 shows the key point of the Arnoldi algorithm. In this step, we compute the matrix-vector product $Q_r = A_{chip}Q_{r-1}$. Because $A_{chip}$ is a large and dense matrix, (3.45) does not construct the matrix $A_{chip}$ explicitly. Instead, it computes $A_{chip}Q_{r-1}$ implicitly. For example, we can multiply all terms in (3.40) by $Q_{r-1}$ separately and then add them together. In this way, $A_{chip}$ in (3.40) can be obtained by a summation of the products of many sparse or low-rank matrices. As a result, we can compute the implicit matrix-vector product in (3.45) with linear computational complexity. More details are provided in [35].

**Algorithm**: Simplified Arnoldi algorithm.

**Input**: Full matrix $A_{chip}$
**Output**: Extracted Krolov subspace $Q$

```
1. Randomly select an initial vector $Q_0 \in R^m$.
2. $Q_1 = Q_0/\|Q_0\|_F$
3. For $r = 2, 3, \ldots, R$
```

$$
\begin{aligned}
Q_r \;=\; & \alpha \cdot \sum_{j=1}^{K} P_{G,j} \cdot diag(\Phi \otimes \Lambda_{G,j}) \cdot P_{G,j}^T \cdot Q_{r-1} \\
& + \frac{\alpha}{2} \cdot B_G \cdot diag(\Phi) \cdot B_G^T \cdot Q_{r-1} \\
& - \frac{\alpha^2}{2} \cdot B_G \cdot \Phi\Phi^T \cdot B_G^T \cdot Q_{r-1} \qquad (3.45)
\end{aligned}
$$

```
4. Orthogonalize $Q_r$ to all $Q_i (i = 1, 2, \ldots, r-1)$
5. $Q_r = Q_r/\|Q_r\|_F$
6. End For
7.    $Q = [Q_R, \ldots, Q_2, Q_1]$    (3.46)
```

Figure 3.8: The flow of simplified Arnoldi algorithm

B. Orthogonal Iteration

Note that the Krylov subspace we compute from Fig. 3.8 is not exactly equal to the dominant invariant subspace. Because of this, we need further apply an orthogonal iteration [21] which exactly converges to the dominant invariant subspace, and the matrix $Q$ in (3.46) can be used as the initial value in the iteration. The starting

**Algorithm**: Orthogonal iteration.

**Input**: the matrix $Q \in R^{m \times R}$.
**Output**: $Q_{chip}$ and $U_{chip}$.

```
1. $Q^{(1)} = Q$, where the superscript stands for the iteration index.
2. For $i = 2, 3, \ldots$
       $$Z^{(i)} = \alpha \cdot \sum_{j=1}^{K} P_{G,j} \cdot diag(\Phi \otimes \Lambda_{G,j}) \cdot P_{G,j}^T \cdot Q^{(i-1)} + \frac{\alpha}{2} \cdot B_{G,} \cdot diag(\Phi) \cdot B_{G,}^T \cdot Q^{(i-1)} - \frac{\alpha^2}{2} \cdot B_{G,} \cdot \Phi\Phi^T \cdot B_{G,}^T \cdot Q^{(i-1)} \quad (3.47)$$
3.     $Q^{(i)}U^{(i)} = Z^{(i)}$ (QR factorization)
4. End For
5. $Q_{chip} = Q^{(i)}, U_{chip} = U^{(i)}$    (3.48)
```

Figure 3.9: The flow of simplified orthogonal iteration algorithm.

point of the orthogonal iteration can be any matrix, theoretically. However, as we mentioned before, the Krylov subspace $Q$ is a good approximation of the dominant invariant subspace, so using $Q$ as the starting point will help the orthogonal iteration converge within a few iteration steps.

Fig. 3.4.3 summarizes a simplified implementation of the orthogonal iteration algorithm. In (3.47), note that $N \gg R$ in $Q^{(i-1)} \in R^{N\times R}$, because $R$ is typically small (e.g., around 10) in most practical applications. As a result, similar to (3.45), computing $Z^{(i)}$ in (3.47) has linear computational complexity. Similarly, since $Z^{(i)} \in R^{m\times R}$ contains only a few columns, the QR factorization in Step 4 of Fig. 3.4.3 also can be done in linear time for the same reason. In Fig. 3.4.3, if the columns in the initial matrix $Q$ are not orthogonal to the dominant invariance subspace, then the orthogonal iteration is probably convergent [21]. After the orthogonal iteration converges, we can use $Q_{chip}$ and $U_{chip}$ in (3.48) to determine the optimal Rank-$R$ approximation of $A_{chip}$ [21]

$$\tilde{A}_{chip} = Q_{chip} U_{chip} Q_{chip}^T. \tag{3.49}$$

Combining (3.49) with (3.18) yields

$$\log(I_{chip}) = E^T \cdot \left( Q_{chip} U_{chip} Q_{chip}^T \right) \cdot E + B_{chip}^T E + C_{chip}, \tag{3.50}$$

where expressions of $C_{chip}$ and $B_{chip}$ are given in (3.39) and (3.40). We assume a given approximation rank $R$ in the algorithms in Fig. 3.8 and Fig. 3.4.3. Based on the approximation error, the value of $R$ can be iteratively determined. For example, starting from a low-rank approximation, if the modeling error remains large, we should iteratively increase the value of $R$. In most practical cases, $R$ in the range of $5 \sim 15$ can provide approximation results which is already sufficiently accurate.

In summary, to extract the low-rank quadratic model of the full-chip leakage current, we can use a two-step iterative algorithm which only involves simple vector operations and sparse matrix-vector multiplications. Therefore, this algorithm has linear computational complexity in relation to circuit size. Furthermore, it is not necessary to construct the matrix $\tilde{A}_{chip}$ in (3.49) explicitly. An algorithm that efficiently estimates the leakage current distribution will be introduced in the following part.

The quadratic function in (3.50) is $m$-dimensional ($m$ is large, typically). Hence, estimating the leakage distribution directly from (3.50) is not efficient. Hence, we will use a quadratic model compaction algorithm to convert the high-dimensional model to a low-dimensional one, and keep the leakage distribution unchanged at the same time.

```
Algorithm: QUADRATIC MODEL COMPACTION ALGORITHM.
Input: the quadratic model in (3.50).
Output: full chip leakage distribution.

1. $Q_{comp}\left[U_{comp}B_{comp}\right] = \left[Q_{chip}B_{chip}\right]$ (QR factorization).
2.     $\Omega = Q_{comp}^T E$                                              (3.51)
3. $\log(I_{chip}) = \Omega^T \cdot \left(U_{comp}U_{chip}U_{comp}^T\right) \cdot \Omega + B_{comp}^T\Omega + C_{chip}$    (3.52)
```

Figure 3.10: The flow of quadratic model compaction algorithm.

Fig. 3.10 shows the basic steps in the proposed quadratic model compaction algorithm. In [35], it is proved that the quadratic models in (3.50) and (3.52) are equivalent, and the random variables $\Omega$ defined in (3.51) are independent and satisfy the standard normal distribution.

The quadratic function in (3.52) is $(R+1)$-dimensional, where $R+1 \ll N$. We can apply (3.52) to extract the PDF/CDF of $\log(I_{chip})$, for example, using either Monte

Carlo analysis or APEX [37]. Then, the distribution of $I_{chip}$ can be easily computed by a simple nonlinear transform [57].

There is another point we want to mention here. Leakage analysis can be incremental. If any local changes to a circuit are made, incremental leakage analysis can facilitate a quick update on the leakage distribution. For simplicity, we only discuss the case where one gate is changed. It can be directly extended to handle the simultaneous change of multiple gates.

Assume the $i$-th gate is changed (e.g., a high $V_{th}$ gate is replaced by a low $V_{th}$ gate), then the full-chip leakage changes as

$$I_{chip}^{new} = I_{chip}^{old} - I_{G,i}^{old} + I_{G,i}^{new}, \tag{3.53}$$

where $I_{chip}^{old}/I_{chip}^{new}$ and $I_{G,i}^{old}/I_{G,i}^{new}$ represent the leakage currents of the entire chip and the $i$-th gate before /after the change, respectively. Once we know the low-rank quadratic models of $\log\left(I_{chip}^{old}\right)$, $\log\left(I_{G,i}^{old}\right)$ and $\log\left(I_{G,i}^{new}\right)$, incremental leakage analysis can quickly generate the low-rank model for $\log\left(I_{chip}^{new}\right)$. Compared with (3.17), (3.53) only contains a few terms. As a result, it is much more efficient to update the leakage distribution using (3.53) than the full leakage analysis from (3.17). The iterative algorithm and compaction algorithm we discussed above can be directly applied to (3.53).

As mentioned earlier, the projection-based approach has its limitations since it starts with independent random variables generated by PCA. The pre-process PCA can be expensive (at least non-linear) for a large number of resulting independent random variables. If we assume a small number of independent variables, the proposed Krylov based reduction will become less relevant as variable reduction has been done in the PCA step.

## 3.5 Summary

In this chapter, we have presented problem of statistical leakage analysis under process variations and spatial correlations. We then briefly discussed the existing approaches. Then we presented two statistical leakage analysis methods: the stochastic spectral based method with variable reduction techniques and the virtual grid based approach.

We have presented a linear algorithm for full-chip statistical analysis of leakage currents in the presence of any condition of spatial correlation (strong or weak). The new algorithm adopts a set of uncorrelated virtual variables over grid cells to represent the original physical random variables with spatial correlation and the size of grid cell is determined by the correlation length. As a result, each physical variable is always represented by virtual variables in local neighbor set. Furthermore, a look-up table is used to cache the statistical leakage information of each type of gate in the library to avoid computing leakage for each gate instance. As a result, the full-chip leakage can be calculated with $O(N)$ time complexity, where $N$ is the number of grid cells on chip. The new method maintains the linear complexity from strong to weak spatial correlation and has no limitation of leakage current model or variation model. This paper also offers an incremental analysis capability to update the leakage distribution more efficiently when local changes to a circuit are made. Experimental results show the proposed method is about 1000X faster than the recently proposed method [9] with similar accuracy and many orders of magnitude times over the Monte Carlo method. Numerical results shows the proposed incremental analysis can further achieve significant speedup over the full leakage analysis.

# Chapter 4

# Gate-based Statistical Leakage Power Analysis

## 4.1 Introduction

In the chapter, we propose a new general full-chip leakage modeling and analysis method. The new method starts with the process variational parameters such as the channel length, $\delta L$, gate oxide thickness, $\delta T_{ox}$, and it can derive the full-chip leakage current $I_{leak}$ in terms of those variables directly (or their corresponding transformed variables). Unlike existing grid-based methods, which trade the accuracy for speedups, the new method is gate-based method and uses principal component analysis (PCA) to reduce the number of variables with much less accuracy loss assuming that the geometrical variables are Gaussian. For non-Gaussian variables, independent component analysis (ICA) [24] can be used. The new method considers both inter-die and intra-die variations and it can work with various spatial correlations. The proposed method becomes linear under strong spatial correlations. Unlike the existing approaches [9, 22], the new method does not make any assumptions about the distributions of final total leakage currents for both gates and chips and does not require any grid-based partitioning of the chip. Compared with [6], the proposed method applies a more efficient multi-dimensional numerical quadrature method (versus on reduced number of variables using inter-production via the moment matching), considers more accurate leakage models and presents more comprehensive comparisons with other methods.

In the new method, we first fit both the subthreshold and gate oxide leakage currents into analytic expressions in terms of parameter variables. We show that by using more terms in the gate level analytic models, we can achieve better accuracy than [9]. Second, The new method employs the orthogonal polynomials, which gives the best representation for specific distributions [20] and is also called the *spectral stochastic* method, to represent the variational gate leakages in an analytic form in terms of the random variables. The step is achieved by using the numerical Gaussian quadrature method, which is much faster than the Monte Carlo method. The total leakage currents are finally computed by simply summing up the resulting analytical orthogonal polynomials of all gates (their coefficients). The spatial correlations are taken care of by PCA or ICA, and at the same time, the number of random variables can also been substantially reduced in the presence of strong spatial correlations during the decomposition process. Experimental results on the PDWorkshop91 benchmarks on a 45nm technology show that the proposed method is about 10$\times$ faster than the recently proposed method [9] with constant better accuracy.

## 4.2 Flow of gate-based method

To analyze the statistical model of chip-level leakage current, traditional methods are grid-based. Since the number of gates on a whole chip is very large, and every gate has its own variational parameters, which means that the number of random variables is huge. So considering efficiency, the traditional methods partition a chip to several grids, and assume that all the gates in one grid have the same parameters as mentioned in 3.4.2. However, this is not the real case. Take Fig. 4.1 as one example. Here the distance between Gate1 and Gate2 is smaller than the distance between Gate1 and Gate3. In grid-based method, we suppose that Gate1 has strong correlation with Gate3, and has weak correlation with Gate2. But actually, the situation is opposite. In this section, we will present the new full-chip statistical leakage analysis

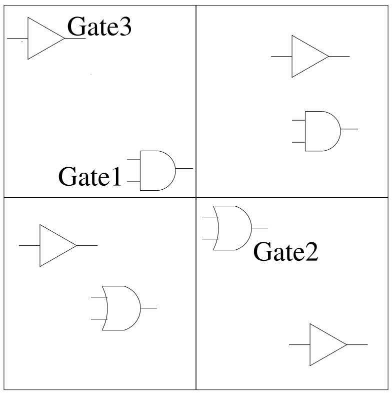

Figure 4.1: An example of a grid-based partition.

method. This method is gate-based instead of grid-based, while it can gain better speed as well as better accuracy than the method in [9], which is based on grid. Our algorithm is shown in Fig. 4.2. The new algorithm basically consists of three major parts. The first part (step 1) is pre-characterization, which builds the analytic leakage expressions (3.1) and (3.2) for each type of gates. This step only need to be done once for a standard cell library. The second part (step 2-5) generates a set of independent random variables and builds the gate-level analytic leakage current expressions and covariances. The final part (step 6) computes the final leakage expressions by simple polynomial additions and calculates other statistical information.

**Algorithm**: New full-chip leakage current computation algorithm.

**Input**: standard cell lib, netlist, placement information of design, $\sigma$ of $L$ and $T_{ox}$.
**Output**: analytic expression of the full-chip leakage currents in terms of Hermite polynomials.

1. Generate fitting parameter matrices $a_{sub}$ and $a_{gate}$ of $I_{sub}$ and $I_{gate}$ in (3.1) and (3.2) for each type of gates (after SPICE run on each input pattern) (Section 3.2).
2. Perform PCA to transform and reduce the original parameter variables in $\mathbf{L}$ into independent random variables in $\mathbf{L_k}$. (Section 4.3).
3. Generate Smolyak quadrature points set $\Theta_n^2$ with corresponding weights.
4. Calculate the coefficients of Hermite polynomial of $I_{sub,k}$ and $I_{gate,k}$ for the final leakage analytic expression for each gate using (4.8) and (4.9).
5. Calculate the analytic expression of the full-chip leakage current by simple polynomial additions and calculate $\mu_{leakage}$, $\sigma_{leakage}$, PDF and CDF of the leakage current if required.

Figure 4.2: The flow of proposed algorithm.

## 4.3 Random variables transformation and reduction

In our gate-based approach, instead of using grid-based partitioning, as in [9], to reduce the number of channel length variables in presence of the strong spatial correlation, we applied the principal component analysis (PCA) to reduce the number of random variables. Our method starts with the following random variable vectors:

$$\mathbf{L} = [L_1, L_2, ..., L_n] + \delta L_{inter}, \quad \mathbf{T_{ox}} = [T_{ox1}, T_{ox2}, ..., T_{oxn}] + \delta T_{ox,inter}, \tag{4.1}$$

where $n$ is the total number of gates on the whole chip, $\delta L_{inter}$ and $\delta T_{ox,inter}$ represent the inter-die (global) variations. In total, we have $2n + 2$ random variables. There exist correlations between $L$ among different gates, represented by the covariance matrix $cov(L_i, L_j)$ computed by (3.11).

The first step is to perform PCA on $L$ to get a set of independent random variables $\mathbf{L}' = [L'_1, L'_2, ..., L'_n]$, where $\mathbf{L} = \mathbf{P}\mathbf{L}'$, and $\mathbf{P} = \{p_{ij}\}$ is the $n$ by $n$ principal component coefficient matrix. In this process, singular value decomposition (SVD) is used on the covariance matrix, and the singular values are arranged in a decreasing order, which means that the elements in $\mathbf{L}'$ are arranged in a decreasing weight order. Then the number of elements in $\mathbf{L}'$ can be reduced by only considering the dominant part of $\mathbf{L}'$ as $[L'_1, L'_2, ..., L'_k]$ (for instance, the weight should be bigger than 1%), where $k$ is the number of reduced random variables. Then every element $L'_i$ in $\mathbf{L}'$ can be represented by orthogonal Gaussian random variable $\xi_i$ with normal distribution.

$$L'_i = \mu_i + \sigma_i \xi_i. \tag{4.2}$$

where $\mu_i$ and $\sigma_i$ are the mean value and standard deviation of $L'_i$. And $\mathbf{L}$ can be represented as

$$\mathbf{L} = \begin{pmatrix} \mu_{L1} \\ \mu_{L2} \\ \vdots \\ \mu_{Ln} \end{pmatrix} + \begin{pmatrix} p_{11} & \dots & p_{1k} \\ p_{21} & \dots & p_{2k} \\ \vdots & \vdots & \vdots \\ p_{n1} & \dots & p_{nk} \end{pmatrix} \begin{pmatrix} \sigma_1\xi_1 \\ \sigma_2\xi_2 \\ \vdots \\ \sigma_k\xi_k \end{pmatrix} + \delta L_{inter} \ . \tag{4.3}$$

For $[T_{ox1}, T_{ox2}, ..., T_{oxn}]$, $\delta L_{inter}$ and $\delta T_{ox,inter}$, we can also represent them using the standard Gaussian variables as

$$T_{ox,j} = \mu_{ox,j} + \sigma_{ox,j}\xi_{ox,j}, \quad \delta L_{inter} = \sigma_{L,inter}\xi_{L,inter}, \quad \delta T_{ox,inter} = \sigma_{ox,inter}\xi_{ox,inter}, \tag{4.4}$$

where $\xi_{ox,j}$, $\xi_{L,inter}$ and $\xi_{ox,inter}$ are independent orthonormal Gaussian random variables. As a result, we can present $\mathbf{L}$ and $\mathbf{T_{ox}}$ by $k + n + 2$ independent orthonormal Gaussian random variables.

$$\xi = [\xi_1, \xi_2, ..., \xi_{k+n+2}]. \tag{4.5}$$

Then the $I_{sub}(\mathbf{L}, \mathbf{T_{ox}})$ and $I_{gate}(\mathbf{L}, \mathbf{T_{ox}})$ can be modeled as $I_{sub}(\xi)$ and $I_{gate}(\xi)$, respectively.

But among the $k+n+2$ variables, only $k+2$ variables related to the channel lengths are correlated. In other words, the $n$ variables $T_{ox,i}$ of each gate are independent. As a result, for the $j$th gate, we only have $k+3$ independent variables, the corresponding variable vector, $\xi_g = \{\xi_{g,j}\}$, is defined as

$$\xi_{g,j} = [\xi_1, ..., \xi_k, \ \ \xi_{ox,j}, \xi_{L,inter}, \xi_{ox,inter}]. \tag{4.6}$$

## 4.4 Computation of full-chip leakage currents

For each gate, we need to present the leakage currents in order-2 Hermite polynomials first as shown below for both subthreshold and gate leakage currents – $I_{sub}(\xi_{\mathbf{g,j}})$ and $I_{gate}(\xi_{\mathbf{g,j}})$:

$$I_{sub}(\xi_{\mathbf{g,j}}) = \sum_{i=0}^{P} I_{sub,i,j} H_i^2(\xi_{\mathbf{g,j}}), \quad I_{gate}(\xi_{\mathbf{g,j}}) = \sum_{i=0}^{P} I_{gate,i,j} H_i^2(\xi_{\mathbf{g,j}}), \tag{4.7}$$

where $H_i^2(\xi_{\mathbf{g,j}})$s are order-2 Hermite polynomials. $I_{sub,i,j}$ and $I_{gate,i,j}$ are then computed by the numerical Gaussian quadrature method discussed in Section 2.2.3. Let $S$ be the size of $Z$-dimensional second order (level-2) quadrature point set $\Theta_Z^2$ and $Z = k + 3$. Then $I_{sub,i}$ and $I_{gate,i}$ can be computed as the following:

$$I_{sub,i,j} = \sum_{l=1}^{S} I_{sub}(\vec{\gamma_l}) H_i^2(\vec{\gamma_l}) w_l / < H_i^2(\xi_{\mathbf{g,j}}) >, \tag{4.8}$$

$$I_{gate,i,j} = \sum_{l=1}^{S} I_{gate}(\vec{\gamma_l}) H_i^2(\vec{\gamma_l}) w_l / < H_i^2(\xi_{\mathbf{g,j}}) >, \tag{4.9}$$

where $I_{sub}(\vec{\gamma_l})$ and $I_{gate}(\vec{\gamma_l})$ are computed using (3.1) and (3.2).

As a result, their coefficients for $i$th Hermite polynomial at $j$th gate can be added directly as

$$I_{leakage,i,j} = \sum I_{sub,i,j} + \sum I_{gate,i,j}. \tag{4.10}$$

After the leakage currents are calculated for each gate, we can proceed to compute the leakage current for the whole chip as follows:

$$I_{leakage}(\xi) = \sum_{j=1}^{n} (I_{sub}(\xi_{\mathbf{g,j}}) + I_{gate}(\xi_{\mathbf{g,j}})). \tag{4.11}$$

The summation is done for each coefficient of Hermite polynomials. Then we obtain the analytic expression of the final leakage currents in terms of the $\xi$.

We can then obtain the mean value, variance PDF and CDF of the leakage current very easily. For instance, the mean value and variance for the full-chip leakage current are

$$\begin{aligned}\mu_{leakage} &= I_{leakage,0th}, && (4.12)\\ \sigma^2_{leakage} &= \sum I^2_{leakage,1st} + 2\sum I^2_{leakage,2nd,type1} + \sum I^2_{leakage,2nd,type2}, && (4.13)\end{aligned}$$

where $I_{leakage,ith}$ is the leakage coefficient for $i$th Hermite polynomial of second order defined as follows,

$$H_{0th}(\xi) = 1,\ H_{1st}(\xi) = \xi_i,\ H_{2nd,type1}(\xi) = \xi_i^2 - 1,\ H_{2nd,type2}(\xi) = \xi_i\xi_j,\ i \neq j. \quad (4.14)$$

## 4.5 Time complexity analysis

To analyze the time complexity, one typically does not count the pre-characterization cost of step 1 in Fig. 4.2. For PCA step (step 2), which essentially uses singular value decomposition (SVD) on the covariance matrix, its computation cost is $O(nk^2)$, if we are only interested in the first $k$ dominant singular values. This is the case for strong spatial correlation.

In step 3, we need to compute the weights of Level 2 $(k+3)$-dimensional Smolyak quadrature point set. For quadratic model with $k+3$ variables, the number of Smolyak quadrature points is about $(k+3)^2$. So the time cost for generating Smolyak quadrature points set is $O((k+3)^2)$.

In step 4, we need to call (3.1) and (3.2) $S$ times for each gate. In each call, we need to compute $k+3$ variables in the Hermite polynomials. The computing cost for the two steps is $(O(n(k+3) \times S))$, where $n$ is the number of gates. After the leakage currents are computed for each gate, it takes $O(n(k+3))$ to compute the full-chip leakage current.

The total computing cost is $O(nk^2 + (k+3)^2 + n(k+3)S + n(k+3))$. For second order Hermite polynomials, $S \propto k^2$, so the time complexity becomes $O(nk^3)$. If $k \ll n$ (for strong spatial correlation), we end up with a linear time complexity O(n). In the sub-90nm VLSI technologies, the spatial correlation is really strong, and in the down-scaling process, the spatial correlation will become stronger, which makes sure our method can achieve pretty good time complexity.

## 4.6 Experimental results

The proposed method is implement in Matlab 7.4.0. For comparison purpose, we also implement the grid-based method in [9] and the pure Monte-Carlo method. All the experimental results are carried out in a Linux system with quad Intel Xeon CPUs with 2.99Ghz and 16GB memory.

The methods for full-chip statistical leakage estimation are tested on circuits in the PDWorkshop91 benchmark set. The circuits are synthesized with Nangate Open Cell Library and the placement is from MCNC [41]. The technology parameters come from the 45nm FreePDK Base Kit and PTM models [59].

Table 4.1 shows the detailed parameters for gate length and gate oxide thickness variations. Here we choose two set of $\sigma^2$ distributions. The last column of Table 4.1 shows the standard deviation $(\sigma)$ of each variation. The $3\sigma$ values of parameter variations for $L$ and $T_{ox}$ are set to 12% of the nominal parameter values, of which inter-die variations constitute 20% and intra-die variations, 80% (Case 1); inter-die

Table 4.1: Process variation parameter breakdown for 45nm technology.

| Case 1 | | | |
|---|---|---|---|
| | $\sigma^2$ Distribution | | $(\sigma)$ |
| Gate Length(L) | Inter-die<br>Intra-die<br>$*$ Spatial Correlated | 20%<br><br>80% | $4\% \times 18nm$ |
| Gate Oxide Thickness($T_{ox}$) | Inter-die<br>Intra-die<br>$*$ Non-Correlated | 20%<br><br>80% | $4\% \times 1.8nm$ |

| Case 2 | | | |
|---|---|---|---|
| | $\sigma^2$ Distribution | | $(\sigma)$ |
| Gate Length(L) | Inter-die<br>Intra-die<br>$*$ Spatial Correlated | 50%<br><br>50% | $4\% \times 18nm$ |
| Gate Oxide Thickness($T_{ox}$) | Inter-die<br>Intra-die<br>$*$ Non-Correlated | 50%<br><br>50% | $4\% \times 1.8nm$ |

variations constitute 50% and intra-die variations, 50% (Case 2). The parameter $L$ is modeled as sum of correlated sources of variations, and the gate oxide thickness $T_{ox}$ is modeled as an independent source of variation. The same framework can be easily extended to include other parameters of variations. Both $L$ and $T_{ox}$ in each gate are modeled as Gaussian parameters. For the correlated $L$, the spatial correlation is modeled based on the exponential special correlation in (3.11). For [9], we still partition the chip into a number of regular grids and the numbers of grid partitions of spatial correlation model used for the benchmarks are given in Table 4.1.

For comparison purposes, we perform Monte Carlo (MC) simulations with 500,000 runs, the grid-based method in [9], and the new method on the benchmarks. The large number of MC runs is due to the fact that proposed method is quite accurate. Fig. 4.3 shows the full-chip leakage current distribution (PDF and CDF) of circuit SC0 with 125 gates, considering variation in gate length and gate oxide thickness as in Table 4.1 for Case 1, and the spatial correlation of gate length. It shows that our method fits very well with the MC results, and is more accurate than [9]. Other test cases show the similar comparison results. The results of the comparison of mean

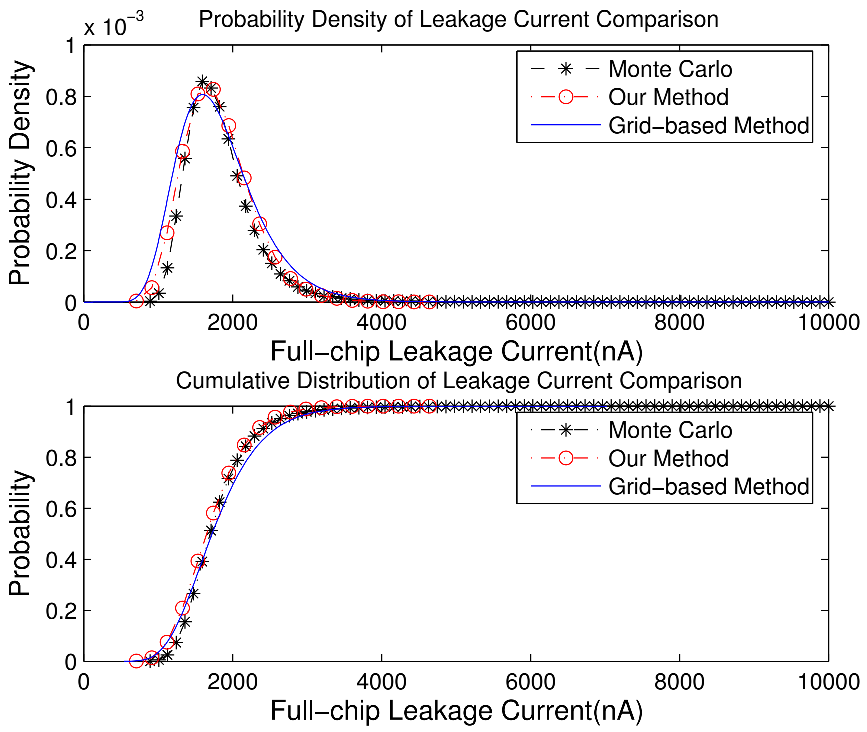

Figure 4.3: Distribution of the total leakage currents of the proposed method, the grid-based method and the MC method for circuit SC0 (Process variation parameters set as Case 1).

values and standard deviations of full-chip leakage currents are shown in Table 4.2 and Table 4.3. For Case 1, the average errors for mean value and standard deviation of the new gate-based method are 0.8% and 4.04%, respectively. While for the grid-based method in [9], the average errors for mean value and standard deviation are 4.08% and 39.7%, respectively. For Case 2, the average errors for mean value and standard deviation of the new gate-based method are 0.8% and 5.51%, respectively. While for the grid-based method in [9], the average errors for mean value and standard deviation are 4.17% and 28.4%, respectively. Our gate-based method is more accurate than the grid-based method, especially for standard deviation value. Since we use

Table 4.2: Comparison of the mean values of full-chip leakage currents among three methods.

| *Circuit* | *Gate* | *Grid* | Variation | $\mu$ of $I_{leak}$ ($\mu$A) | | | *Errors (%)* | |
|---|---|---|---|---|---|---|---|---|
| *Name* | *#* | *#* | Setting | *MC* | [9] | *New* | [9] | *New* |
| SC0 | 125 | 4 | Case 1 | 1.84 | 1.75 | 1.82 | -4.67 | -0.84 |
| | | | Case 2 | 1.84 | 1.75 | 1.82 | -4.85 | -0.87 |
| SC2 | 1888 | 16 | Case 1 | 29.98 | 28.88 | 29.70 | -3.65 | -0.91 |
| | | | Case 2 | 30.02 | 28.89 | 29.75 | -3.77 | -0.89 |
| SC5 | 6417 | 64 | Case 1 | 107.9 | 103.6 | 107.2 | -3.93 | -0.65 |
| | | | Case 2 | 107.9 | 103.6 | 107.2 | -3.9 | -0.65 |

Table 4.3: Comparison standard deviations of full-chip leakage currents among three methods.

| *Circuit* | Variation | $\sigma$ of $I_{leak}$ ($\mu$A) | | | *Errors (%)* | |
|---|---|---|---|---|---|---|
| *Name* | Setting | *MC* | [9] | *New* | [9] | *New* |
| SC0 | Case 1 | 0.495 | 0.668 | 0.524 | 35.0 | -5.77 |
| | Case 2 | 0.632 | 0.726 | 0.689 | 14.9 | 9.04 |
| SC2 | Case 1 | 8.606 | 10.86 | 8.798 | 26.2 | 2.23 |
| | Case 2 | 10.71 | 12.03 | 11.36 | 12.33 | 6.13 |
| SC5 | Case 1 | 26.19 | 41.36 | 25.11 | 57.9 | -4.12 |
| | Case 2 | 26.19 | 41.36 | 25.11 | 57.9 | -4.12 |

45nm technology, while the results in [9] is based on 100nm technology, the error ranges are different (In [9], the average errors for mean value and standard deviation are 1.3% and 4.1%). Results of the grid-based method in [9] will become worse when the technology scales down, since the dominant state assumption is not working any more.

And Table 4.4 also compares the CPU times of the three methods. From this table we can see that even our method is gate-based, it is still faster than the method in [9], which is grid-based. And the proposed method is much faster than the Monte Carlo method. On average, the proposed method has about 16X speedup over the grid based method in [9]. We notice that method in [9] will become faster with smaller number of grids used. But this can lead to large errors even with strong spatial correlations.

Table 4.4: CPU time comparison among three methods.

| *Circuit* | Variation | *Cost time(s)* | | | *Speedup (%)* | |
|---|---|---|---|---|---|---|
| *Name* | Setting | *MC* | [9] | *New* | [9] | *New* |
| SC0 | Case 1 | 378.1 | 11.35 | 1.40 | 8.11 | 270.1 |
| | Case 2 | 358.6 | 7.47 | 1.41 | 5.30 | 254.33 |
| SC2 | Case 1 | $1.35 \times 10^4$ | 168.51 | 18.79 | 30.6 | 718.5 |
| | Case 2 | $1.35 \times 10^4$ | 87.94 | 17.23 | 5.10 | 437.96 |
| SC5 | Case 1 | $2.76 \times 10^5$ | 3335 | 121.2 | 27.52 | 2277 |
| | Case 2 | $2.06 \times 10^5$ | 7798.3 | 443.95 | 17.56 | 464.33 |

## 4.7 Summary

In this chapter, we have presented a gate-based method for analyzing the full-chip leakage current distribution of digital circuit. The method considers both intra-die and inter-die variations with spatial correlations. The new method employs the orthogonal polynomials and multi-dimensional Gaussian quadrature method to represent and compute variational leakage at the gate level, and uses the orthogonal decomposition to reduce the number of random variables by exploiting the strong spatial correlations of intra-die variations. The resulting algorithm compares very favorable with the existing grid-based method in terms of both CPU time and accuracy. The presented method has about 16X speedup over [9] with constant better accuracy.

# Chapter 5

# Linear Statistical Leakage Power Analysis Using New Characterization in Standard Cell Library

## 5.1 Introduction

When the spatial correlation is weak, existing general approaches mentioned in 3 and 4 do not work well as the number of correlated variables can not be reduced too much. Recently an efficient method was proposed [82] to address this problem. The method is based on simplified gate leakage models and formulates the major computation tasks into matrix-vector multiplications via Taylor's expansion. It then applies fast numerical methods like the Fast Multi-Pole method or the pre-corrected FFT method to compute the multiplication. However, this method assumes the gate-level leakage currents are purely lognormal, and the chip-level leakage is also approximated by lognormal distribution, which is not the case as we will show in the paper. Also it can only give the mean and variances, not the complete distribution of the leakage powers.

In this chapter, firstly, we present a new linear-time algorithm for statistical leakage analysis in the presence of any spatial correlation (from no spatial correlation to 100% correlated situation). The new algorithm exploits the following property: leakage current of a gate in the presence of spatial correlation is affected by process variations in the neighbor area. As a result, gate leakage current can be efficiently computed by considering the neighbor area in constant time. We adopt a newly proposed spatial correlation model where a new set of location-dependent uncorrelated virtual variables are defined over grid cells to represent original correlated random variables via fitting. To compute the statistical leakage current of a gate on the new set of variables, the orthogonal polynomials based collocation method is applied and the variational gate leakages and total leakage currents are represented in an analytic form in terms of the random variables, which can give complete statistic information. The new method considers both inter-die and intra-die variations and can work with any spatial correlations (strong or weak, as defined in Section 3.3). Unlike the existing approaches [9, 22], the new method does not make any assumptions about the final distributions of total leakage currents for both gate and chip levels. In case of medium and strong correlations, the proposed method can also work in linear time by properly sizing the grid cells so both locality of correlation and accuracy are still preserved.

Furthermore, we bring forth a novel characterization of standard cell library for statistical leakage information and we have the following observations: (1) The set of neighbor cells is usually small ($\sim 10$), and only considering the relative position, not the absolute position on chip. (2) As proved later, the number of neighbor cells involved in our model is not related to the strength (level) of spatial correlation. (3) The orthogonal polynomials based stochastic collocation method is applied and the variational leakage of a gate is represented in an analytic form in terms of the virtual random variables, which can give complete distribution. (4) The gate-level leakage distribution is only related to the type of gates in a standard cell library. This statistical leakage characterization can be stored in a look-up table, which only needs to be built once for a standard cell library. And the full-chip leakage of any chip can be easily calculated by summing up certain items in the look-up table.

The main contributions of this chapter are as follows:

1. We applied the virtual grid based model for spatial correlation modeling in the statistical leakage analysis and making the resulting algorithm linear time for the first time for all the spatial correlation (weak or strong) cases.

2. We proposed a new characterization in SCL for statistical leakage analysis. The corresponding algorithm can accelerate full-chip full-chip statistical analysis for all spatial correlation conditions (from weak to strong). To the best knowledge of the authors, the proposed approach is the first published algorithm which can guarantee $O(N)$ time complexity for all spatial correlation conditions.

3. In addition, an incremental algorithm has been proposed. When a few local changes are made, only a small circuit (includes the changing gates) is involved in the updating process. Our numerical examples show the incremental analysis can achieve $10\times$ further speedup compared with the library-enabled full-chip analysis approach.

In addition to the main contributions, we also present a forward-looking way to extend the proposed method to handle run-time leakage analysis. In order to estimate maximum run-time leakage, the input state under the maximum leakage input vector needs to be chosen. While for transient run-time leakage simulation, every time the input vector changes, the input states of some gate on a chip will be updated. Therefore, the incremental technique makes efficient run-time leakage simulation possible. More details are given in Section 5.4.6.

Experimental results on the PDWorkshop91 benchmarks on a 45nm technology show the proposed method using novel characterization in SCL is on average two orders of magnitude faster than the recently proposed method [9] with similar accuracy. For weak correlation situation, more speed-up can be observed. We remark that the experiment in this work is based on idle-time leakage. However, the linear time algorithm can also be applied to run-time leakage by selecting different input states under certain input vectors. Notice that glitch events is ignored in the simplified discussion, which may cause estimation errors [39], and needs to be considered in the future work. More details are discussed in Section 5.4.6.

## 5.2 Virtual grid-based spatial correlation model

The virtual grid-based model is based on the observation that the leakage current of a gate in the presence of spatial correlation only correlates to its neighbor area. If we can introduce a set of uncorrelated variables to model the localized correlation, computing the leakage current of one gate can be done in a constant time by only considering its neighbor area. Hence total full-chip statistical leakage currents can then be computed by simply adding all the gate leakage currents together in terms of the virtual set of variables in linear time. Notice that the virtual random variables in different grids are always independent, which is different from traditional grid-based model. This idea was proposed recently for fast statistical timing analysis [11] to address the computational efficient modeling for weak spatial correlation, which is similar to the PCA-based approach [64], but with different set of independent variables.

Specifically, the chip area is still divided into a set of grid cells. When the spatial correlation is weak enough to be ignored, the cell can become so small that one cell only contains one gate. Then we introduce a "virtual" random variable for each cell for one source of process variation.

These virtual random variables are *independent* and will be the basis for statistical leakage current calculation concerned with spatial correlation. Then we can express the original physical random variable of a gate in a grid cell as a linear combination of the virtual random variables of its own cell as well as its nearby neighbors. Since virtual random variable in each cell has specific location on chip, such location-dependent correlation model still retains the important spatial physical meaning (in contrast to PCA-based models). The grid partition can be made of any shape. We use hexagonal grid cells [11] in this work since they have minimum anisotropy for 2D space.

Here we define the distance between centers of two direct neighbor grid cells as the grid length $d_c$. Gates located in the same cell have strong correlation (larger than a given threshold value $\rho_{high}$) and are assumed to have the same parameter variations. And "spatial correlation distance" $d_{max}$ is defined as the minimum distance beyond which the spatial correlation between any two cells is sufficiently small (or smaller than a given threshold value $\rho_{low}$) so we can ignore it.

In this model, the $j$th grid cell is associated with one virtual random variable $\xi_j \sim N(0,1)$, which is independent of all other virtual random variables. $\Delta L_j$ can then be expressed as its $k$ closest neighbor cells. We introduce the concept of *correlation index neighbor* set $T(j)$ for cell $j$, and the corresponding variable vector, $\vec{\xi}_{g,j}$, is defined as

$$\vec{\xi}_{grid_j} = [\xi_q, q \in T(j)], \tag{5.1}$$

to model the spatial correlation of $\Delta L_j$ as

$$\Delta L_j = \sum\nolimits_{q \in T(j)} \alpha_q \cdot \xi_q. \tag{5.2}$$

For example, hexagonal grid partition is used as shown in Fig. 5.1, and if $T(i)$ for each cell is defined as its closest $k = 7$ neighbor cells, then $\Delta L$ located at cell $(x_i, y_i)$ can be represented as a linear combination of seven virtual random variables located in its neighbor set. Take $\Delta L_1$ in Fig. 5.1 for instance, we have $\Delta L_1 = \alpha_1\xi_1 + \alpha_2\xi_2 + \ldots + \alpha_7\xi_7$.

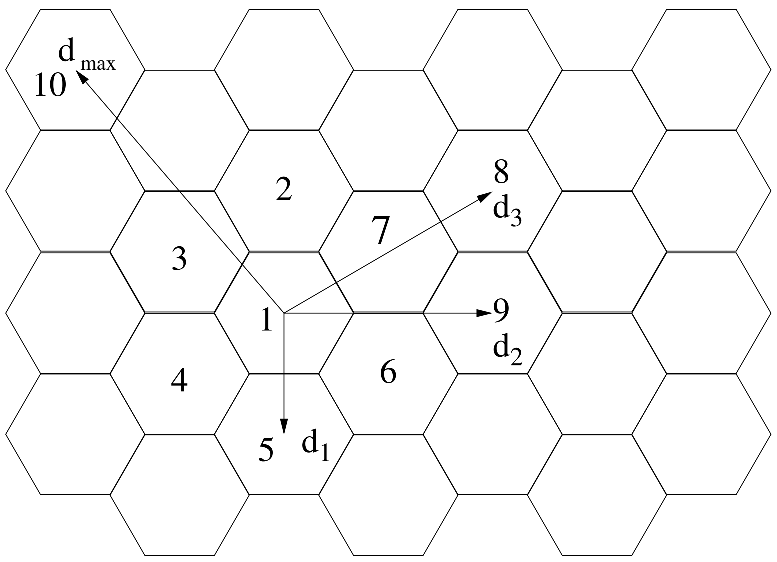

Figure 5.1: Location-dependent modeling with the $T(i)$ of grid cell $i$ defined as its seven neighbor cells.

This concept of virtual random variable helps to model the spatial correlation. Two cells close to each other will share more common spatial random variables, which means the correlation is strong. On the other hand, two cells physically far away from each other will share less or no common spatial random variables. In this way, the spatial correlation is modeled as a *homogeneous and isotropic random field* and the spatial correlation is only related to distance. That is to say, spatial correlation can be fully described by $\rho(d)$ in (3.11). $d_{max}$ is the distance beyond which $\rho(d)$ becomes small enough to be approximated as zero.

Since $\rho(d)$ is only a function of distance, the number of unique distance values between two correlated grid cells equals the number of unique element values in $\Omega_N$. From Fig. 5.1, the spatial correlation distance equals to the distance between cell 1 and cell 10 which is $d_{max} = \sqrt{7}d_c$, and there are only three unique correlation distances $d_1$ to $d_3$. Correspondingly, there are only three unique elements in $\Omega_N$, without including two special values: 0 for $d \geq d_{max}$ or 1 for distance within one cell.

Furthermore, the same correlation index can be used for all grid cells and the coefficient $\alpha_k$ should be the same for the same distance because of the homogeneousness and isotropy of spatial correlation. For the cell marked 1 in Fig. 5.1, we only have two unique values among the seven coefficients, i.e., we set $p_0 = \alpha_1$, $p_1 = \alpha_i, i = 2, 3, \ldots, 7$. In other words, we have

$$\Delta L_1 = p_0\xi_1 + p_1(\xi_2 + \ldots + \xi_7). \tag{5.3}$$

In this way, although there are seven random variables involved in the neighbor set, there are only two unknown coefficients left in the linear function in (5.3) due to the symmetry property of hexagonal partition.

According to (3.11), a nonlinear over-determined system can be built to determine the two unique values of $p_0$, $p_1$ as follows,

$$\begin{aligned}\rho(0) &= E(\Delta L_1^2) = p_0^2 + 6p_1^2 \\ \rho(d_1) &= E(\Delta L_1, \Delta L_2) = 2p_0p_1 + 2p_1^2 \\ \rho(d_2) &= E(\Delta L_1, \Delta L_9) = 2p_1^2 \\ \rho(d_3) &= E(\Delta L_1, \Delta L_8) = p_1^2\end{aligned} \tag{5.4}$$

The system in (5.5) can be solved by formulating them as a non-linear least square optimization problem. In the matrix form, we can rewrite (5.2) for a whole chip as

$$\Delta L = P_{N,N} \cdot \vec{\xi}, \tag{5.5}$$

where $N$ is the number of grid cells, and $\vec{\xi} = [\xi_1, \xi_2, \ldots, \xi_N]$. According to (5.2), the correlation index set contains only $k$ spatial random variables, which is a very small fraction of the total spatial random variables. As a result, $P_{N,N}$ is a sparse matrix. Every gate only is concerned with $k$ virtual random variables, which has specific location information.

Fundamentally, PCA-based method performs a similar process and has a similar new transformation matrix between the original and new set of variables:

$$\Delta L = V_{n,n} \cdot \vec{\xi}, \tag{5.6}$$

where $V_{n,n}$ is the transformation matrix obtained from eigen-value decomposition of the correlation matrix in PCA. The major difference is that $V_{n,n}$ is a dense matrix even though the original correlation matrix is sparse. This makes a huge difference especially when the spatial correlation is weak as eigen-decomposition will take almost

$O(n^3)$ to compute. The virtual independent spatial correlation model also works for medium and strong correlation cases, which will be shown in the next section.

## 5.3 Linear chip-level leakage power analysis method

In this section, we will present the new full-chip statistical leakage analysis method. We first introduce the overall flow of the proposed method and highlight the major computing steps. The new algorithm flow is summarized in Fig. 5.2.

**Algorithm**: New full-chip statistical leakage anlaysis.

**Input**: standard cell lib, netlist, placement information of design, standard deviation of $L$ and $T_{ox}$.
**Output**: analytic expression of the full-chip leakage currents in terms of Hermite polynomials.

1. Generate $a_1$ through $a_5$ for $I_{sub}$ and $I_{gate}$ in (3.1) and (3.2) for each type of gates (Section 3.2).
2. Solve (5.5) to determine coefficients in (5.3).
3. Calculate the coefficients of Hermite polynomial of $I_{sub}$ and $I_{gate}$ for the leakage analytic expression for each gate.
4. Calculate the analytic expression of the full-chip leakage current by simple polynomial additions and calculate mean value, standard deviation, PDF and CDF of the leakage current if required.

Figure 5.2: The flow of proposed algorithm.

The new algorithm consists of three major parts. The first part (Step 1 and 2) is pre-characterization. Step 1 builds the analytic leakage expressions (3.1) and (3.2) for each type of gates, which only needs to be done once for a standard cell library. Step 2 deals with a small-sized non-linear over-determined system, which can be solved with any least-square optimization algorithm. The second part (Step 3) generates a small set of independent virtual random variables and builds the analytic leakage current expressions and covariances for each gate on top of the new random variables. The final part (Step 4) computes the final full-chip leakage expressions by simple polynomial additions. From the final expressions, we can calculates important statistical information (like mean, variance, and even the whole distributions). In the following, we briefly explain some important steps.

### 5.3.1 Computing gate leakage by the orthogonal polynomial method

In the following, we use the orthogonal polynomial based modeling approaches mentioned in 2.2.3. Note that for Gaussian and log-normal distributions, Hermite polynomial is the best choice as it leads to exponential convergence rate [20]. For non Gaussian and non log-normal distributions, there are other orthogonal polynomials. The proposed method can be extended to other distributions with different orthogonal polynomials.

In our problem, $y(\vec{\xi})$ in (2.22) will be the leakage current for each gate, and eventually for the full chip. For the $j$th gate, from (5.2), $\Delta L_j$ only relates to $k$ independent virtual random variables in $T(j)$. Since $k$ is a small number, Step 3 in Fig. 5.2 can be very efficient.

To compute the gate leakage current, we need to present both $I_{sub}$ and $I_{gate}$ of each gate in the second-order Hermite polynomials, respectively:

$$
\begin{aligned}
I_{sub}(\vec{\xi}_{grid_j}) &= \sum\nolimits_{i=0}^{P} I_{sub,i,j} H_i(\vec{\xi}_{grid_j}), && (5.7)\\
I_{gate}(\vec{\xi}_{grid_j}) &= \sum\nolimits_{i=0}^{P} I_{gate,i,j} H_i(\vec{\xi}_{grid_j}), && (5.8)
\end{aligned}
$$

where $H_i(\vec{\xi}_{grid_j})$ are second-order Hermite polynomials defined as in (4.14). And $I_{sub,i,j}$ and $I_{gate,i,j}$ are then computed by the numerical Smolyak quadrature method in (2.32).

Notice that the time complexity of computing leakage for a gate is $O(k^2)$. And the number of involved independent random variables $k$ is very small compared to total number of gates. The analytic expression is also functions of those involved random variables.

### 5.3.2 Computation of full-chip leakage currents

After the leakage currents are calculated for each gate, we can proceed to compute the leakage current for the whole chip as follows:

$$I_{chip}(\vec{\xi}) = \sum\nolimits_{j=1}^{n} (I_{sub}(\vec{\xi}_{grid_j}) + I_{gate}(\vec{\xi}_{grid_j})). \tag{5.9}$$

The summation is done for each coefficient of Hermite polynomials. Then we obtain the analytic expression of the final leakage currents in terms of $\vec{\xi}$.

We can then obtain the mean value and variance of full-chip leakage current very easily as follows,

$$\begin{aligned} \mu_{chip} &= I_{chip,0th}, \quad (5.10) \\ \sigma^2_{chip} &= \sum I^2_{chip,1st} + 2\sum I^2_{chip,2nd,type1} \\ &\quad + \sum I^2_{chip,2nd,type2}, \quad (5.11) \end{aligned}$$

where $I_{chip,ith}$ is the leakage coefficient for $i$th Hermite polynomial of second order defined in (4.14). Since Hermite polynomials with orders higher than two have no contribution to mean value or standard deviation. Second order is good enough for estimating $\mu_{chip}$ and $\sigma_{chip}$ in (5.10) and (5.11).

### 5.3.3 Time complexity analysis

To analyze the time complexity, one typically does not count the pre-characterization cost of Step 1 in Fig. 5.2, and the time cost of Step 2 is ignorable compared to the following steps. In Step 3, we need to compute the weights of Level 2 $k$-dimensional Smolyak quadrature point set. For quadratic model with $k+3$ variables, the number of Smolyak quadrature points is $S \sim O(k^2)$ based on the discussion in Section 5.3.1. So the time cost for generating Smolyak quadrature points set is $O(k^2)$. In step 4, we need to call (3.1) and (3.2) $S$ times for each gate. In each call, we need to compute $k+3$ variables in the Hermite polynomials. The computational cost for the two steps is $(O(nk \times S))$, where $n$ is the number of gates. After the leakage currents are computed for each gate, it takes $O(n(k+3))$ to compute the full-chip leakage current.

For the second order Hermite polynomials, $S \propto k^2$, and the $k$ is the number of grid cells in the correlated neighbor index set, which is a very small constant number. As a result, the time complexity of our approach becomes linearly – $O(n)$.

## 5.4 New statistical leakage characterization in SCL

In this section, we will present why a new characterization modeling statistical leakage can be added to SCL, and how it can be applied in our new full-chip statistical leakage analysis method.

### 5.4.1 Acceleration by look-up table approach

The spatial correlation in (5.2) is related to distance between two grid cells. As a result, neighbor set $T(i)$ represents the relative location, not the absolute location. In other words, a local neighbor set $T$ and a local set of variables $\vec{\xi}_{loc} = [\xi_1, \ldots, \xi_k]$ can be shared by all the gates in all the cells.

The local neighbor set $T$ and the coefficients in (5.2) are determined by $d_{max}/d_c$. From the specific spatial correlation model in (3.11), (as shown in Fig. 5.3),

$$d_{max} = \eta\sqrt{-\ln(\rho_{low})},\ d_c = \eta\sqrt{-\ln(\rho_{high})}, \tag{5.12}$$

then the ratio of spatial correlation distance $d_{max}$ over grid length $d_c$ becomes

$$d_{max}/d_c = \sqrt{ln(\rho_{low})/ln(\rho_{high})}. \tag{5.13}$$

Once the threshold values $\rho_{high}$ and $\rho_{low}$ are set, $d_{max}/d_c$ is not related to the correla-

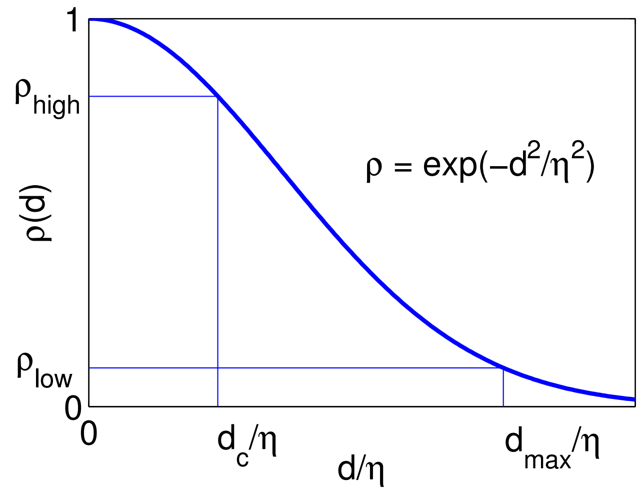

Figure 5.3: Relation between $\rho(d)$ and $d/\eta$.

tion length $\eta$. This means we can determine the grid length once we know the spatial correlation distance for a specific correlation formula at cost of controlled errors (by $\rho_{high}$ and $\rho_{low}$).

Furthermore, (5.13) shows the spatial correlation (strong or weak) has nothing to do with $T$ and the virtual random variables used in our model. At the same time, the fitting parameters of static leakage in (3.1) and (3.2) is only related to the types of gates in a library. As a result, the coefficients of Hermite polynomials for the leakage of one gate are only functions of the type of the gate, $\rho_{high}$ and $\rho_{low}$. Therefore, a simple look-up table can be used to store the coefficients of Hermite polynomials of each *type* of gates in the library. In other words, we do not need compute the coefficients of Hermite polynomials for each gate, just look them up from table instead. This makes a big difference, as the time complexity is reduced from $O(n)$ to $O(N)$, where $n$ is the number of gates and $N$ is the number of grid cells on chip.

For the look-up table, suppose $Q$ is the number of Hermite polynomials involved and $m$ is the number of gate types in the library, then it includes two matrices as follows:

$$C_S = \{I_{sub,q,j}\}, C_G = \{I_{gate,q,j}\}. \tag{5.14}$$

Here $I_{sub,q,j}$ represents the coefficient of $H_q$ for $j$th kind of gate in the library for subthreshold leakage; and $I_{sub,q,j}$ represents the coefficient of $H_q$ for $j$th kind of gate in the library for gate oxide leakage. $C_S$ and $C_G$ are $Q \times m$ matrices. Notice the table needs to only be built once and can be reused for different designs with different conditions of spatial correlations since the new algorithm is independent of spatial correlation length $\eta$ or the circuit design information. In this way, the look-up table actually builds a new characterization in SCL, which present the statistical leakage behavior of each standard cell.

## 5.4.2 Enhanced algorithm

The enhanced new algorithm consists of two parts. The first part is pre-characterization as shown in Fig. 5.4. We build analytic leakage current expressions for each kind of gate on top of a small set of independent virtual random variables. For fixed values of $\rho_{high}$, $\rho_{low}$ and one library, a new characterization is added to the SCL by building a look-up table, which stores coefficients of Hermite polynomials of $I_{sub}$ and $I_{gate}$ for the leakage analytic expressions for each kind of gate. This process only needs to be done once for one LIBRARY, given $\rho_{high}$ and $\rho_{low}$. Besides, it involves a small-size non-linear over-determined problem, which can be solved fast with any least-square algorithm.

**Algorithm**: Characterization of statistical leakage information in SCL

**Input**: standard cell lib, $\rho_{high}$, $\rho_{low}$.
**Output**: look-up table for coefficients of Hermite polynomials of $I_{sub}$ and $I_{gate}$ for the leakage analytic expressions for each kind of gate.

```
1. Generate fitting parameter matrices $a_{sub}$ and $a_{gate}$ of $I_{sub}$ and $I_{gate}$
   in (3.1) and (3.2) for each type of gates (after SPICE simulation
   on each input pattern) (Section 3.2).
2. Calculate $d_{max}/d_c$ from $\rho_{high}$ and $\rho_{low}$ to determine the neighbor
   set. And then solve (5.5) to determine coefficients in (5.3).
3. Generate Smolyak quadrature points set $\Theta_z^2$ with corresponding
   weights.
4. Calculate the coefficients of Hermite polynomials of $I_{sub}$ and $I_{gate}$
   for the leakage analytic expressions for each kind of gate in library.
```

Figure 5.4: The flow of statistical leakage characterization in SCL.

When we deal with full-chip statistical leakage analysis, the coefficients of local Hermite polynomials in the neighbor grid cell set for each cell can be simply calculated by the look-up table. After transferring the local coefficients to corresponding global positions, we can compute the final full-chip leakage expressions by simple polynomial additions. From the resulting expression, we can calculate other statistical information (like mean, variance, and even the whole distributions). The new algorithm flow is summarized in Fig. 5.5. In the following, we briefly explain some important steps.

**Algorithm**: NEW FULL-CHIP STATISTICAL LEAKAGE ANALYSIS ALGORITHM.

**Input**: Look-up table for coefficients of Hermite polynomial of $I_{sub}$ and $I_{gate}$ for the leakage analytic expression for each kind of gate. netlist, placement information of design, standard deviation of $L$ and $T_{ox}$.
**Output**: analytic expression of the full-chip leakage currents in terms of Hermite polynomials.

1. For every grid cell on chip, calculate the coefficients of local Hermite polynomials in the neighbor cell set by the look-up table.
2. Transfer the local coefficients to their corresponding global positions.
3. Calculate the analytic expression of the full-chip leakage current by simple polynomial additions and calculate mean value, standard deviation, PDF and CDF of the leakage current if required.

Figure 5.5: The flow of the proposed algorithm using statistical leakage characterization in standard cell library.

### 5.4.3 Computation of full-chip leakage currents

Here we define a gate mapping matrix as follows

$$G_{N\times m} = \{g_{i,j}\}, \tag{5.15}$$

where $g_{i,j}$ represents the number of $j$th kind of gate in library located in $i$th grid cell. Then the coefficients of local Hermite polynomials in neighbor set for all the cells on chip can be easily calculated by the look-up table as follows,

$$I_{sub,loc} = G \cdot C_S^T, \quad I_{gate,loc} = G \cdot C_G^T \tag{5.16}$$

In order to get the full-chip leakage current, the local coefficients need to be transferred to their corresponding global positions

$$T(i) = (x_i, y_i) + T. \tag{5.17}$$

For the $i$th grid cell, the local set of random variables $\vec{\xi}_{loc}$ should be transferred to the corresponding positions in $T(i)$. Therefore, $I_{sub,loc}$ and $I_{gate,loc}$ can be transferred to the corresponding global coefficients based on the global virtual random variable set $\vec{\xi}$. For example, the coefficient of $\xi_i$ in the $i$th cell is

$$I_{sub}(\xi_i) = \sum_{k, i \in T(k)} I_{sub,loc}(\xi_{T(k)-(x_k,y_k)}) \tag{5.18}$$

Next, we can proceed to compute the leakage current of the whole chip as follows,

$$I_{chip}(\vec{\xi}) = \sum I_{sub}(\vec{\xi}) + I_{gate}(\vec{\xi}) \tag{5.19}$$

The summation is done for each coefficient of global Hermite polynomials to obtain the analytic expression of the final leakage currents in terms of $\vec{\xi}$. We can then obtain the mean value, variance, PDF and CDF of the leakage current very easily. For instance, the mean value and variance for the full-chip leakage current are

$$\begin{aligned}\mu_{chip} &= I_{chip,0th}, &(5.20)\\ \sigma^2_{chip} &= \sum I^2_{chip,1st} + 2\sum I^2_{chip,2nd,type1} \\ &\quad + \sum I^2_{chip,2nd,type2}, &(5.21)\end{aligned}$$

where $I_{chip,ith}$ is the leakage coefficient for $i$th Hermite polynomial of second order defined in (4.14).

### 5.4.4 Incremental leakage analysis

During the leakage-aware circuit optimizations, a few small changes might be made to the circuit. But we do not want to compute the whole chip leakage from scratch again. In this case, incremental analysis become necessary. In this section, we show how this can be done in our look-up table based framework.

For brevity, we only consider the case where one gate is changed. However, the proposed incremental approach can be easily extended to handle a number of gates.

Assume one gate located in the $i$th grid cell is changed (e.g. a $j$th type of gate is replaced by a $(j+1)$th type), resulting in:

$$I^{new}_{chip} = I^{old}_{chip} - I^{old}_{grid-i} + I^{new}_{grid-i} \tag{5.22}$$

where $I^{new}_{chip}$ and $I^{old}_{chip}$ denote the full-chip leakage currents after and before change, respectively; and $I^{old}_{grid-i}$ and $I^{new}_{grid-i}$ are the leakage currents in the $i$th grid cell before and after change, respectively.

As defined in (5.15), $g_{i,j}$ in gate mapping matrix represents the number of $j$th kind of gate in the library located in the $i$th cell on a chip. Therefore, we can quickly generate the new gate mapping matrix $G^{new}$ by updating only two elements in $G^{old}$

$$\begin{aligned} g_{i,j}^{new} &= g_{i,j}^{old} - 1, \\ g_{i,j+1}^{new} &= g_{i,j+1}^{old} + 1. \end{aligned} \tag{5.23}$$

In the incremental analysis processes, we can consider the updating part as a small circuit, in which there is only one grid cell (the $i$th cell on chip) and only two types of gates in the library (the $j$th and the $(j+1)$th). Then the updating gate mapping matrix is

$$G^{update} = [-1 \quad 1], \tag{5.24}$$

and look-up tables in (5.14) using in the small circuit are only

$$\begin{aligned} C_S^{update} &= [I_{sub,j}, I_{sub,j+1}] \\ C_G^{update} &= [I_{gate,j}, I_{gate,j+1}], \end{aligned} \tag{5.25}$$

where $I_{sub,j/(j+1)}$, $I_{gate,j/(j+1)}$ are the $j/(j+1)$th column in $C_S$ and $C_G$, respectively.

Compared to the size of the whole chip, the small circuit is much simpler and only contains a few terms. Therefore, updating the leakage distribution using (5.24) and (5.25) is much cheaper than the full-blown chip leakage analysis. More details on the incremental look-up table approach are not included in this paper due to the limited number of pages.

### 5.4.5 Time complexity analysis

Considering statistical leakage analysis of a certain chip, for each grid cell, we need to do a weighted sum up of $m$ kinds of gates in this cell for every coefficient in the neighbor set(size $k$). For quadratic model with $k$ variables, the number of coefficients is about $S \sim k^2$. So the time cost for this step is $O(k^2 \times m \times N)$, where $N$ is the number of cells. For transferring the local coefficients to their global positions and summing them up, the time cost is $O(N)$. Next, it takes $O(N)$ to compute the full-chip leakage current. Since $k$ and $m$ are very small constant numbers, as a result, the time complexity of our approach becomes $O(N)$.

### 5.4.6 Discussion of extension to statistical run-time leakage estimation

The leakage current for each input combination we obtained in 3.2 can be the leakage in stand-by mode (idle) as well as active mode (run-time), by choosing different selections of input vectors.

For idle leakage analysis, we take the average of the leakage currents of all the input combinations to arrive at analytic expression for each gate as in (5.26), in lieu of the dominant states used in [9]. The reason for keeping all input states is that the technology down-scaling narrows the gap between leakage under dominant states and others. Only consider one state in leakage analysis will lead to large error compared to the simulation results.

$$\begin{aligned} I_{sub}^{avg} &= \sum_{i \in all\ input\ states} P_i I_{sub,i}, \\ I_{gate}^{avg} &= \sum_{i \in all\ input\ states} P_i I_{gate,i}, \end{aligned} \tag{5.26}$$

where $P_i$ is the probability of input state $i$ and $I_{sub,i}$ and $I_{gate,i}$ is the subthreshold leakage and gate leakage value at input state $i$, respectively.

On the other hand, run-time leakage might change when a new input vectors is applied. By choosing the input state at gate-level under certain input vector, the final analytic expression for run-time leakage can be obtained. Notice that the size of the look-up table of run-time leakage is larger than the one used in idle-time leakage analysis. For run-time leakage, the analytic expressions of all input patterns cannot be combined and have to be stored separately.

The proposed statistical characterization in SCL is fast enough to make run-time leakage estimation under a series of input vectors possible. More details for statistical run-time leakage analysis is given in the following part.

Here we present a forward-looking way to extend the proposed method to handle run-time leakage current estimation. In traditional power analysis, leakage was considered important only in the idle time. However, as technology scales down, the growth of leakage power become significant even during run time for instance for computing the maximum power bound [16].

Run-time leakage, however, is input-signal dependent and changes each time the input signals change, which means it becomes time varying. As a result, the run-time leakage analysis will take extremely long time as we need to perform the statistical analysis for each input vector along the time domain. Fortunately, with the novel statistical characterization in SCL and the incremental approach discussed in Section 5.4.4, leakage analysis at each cycle is fast enough to make run-time leakage estimation possible.

In the following, we show how to extend the proposed statistical leakage method to handle the run-time leakage analysis. First, in the run-time leakage analysis, given the initial input vector and initial state of each gate on a chip, the initial leakage analysis can be done using the algorithm in Fig. 5.5. After that, every time the input vector changes, the input states of some gates on the chip will be updated. Instead of computing the chip-level leakage from the very beginning, the incremental technique discussed in Section 5.4.4 can be applied here to update the run-time leakage information. The flow of proposed statistical analysis of run-time leakage is shown in Fig 5.6.

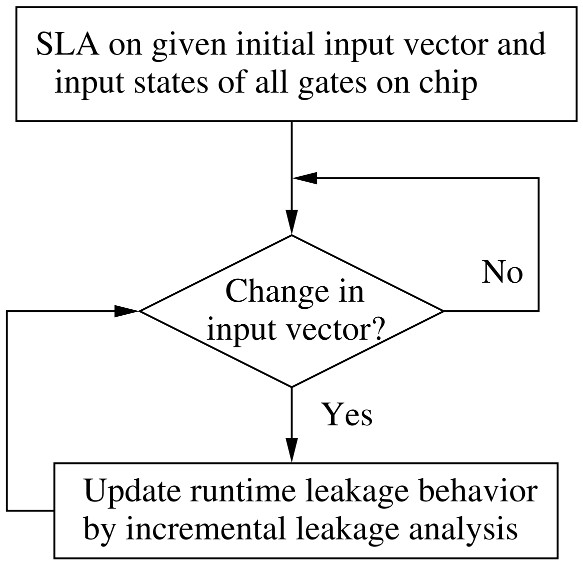
SLA on given initial input vector and input states of all gates on chip
Change in input vector?
No
Yes
Update runtime leakage behavior by incremental leakage analysis

Figure 5.6: Simulation flow for full-chip run-time leakage.

Also one notable difference is that the gate-level leakage analytical expressions in (3.1) and (3.2) for all input states need to be stored for run-time leakage analysis instead of the average value in (5.26) for idle-time leakage analysis.

Second, sometimes the maximum statistical run-time leakage estimation is required instead of such transient results of leakage. In fact, the maximum run-time leakage of a circuit can be much greater than the minimum leakage (by a few orders of magnitude [39]). Besides, the input vectors causing the maximum leakage current highly depends on process variations due to the shrinking physical dimensions.

To obtain the maximum statistical run time leakage, we follow the work in [16], which proposed a technique to accurately estimate the run-time maximum/minimum leakage vector considering both cell functionalities and process variations. One can first run the tool in [16] to obtain input vector giving the maximum leakage power first. Then one can apply the proposed SCL tool to obtain the maximum/minimum statistical leakage power under the input. The proposed statistical leakage characterization in SCL will work as long as the input vector is given.

We note that glitch events also have effect on run-time leakage power and the ignoring glitching can cause an estimation error of approximately 5-20% depending on circuit topology [39]. However, glitch has not been considered in any existing statistical run-time leakage analysis works so far and will be investigated in the future.

### 5.4.7 Discussion about run-time leakage reduction technique

Run-time leakage reduction technology such as power gating [3] is widely applied in design of mobile devices nowadays. Although the model of leakage power used in this work is idle-time leakage, the proposed method can be extended to leakage computation under the run-time scenario with leakage reduction.

By shutting off the idle blocks, power-gating is an effective technique for saving leakage power. Following the run-time leakage model for power-gating in [27], the variational part of full-chip leakage can be estimated as

$$I_{leak} = (1 - W) \sum_{i \in allgates} I_i^{gate}, \tag{5.27}$$

where $W$ is the empirical switching factor. And from [80], the leakage of a gate $I^{gate}$ can be approximated into a single exponential function of its virtual ground voltage ($V_{VG}$)

$$I^{gate} \approx \hat{I} e^{-K_{gate} V_{VG}}, \tag{5.28}$$

where $K_{gate}$ is the leakage reduction exponent, and $\hat{I}$ is zero-$V_{VG}$ leakage current. Notice both the switching factor $W$ in (5.27) and the leakage reduction exponent

$K_{gate}$ in (5.28) are related only to the type of gates and not to a statistical factor. Therefor, the proposed look-up table approach can work for both idle leakage and run-time leakage with power-gating activities.

## 5.5 Experimental results

The proposed methods with and without using look-up table have been implemented in Matlab 7.8.0. Since the leakage model for method in [82] has to be purely log-normal (linear terms in exponent parts), we did not choose it for comparing purpose. All the experimental results are carried out in a Linux system with quad Intel Xeon CPUs with 2.99Ghz and 16GB memory.

The methods for full-chip statistical leakage analysis were tested on circuits in the PDWorkshop91 benchmark set. The circuits were synthesized with Nangate Open Cell Library [53] and the placement is from MCNC [41]. The technology parameters come from the 45nm FreePDK Base Kit and PTM models [59].

Table 5.1: Summary of test cases used in this chapter.

| *Circuit* | *Gate #* | *Area/$\mu m^2$* | *Testcase* | $d_{max}/\mu m$ | $d_c/\mu m$ | *Grid #* |
|---|---|---|---|---|---|---|
| SC0 | 125 | 1459×1350 | Case 1 | 2190 | 730 | 2×2 |
| | | | Case 2 | 1095 | 365 | 4×4 |
| SC1 | 1888 | 4892×4874 | Case 3 | 1896 | 612 | 8×8 |
| | | | Case 4 | 918 | 328 | 16×16 |
| SC2 | 6417 | 10092×10466 | Case 5 | 984 | 328 | 32×31 |
| | | | Case 6 | 482 | 164 | 64×64 |
| VLSI | 2e6 | SC2 × 256 | Case 7 | 6301 | 2144 | 112×112 |

According to [1], $L$ and $T_{ox}$ for high performance logic in 45nm technology will be 18nm and 1.8nm, respectively. And the physical variation should be controlled within +/-12%. So the $3\sigma$ values of variations for $L$ and $T_{ox}$ were set to 12% of the nominal values, of which inter-die variations constitute 20% and intra-die variations, 80%. $L$ is modeled as sum of spatial correlated sources of variations, and $T_{ox}$ is modeled as an independent source of variation. The same framework can be easily extended to include other parameters of variations. Both $L$ and $T_{ox}$ are modeled as Gaussian parameters. For the correlated $L$, the spatial correlation is modeled based on (3.11), and the partition adopts Fig. 5.1. The test cases are given in Table 5.1 (all length units in $\mu m$), where test case "VLSI" is generated from duplicating SC2 as unit block to 16×16 array.

For comparison purposes, we performed Monte Carlo (MC) simulations with 50,000 runs using (3.1) and (3.2), the method in [9] (only consider spatial correlation of neighbor grid cells) and the proposed approaches on the benchmarks.

### 5.5.1 Accuracy and CPU time

The results of the comparison of mean value and standard deviations of full-chip leakage current are shown in Table 5.2, where *New* is the proposed method. The average errors for mean and standard variance ($\sigma$) values of the new technique are 4.52% and 3.92%, respectively. While for the method in [9], the average errors for mean value and $\sigma$ are 4.12% and 3.83%, respectively. Table 5.2, shows these two algorithms have almost the same accuracy and our method can handle both strong and weak spatial correlations by adjusting grid size. For very large circuit such as Case7 Monte Carlo and method in [9] run out of memory but the proposed method still works.

Table 5.3 compares the CPU times of MC, method in [9], proposed method(*New*), and proposed method using statistical leakage characterization in SCL (shorted as LUT). This table shows the new method, *New*, is much faster than the method in [9]

Table 5.2: Accuracy comparison of different methods based on Monte Carlo.

| *Test* | *Grid* | *Mean Value ($\mu A$)* | | | *Errors (%)* | |
|---|---|---|---|---|---|---|
| *Case* | *#* | MC | Method [9] | New | Method [9] | New |
| Case1 | $2 \times 2$ | 3.311 | 3.105 | 3.169 | -6.20 | -4.28 |
| Case2 | $4 \times 4$ | 3.310 | 3.105 | 3.169 | -6.20 | -4.28 |
| Case3 | $8 \times 8$ | 30.04 | 28.88 | 30.46 | -3.85 | -1.38 |
| Case4 | $16 \times 16$ | 30.04 | 28.88 | 30.46 | -3.85 | -1.38 |
| Case5 | $32 \times 32$ | 191.6 | 179.0 | 182.7 | -6.59 | -4.65 |
| Case6 | $64 \times 64$ | 191.6 | 179.0 | 182.7 | -6.59 | -4.65 |
| Case7 | $112 \times 112$ | – | – | 2.6e4 | – | – |

| *Test* | *Grid* | *Standard Deviation ($\mu A$)* | | | *Errors (%)* | |
|---|---|---|---|---|---|---|
| *Case* | *#* | MC | Method [9] | New | Method [9] | New |
| Case1 | $2 \times 2$ | 0.904 | 0.837 | 0.861 | -7.40 | -4.69 |
| Case2 | $4 \times 4$ | 0.594 | 0.547 | 0.548 | -7.91 | -7.74 |
| Case3 | $8 \times 8$ | 5.713 | 5.494 | 5.417 | -3.83 | -5.18 |
| Case4 | $16 \times 16$ | 5.307 | 5.400 | 5.067 | 1.75 | -4.52 |
| Case5 | $32 \times 32$ | 33.87 | 31.83 | 32.25 | -6.02 | -4.78 |
| Case6 | $64 \times 64$ | 33.20 | 30.27 | 29.34 | -8.83 | -11.63 |
| Case7 | $112 \times 112$ | – | – | 4.1e3 | – | – |

and MC simulation. On average, the proposed algorithm has about 113X speedup over [9] and many order of magnitudes over the MC method. And the speed of our approach is not affected by the total number of grid cells. If the spatial correlation is strong, which means $d_{max}$ is large, $d_c$ can be increased at the same time without loss of accuracy. So the number of neighbor grid cells in $T(i)$ will still be much smaller than the number of gates. The new method will be efficient and linear under both cases. Table 5.3 also shows proposed method can gain further speed up with look-up table technique using statistical leakage characterization in SCL.

## 5.5.2 Incremental analysis

For comparison purpose, one gate in each benchmark circuit is changed, and the proposed incremental algorithm is applied to update the leakage value locally. Table 5.4

Table 5.3: CPU time comparison.

| *Test Case* | MC | Method in [9] | New | LUT |
|---|---|---|---|---|
| Case1 | 83.14 | 2.96 | 0.10 | 0.023 |
| Case2 | 87.09 | 13.16 | 0.14 | 0.036 |
| Case3 | 828.42 | 26.24 | 0.86 | 0.033 |
| Case4 | 869.12 | 74.50 | 0.87 | 0.609 |
| Case5 | 7532.77 | 117.77 | 8.65 | 1.005 |
| Case6 | 7873.54 | 490.84 | 10.67 | 7.191 |
| Case7 | – | – | 2598 | 3.7313 |

shows the computational cost of the incremental analysis and the speedup over four different leakage analysis methods in Table 5.3. Compared with the look-up table approach (the fifth column in Table 5.3), the incremental analysis achieves $13 - 3.1e4$X speedup. As discussed in Section 5.4.4, the mini-circuit for updating only contains a small constant number of terms. Therefore, when the problem size increases further, we expect the incremental analysis could achieve more speedup over the full leakage analysis.

Table 5.4: Incremental leakage analysis cost.

| *Test* | *Cost time(s)* | *Speedup over* | | | |
|---|---|---|---|---|---|
| *Case* | Incremental LUT | MC | [9] | New | LUT |
| Case1 | 3.78e-4 | 2.2e5 | 2.7e4 | 265 | 53 |
| Case2 | 1.53e-4 | 5.7e5 | 8.1e4 | 915 | 157 |
| Case3 | 0.0026 | 3.2e5 | 3.7e4 | 331 | 13 |
| Case4 | 1.12e-4 | 7.8e6 | 6.7e5 | 7768 | 407 |
| Case5 | 0.0095 | 7.9e5 | 1.1e5 | 911 | 16 |
| Case6 | 2.77e-4 | 2.8e7 | 6.1e6 | 3.9e4 | 3.1e4 |

## 5.6 Summary

In this chapter, we have presented problem of statistical leakage analysis under process variations and spatial correlations. We then briefly discussed the existing approaches. Then we presented two statistical leakage analysis methods: the stochastic spectral based method with variable reduction techniques and the virtual grid based approach.

We have presented a linear algorithm for full-chip statistical analysis of leakage currents in the presence of any condition of spatial correlation (strong or weak). The new algorithm adopts a set of uncorrelated virtual variables over grid cells to represent the original physical random variables with spatial correlation and the size of grid cell is determined by the correlation length. As a result, each physical variable is always represented by virtual variables in local neighbor set. Furthermore, a look-up table is used to cache the statistical leakage information of each type of gate in the library to avoid computing leakage for each gate instance. As a result, the full-chip leakage can be calculated with $O(N)$ time complexity, where $N$ is the number of grid cells on chip. The new method maintains the linear complexity from strong to weak spatial correlation and has no limitation of leakage current model or variation model. This work also offers an incremental analysis capability to update the leakage distribution more efficiently when local changes to a circuit are made. Experimental results show the proposed method is about 1000X faster than the recently proposed method [9] with similar accuracy and many orders of magnitude times over the Monte Carlo method. Numerical results shows the proposed incremental analysis can further achieve significant speedup over the full leakage analysis.

# Chapter 6

# Statistical Capacitance Modeling and Extraction

## 6.1 Introduction

It is well accepted that the process-induced variability has huge impacts on the circuit performance in the sub-100nm VLSI technologies [51, 50]. The variational consideration of process has to be assessed in various VLSI design steps to ensure robust circuit design. Process variations consist of both systematic ones, which depend on patterns and other process parameters, and random ones, which have to be dealt with using stochastic approaches. Efficient capacitance extraction approaches by using the boundary element method (BEM) such as the fastCap [47], HiCap [66] and PHiCap [81] have been proposed in the past. To consider the variation impacts on the interconnects, one has to consider the RLC extraction processes of the three-dimensional structures modeling the interconnect conductors. In this chapter, we investigate the geometry variational impacts on the extracted capacitance.

Statistical extraction of capacitance considering process variations has been studied recently and several approaches have been proposed [34, 86, 28, 87, 85] under different variational models. Method in [34], uses analytical formulae to consider the variations in capacitance extraction and it has only first-order accuracy. The FastSies program considers the rough surface effects of the interconnect conductors[87]. It assumes only Gaussian distributions and has high computational costs. Method in [28] combines the hierarchical extraction and principle factor analysis to solve the capacitance statistical extraction.

Recently, a spectral stochastic collocation based capacitance extraction method was proposed [86, 84]. This approach is based on the Hermite orthogonal polynomial representation of the variational capacitance. It applies the numerical quadrature (collocation) method to compute the coefficients of the extracted capacitances in the Hermite polynomial form where the capacitance extraction processes (by solving the potential coefficient matrices) are performed many times (sampling). One of the major problems with this method is that many redundant operations are carried out (such as the set up of potential coefficient matrices for each sampling, which corresponds to solve one particular extraction problem). For the second-order Hermite polynomials, the number of samplings is O($m^2$), where $m$ is the number of variables. So if $m$ is large, the approach will lose it efficiency compared to the Monte Carlo method.

In this chapter, instead of using the numerical quadrature method, we use a different spectral stochastic method, where the Galerkin scheme is used. Galerkin-based spectral stochastic method has been applied for statistical interconnect modeling [74, 15] and on-chip power grid analysis considering process variations in the past [44, 42, 43]. The new method, called *statCap*, first transforms the original stochastic potential coefficient equations into a deterministic and larger one (via the Galerkin method) and then solves it using an iterative method. It avoids the less efficient sampling process in the existing collocation-based extraction approach. As a result, the potential coefficient equations and the corresponding augmented system only need to be set up once versus many times in the collocation based sampling method. This can lead to a significant saving in CPU time. Also the augmented potential coefficient system is sparse, symmetric and low-rank, which is further exploited by an iterative solver to gain further speedup. To consider second order effects, we derive the closed-form orthogonal polynomials for the capacitance integral equations directly in terms of variational variables without the loss of speed compared with the linear model. Experimental results show that the proposed method based on the first order and second order effects can deliver two orders of magnitude speedup over the collocation based spectral stochastic method and many orders of magnitude over the Monte Carlo method.

The main contributions of the this chapter are as follows:

1. Proposing the Galerkin-based spectral stochastic method to solve the statistical capacitance extraction problem where Galerkin scheme (versus the collocation method) is used to compute the coefficients of capacitances;

2. Deriving the closed-form coefficients hermite polynomial for potential coefficient matrices in both first order and second order forms

3. Studying the augmented matrix properties and showing that augmented matrix is still quite sparse, low rank and symmetric;

4. Solving the augmented systems by Minimum Residue Conjugate Gradient method [56] to take advantage of the sparsity, low rank and symmetric properties of the augmented matrices;

5. Comparing with the existing statistical capacitance extraction method based

on the spectral stochastic collocation approach [86] and Monte-Carlo method and showing the superiority of the proposed method.

We remark that we have put less emphasis on the acceleration techniques during the extraction processes such as the multiple-pole scheme [47], the hierarchical methods [66, 81], using the more sophisticated iterative solvers such as general minimal residue (GMRES) [61], which actually are the key components of those methods. The reason is that this is not the focus area where our major contributions are made. We believe those existing acceleration techniques can significantly speedup the proposed method as they did for the deterministic problem. This is especially the case for the hierarchical approach [66], the number of panels (thus the random variables) can be considerably reduced and the interaction between panels are constant. These are the areas for our future investigations.

## 6.2 Problem formulation

For $m$ conductors system, the capacitance extraction problem based on the Boundary Element Method (BEM) formulation is to solve the following integral equation [48]:

$$\int_S \frac{1}{\mid \overrightarrow{x_i} - \overrightarrow{x_j} \mid} \rho(\overrightarrow{x_j}) da_j = v(\overrightarrow{x_i}) \tag{6.1}$$

where $\rho(\overrightarrow{x_j})$ is the charge distribution on the surface at conductor $j$, $v(\overrightarrow{x_i})$ is the potential at conductor $i$ and $\frac{1}{|\overrightarrow{x_i}-\overrightarrow{x_j}|}$ is the free space Green function [1]. $da_j$ is the surface area on the surface $S$ of conductor $j$. $\overrightarrow{x_i}$ and $\overrightarrow{x_j}$ are point vectors. To solve for capacitances from one conductor to the rest of others, we set the conductor's potential to be one and all other $m-1$ conductors' potential to be zero. The resulting charges

[1]Note that the scale factor $1/(4\pi\epsilon_0)$ can be ignored here to simplify the notation and is used in the implementation to give results in units of farads.

computed are capacitances. BEM method divides the surfaces into $N$ small panels and assume uniform charge distribution on each panel, which transforms (6.1) into a linear algebraic equation

$$Pq = v \tag{6.2}$$

where $P \in R^{N\times N}$ is the potential coefficient matrix, $q$ is the charge on panels, $v$ is the pre-set potential on each panel. By solving above linear equation, we can obtain all the panel charges (thus capacitance values). In potential coefficient matrix $P$, each element is defined as

$$P_{ij} = \frac{1}{s_j}\int_{S_j} G(\overrightarrow{x_i}, \overrightarrow{x_j})da_j \tag{6.3}$$

where $G(\overrightarrow{x_i}, \overrightarrow{x_j}) = \frac{1}{|\overrightarrow{x_i}-\overrightarrow{x_j}|}$ is the Green function of point source at $\overrightarrow{x_j}$. $S_j$ is the surface of panel $j$ and $s_j$ is the area of panel $j$.

Process variations introducing conductor geometry variations are reflected on the fact that the size of panel and distances between panels become random variables. Here we assume the panel is still a two-dimensional surface. These variations will make each element in capacitance matrix follow some kinds of random distributions. The problem we need to solve now is to derive this random distribution and then to effectively compute the mean and variance of involved capacitance given geometry randomness parameters.

In this chapter, we follow the variational model introduced in [28], where each point in panel $i$ is disturbed by a vector $\Delta n_i$ that has the same direction as the normal direction of panel $i$.

$$\overrightarrow{x_i}' = \overrightarrow{x_i} + \Delta n_i \tag{6.4}$$

where the length of the $\Delta n_i$ follows Gaussian distribution $|\Delta n_i| \sim N(0, \sigma^2)$. If the value is negative, it means the direction of the perturbation is reversed. The correlation between random perturbation on each panel is governed by the empirical formulation such as the exponential model [88]

$$\gamma(r) = e^{-r^2/\eta^2} \tag{6.5}$$

where $r$ is the distance between two panel centers and $\eta$ is the correlation length.

The most straightforward method is to use Monte Carlo (MC) simulation to obtain distributions, mean values and variances of all those capacitances. But the MC method will be extremely time consuming as each sample run requires the formulation of the changed potential coefficient matrix $P$.

## 6.3 Review of spectral stochastic method

In this section, we briefly explain how to compute coefficients, the mean and variance from Hermite PCs using spectral stochastic or orthogonal polynomial chaos (PC) based stochastic analysis method in 2.2.2.

In case that $q(\xi)$ in (6.2) is unknown random variable vector (with normal distribution), then potential coefficient equation become

$$P(\xi)q(\xi) = v \tag{6.6}$$

Where both $P(\xi)$ and $q(\xi)$ are in Hermite PC form. Then the coefficients can be computed by using Galerkin method. The principle of orthogonality states that the best approximation of $v(\xi)$ is obtained when the error, $\Delta(\xi)$, defined as

$$\Delta(\xi) = P(\xi)q(\xi) - v \tag{6.7}$$

is orthogonal to the approximation. That is

$$< \Delta(\xi), H_k(\xi) >= 0,\ k = 0, 1, \ldots, P, \tag{6.8}$$

where $H_k(\xi)$ are Hermite polynomials. In this way, we have transformed the stochastic analysis process into a deterministic form, whereas we only need to compute the corresponding coefficients of the Hermite PC.

For the illustration purpose, considering two Gaussian variable $\xi = [\xi_1, \xi_2]$, we assume that the charge vector in panels can be written as a second order ($p = 2$) Hermite PC, we have

$$\begin{aligned} q(\xi) &= q_0 + q_1\xi_1 + q_2\xi_2 + q_3(\xi_1^2 - 1) + \\ &\quad q_4(\xi_2^2 - 1) + q_5(\xi_1\xi_2). \end{aligned} \tag{6.9}$$

which will be solved by using augmented potential coefficient matrices to be discussed in Section 6.4. Once the Hermite PC of $q(\xi)$ is known, the mean and variance of $q(\xi)$ can be evaluated trivially. Given an example, for one random variable, the mean and variance are calculated as:

$$\begin{aligned} E(q(\xi)) &= q_0 \\ Var(q(\xi)) &= q_1^2 Var(\xi) + q_2^2 Var(\xi^2 - 1) \\ &= q_1^2 + 2q_2^2. \end{aligned} \tag{6.10}$$

In consideration of correlations among random variables, we apply principal component analysis (PCA) to transform the correlated variables into a set of independent variables.

## 6.4 New orthogonal polynomial based extraction method: StatCap

In this section, we present our new spectral stochastic method based method, *StatCap*, which uses the orthogonal polynomials to represent random variables starting from the geometry parameters.

In our new method, we first represent the variation potential matrix $P$ into a first-order form using Taylor expansion. We then extend our method to handle the second-order variations in the Section 6.5.

### 6.4.1 Expansion of potential coefficient matrix

Specifically, each element in the potential coefficient matrix $P$ can be expressed as:

$$P_{ij} = \frac{1}{s_j} \int_{S_j} G(\vec{x_i}, \vec{x_j}) da_j \tag{6.11}$$

where $G(\vec{x_i}, \vec{x_j})$ is the free space Green function define in (6.3).

Notice that if panel $i$ and panel $j$ are far away (their distance is much larger than the panel area), we can have the following approximation [28]:

$$P_{ij} \approx G(\vec{x_i}, \vec{x_j}) \quad i \neq j \tag{6.12}$$

Suppose variation of panel $i$ can be written as $\Delta n_i = \delta i \ \vec{n_i}$ where $\vec{n_i}$ is the unit normal vector of panel $i$ and $\delta i$ is the scalar variation. Then take Taylor expansion on the Green function,

$$G(\vec{x_i} + \Delta n_i, \vec{x_j} + \Delta n_j) = \frac{1}{| \vec{x_i} - \vec{x_j} + \Delta n_i - \Delta n_j|} \tag{6.13}$$

$$= \frac{1}{\mid \vec{x_i} - \vec{x_j} \mid} + \nabla \frac{1}{\mid \vec{x_i} - \vec{x_j} \mid} \cdot (\Delta n_j - \Delta n_i) + O((\Delta n_i - \Delta n_j)^2) \tag{6.14}$$

From free space Green function, we have

$$\nabla G(\vec{x_i}, \vec{x_j}) = \nabla \frac{1}{\mid \vec{x_i} - \vec{x_j} \mid} = \nabla \frac{1}{\mid \vec{r} \mid} = \frac{\vec{r}}{\mid \vec{r} \mid^3} \tag{6.15}$$

$$\vec{r} = \vec{x_i} - \vec{x_j} \tag{6.16}$$

Now we first ignore the second-order terms to make the variation in the linear form. As a result, the potential coefficient matrix $P$ can be written as

$$P \approx P_0 + P_1 = \begin{pmatrix} G(\vec{x_1} + \Delta n_1, \vec{x_1} + \Delta n_1) & \dots & G(\vec{x_1} + \Delta n_1, \vec{x_n} + \Delta n_n) \\ G(\vec{x_2} + \Delta n_2, \vec{x_1} + \Delta n_1) & \dots & G(\vec{x_2} + \Delta n_2, \vec{x_n} + \Delta n_n) \\ \vdots & \dots & \vdots \\ G(\vec{x_n} + \Delta n_n, \vec{x_1} + \Delta n_1) & \dots & G(\vec{x_n} + \Delta n_n, \vec{x_n} + \Delta n_n) \end{pmatrix} \tag{6.17}$$

where

$$P_0 = \begin{pmatrix} G(\vec{x_1}, \vec{x_1}) & G(\vec{x_1}, \vec{x_2}) & \dots & G(\vec{x_1}, \vec{x_n}) \\ G(\vec{x_2}, \vec{x_1}) & G(\vec{x_2}, \vec{x_2}) & \dots & G(\vec{x_2}, \vec{x_n}) \\ \vdots & \vdots & \dots & \vdots \\ G(\vec{x_n}, \vec{x_1}) & G(\vec{x_n}, \vec{x_2}) & \dots & G(\vec{x_n}, \vec{x_n}) \end{pmatrix}$$

$$P_1 = \begin{pmatrix} 0 & \dots \nabla G(\vec{x_1}, \vec{x_n}) \cdot (\Delta n_n - \Delta n_1) \\ \nabla G(\vec{x_2}, \vec{x_1}) \cdot (\Delta n_1 - \Delta n_2) \dots & \nabla G(\vec{x_2}, \vec{x_n}) \cdot (\Delta n_n - \Delta n_2) \\ \vdots \quad \dots & \vdots \\ \nabla G(\vec{x_n}, \vec{x_1}) \cdot (\Delta n_1 - \Delta n_n) \dots & 0 \end{pmatrix}$$

We can further write the $P_1$ as the following form:

$$P_1 = V_1 \cdot N_1 \cdot J_1 - J_1 \cdot N_1 \cdot V_1 \tag{6.18}$$

$$J_1 = \begin{pmatrix} 0 & \nabla G(\vec{x_1}, \vec{x_2}) & \dots & \nabla G(\vec{x_1}, \vec{x_n}) \\ \nabla G(\vec{x_2}, \vec{x_1}) & 0 & \dots & \nabla G(\vec{x_2}, \vec{x_n}) \\ \vdots & \vdots & \dots & \vdots \\ \nabla G(\vec{x_n}, \vec{x_1}) & \dots & \nabla G(\vec{x_n}, \vec{x_{n-1}}) & 0 \end{pmatrix}$$

$$N_1 = \begin{pmatrix} \vec{n_1} & 0 & \dots \\ 0 & \vec{n_2} & \dots \\ \vdots & \dots & \vdots \\ 0 & \dots & \vec{n_n} \end{pmatrix}$$

$$V_1 = \begin{pmatrix} \delta n_1 & 0 & \dots \\ 0 & \delta n_2 & \dots \\ \vdots & \dots & \vdots \\ 0 & \dots & \delta n_n \end{pmatrix}$$

where $J_1$ and $N_1$ are vector matrices and $V_1$ is a diagonal matrix.

To deal with spatial correlation, $P_1$ can be further expressed as a linear combination of the dominate and independent variables

$$\xi = [\xi_1, \xi_2, \dots, \xi_p] \tag{6.19}$$

through the principal component analysis (PCA) operation. As a result, $V_1$ can be further expressed as

$$\begin{pmatrix} \sum_{i=1}^{p} a_{1i}\xi_i & 0 & \dots \\ 0 & \sum_{i=1}^{p} a_{2i}\xi_i & \dots \\ \vdots & \dots & \vdots \\ \dots & 0 & \sum_{i=1}^{p} a_{ni}\xi_i \end{pmatrix} \tag{6.20}$$

Finally we can represent the $P_1$ as

$$P_1 = \sum P_{1i}\xi_i \tag{6.21}$$

where

$$P_{1i} = A_i \cdot N_1 \cdot J_1 - J_1 \cdot N_1 \cdot A_i \tag{6.22}$$

and

$$A_i = \begin{pmatrix} a_{1i} & 0 & \dots & 0 \\ 0 & a_{2i} & \dots & 0 \\ \vdots & \vdots & \dots & \vdots \\ 0 & \dots & 0 & a_{ni} \end{pmatrix} \tag{6.23}$$

## 6.4.2 Formulation of the augmented system

Once the potential coefficient matrix is represented in the affine form as shown in (6.21), we are ready to solve for the coefficients $P_{1i}$ by using the Galerkin method, which will result in a larger system with augmented matrices and variables.

Specifically, for $p$ independent Gaussian random variables
$\xi = [\xi_1, \dots, \xi_p]$, there are $K = 2p + p(p-1)/2$ first and second-order Hermite polynomials. $H_i(\xi)$ $i = 1, \dots, K$ represents each Hermite polynomial and $H_1 = \xi_1, \dots, H_p = \xi_p$.

So for the vector of variational potential variables $q(\xi)$, it can be written as:

$$q(\xi) = q_0 + \sum_{i=1}^{K} q_i H_i(\xi) \tag{6.24}$$

where each $q_i$ is a vector associated with one polynomial. So the random linear equation can be written as:

$$Pq = (P_0 + \sum_{i=1}^{p} P_{1i} H_i)(q_0 + \sum_{i=1}^{K} q_i H_i) = v \tag{6.25}$$

Expanding the equation and performing inner product with $H_i$ on both sides, we can derive new linear system equations:

$$(W_0 \otimes P_0 + \sum_{i=1}^{p} W_i \otimes P_{1i})Q = V \tag{6.26}$$

where $\otimes$ is the tensor product and

$$Q = \begin{pmatrix} q_0 \\ q_1 \\ \vdots \\ q_K \end{pmatrix}; \;\; V = \begin{pmatrix} v \\ 0 \\ \vdots \\ 0 \end{pmatrix} \tag{6.27}$$

and$W_i =$

$$\begin{pmatrix} < H_i H_0 H_0 > & < H_i H_0 H_1 > & \dots & < H_i H_0 H_K > \\ < H_i H_1 H_0 > & < H_i H_1 H_1 > & \dots & < H_i H_1 H_K > \\ \vdots & \vdots & < H_i H_l H_m > & \vdots \\ < H_i H_K H_0 > & < H_i H_K H_1 > & \dots & < H_i H_K H_K > \end{pmatrix} \tag{6.28}$$

where $< H_i H_l H_m >$ represents the inner product of three Hermite polynomial $H_i$, $H_l$, $H_m$. The matrix $(W_0 \otimes P_0 + \sum_{i=1}^{p} W_i \otimes P_{1i})$ in (6.26) is called the *augmented* *potential coefficient* matrix. Since $H_i$ are at most second order polynomials, we can quickly calculate every element in $W_i$ with a lookup table for any number of random variables.

We remark that matrices $W_i$ are very sparse due to the nature of the inner product. As a result, their tensor products with $P_{1i}$ will also lead to the very sparse augmented matrix in (6.26). As a result, we have the following observations regarding the structure of the $W_i$ and the augmented matrix.

1. Observation 1: $W_0$ is a diagonal matrix.

2. Observation 2: For $W_i$ matrices, $i \neq 0$, all the diagonal elements are zero.

3. Observation 3: All $W_i$ are symmetric and the resulting augmented matrix $W_0 \otimes P_0 + \sum_{i=1}^{p} W_i \otimes P_{1i}$ is also symmetric.

4. Observation 4: If one element at position $(l, m)$ in $W_i$ is not zero, i.e. $W_i(l, m) \neq 0$, then elements at the same position $(l, m)$ of $W_j$, $j \neq i$, must be zero. In other words,

$$W_i(l, m) \cdot W_j(l, m) = 0 \quad \text{when } i \neq j$$
$$\forall\ i, j = 1, \ldots, p \text{ and } \ l, m = 1, \ldots, K$$

Such sparse property can help save the memory significantly as we do not need to actually perform the tensor product as shown in (6.26). Instead, we can add all $W_i$ together and expand each element in the resulting matrix by some specific $P_{1i}$ during the solving process, as there is no overlap among $W_i$ for any element position.

As the original potential coefficient matrix is quite sparse, low rank, the augmented matrix is also low rank. As a result, the sparsity, low rank and symmetric properties can be exploited by iterative solvers to speed up the extraction process as shown in the experimental results. In our implementation, the Minimum Residue Conjugate Gradient method [56] is used as the solver since the augmented system is symmetric.

## 6.5 Second-order *StatCap*

In this section, we extend *StatCap* to consider second order perturbations. We show the derivation of the coefficient matrix element in second-order orthogonal polynomial from the geometric variables. As a result, the second order potential coefficient matrix can be computed very quickly. In our second-order *StatCap*, we consider both of the far field and near field cases when (6.11) is approximated.

### 6.5.1 Derivation of analytic second-order potential coefficient matrix

Each element in the potential coefficient matrix $P$ can be expressed as

$$\begin{aligned} P_{ij} &= \frac{1}{s_i s_j}\int_{S_i}\int_{S_j} G(\vec{x_i},\vec{x_j})da_i da_j \\ &\approx \frac{1}{s_j}\int_{S_j} G(\vec{x_i},\vec{x_j})da_j \quad (6.29) \\ &\approx \frac{1}{s_i}\int_{S_i} G(\vec{x_i},\vec{x_j})da_i \quad (6.30) \end{aligned}$$

where $G(\vec{x_i},\vec{x_j})$ is the free space Green function defined in (6.3).

We assume the same definitions for $\Delta n_i$, $\delta n_i$ and $\vec{n_i}$ as in Section 6.4. If we consider both first-order and second-order terms, we have the following Taylor expansion on

$P_{ij}$,

$$\begin{aligned}
&P_{ij}(\Delta n_i, \Delta n_j)\\
&= P_{i,j,0} + \nabla P_{ij} \cdot \Delta n_i + \nabla P_{ij} \cdot \Delta n_j\\
&+\Delta {n_j}^T \nabla^2 P_{ij} \Delta n_j + \Delta {n_i}^T \nabla^2 P_{ij} \Delta n_i\\
&+2\Delta {n_j}^T \nabla^2 P_{ij} \Delta n_i + O((\Delta n_i - \Delta n_j)^3)\\
&\approx P_{i,j0} + \tfrac{\partial P_{ij}}{\partial \Delta n_i}\delta n_i + \tfrac{\partial P_{ij}}{\partial \Delta n_j}\delta n_j\\
&+\tfrac{\partial^2 P_{ij}}{\partial \Delta {n_i}^2}\delta {n_i}^2 + \tfrac{\partial^2 P_{ij}}{\partial \Delta {n_j}^2}\delta {n_j}^2 + 2\tfrac{\partial^2 P_{ij}}{\partial \Delta n_i \Delta n_j}\delta n_i \delta n_j
\end{aligned} \tag{6.31}$$

And to deal with the spatial correlation, $\Delta n_i$ can be further expressed as a linear combination of the dominate and independent variables in (6.19) through the PCA operation. As a result,

$$\Delta n_i = \delta n_i \vec{n_i} = (a_{i1}\xi_1 + \ldots + a_{ip}\xi_p) \vec{n_i} \tag{6.32}$$

where $a_{iL}$ is defined in (6.20). After that, $P$ will be represented by a linear combination of Hermite polynomials

$$\begin{aligned}
P &= P_0 + \sum_{L=1}^{p} P_{1L}\xi_L + \sum_{L=1}^{p} P_{2L}(\xi_L^2 - 1)\\
&\quad + \sum_{L_1}^{L_1 \neq L_2}\sum_{L_2} P_{2L_1,L_2}\xi_{L_1}\xi_{L_2}
\end{aligned} \tag{6.33}$$

where $P_{2L}$ is the coefficient corresponding to the first type of second order Hermite polynomial, $\xi_L^2 - 1$; and $P_{2L_1,L_2}$ means the coefficient corresponding to the second type of second order Hermite polynomial, $\xi_{L_1}\xi_{L_2}(L_1 \neq L_2)$.

So for each element $P_{ij}$ in $P$, the coefficients of orthogonal polynomials can be computed as follows,

$$P_{ij,1L} = a_{iL}\frac{\partial P_{ij}}{\partial \Delta n_i} + a_{jL}\frac{\partial P_{ij}}{\partial \Delta n_j} \tag{6.34}$$

$$P_{ij,2L} = a_{iL}^2 \frac{\partial^2 P_{ij}}{\partial \Delta {n_i}^2} + a_{jL}^2 \frac{\partial^2 P_{ij}}{\partial \Delta {n_j}^2} \\ +2a_{iL}a_{jL}\frac{\partial^2 P_{ij}}{\partial \Delta n_j \Delta n_i} \tag{6.35}$$

$$P_{ij,2L_1,L_2} = 2a_{iL_1}a_{iL_2}\frac{\partial^2 P_{ij}}{\partial \Delta {n_i}^2} + 2a_{jL_1}a_{jL_2}\frac{\partial^2 P_{ij}}{\partial \Delta {n_j}^2} \\ +2(a_{iL_1}a_{jL_2} + a_{iL_2}a_{jL_1})\frac{\partial^2 P_{ij}}{\partial \Delta n_j \Delta n_i} \tag{6.36}$$

Hence we need to compute analytic expressions for the partial derivatives of $P_{ij}$ to obtain the coefficients of Hermite polynomials. The details of the derivations for computing the derivatives used in (6.34), (6.36) and (6.36) can be found in the appendix section.

## 6.5.2 Formulation of the augmented system

Similarly as Section 6.4, once the potential coefficient matrix is represented in the affine form as shown in (6.33), we are ready to solve the coefficients $P_{1L}$, $P_{2L}$ and $P_{2L_1,L_2}$ by using the Galerkin method.

In this case, $P$ in (6.33) now is rewritten as

$$P = P_0 + \sum_{i=1}^{p} P_{1i}H_i + \sum_{i=p+1}^{K} P_{2i}H_i \tag{6.37}$$

So after considering the first-order and second-order Hermite polynomials in $P$, the random linear equation can be written as:

$$Pq = (P_0 + \sum_{i=1}^{p} P_{1i}H_i + \sum_{i=p+1}^{K} P_{2i}H_i)(q_0 + \sum_{i=1}^{K} q_iH_i) = v \tag{6.38}$$

Expanding the equation and performing inner product with $H_i$ on both sides, we can derive a new linear system:

$$(W_0 \otimes P_0 + \sum_{i=1}^{p} W_i \otimes P_{1i} + \sum_{i=p+1}^{K} W_i \otimes P_{2i})Q = V \tag{6.39}$$

where $\otimes$ is the tensor product and $Q$ and $V$ is the same as in (6.27). and $W_i$ has the same definition as in (6.28).

Again, the matrix in the right-hand side of (6.39) is the *augmented potential coefficient* matrix for the second-order *StatCap*. Since $H_i$ are at most second order polynomials, we can still use lookup table to calculate every element in $W_i$ for any number of random variables.

Now we study the properties of augmented potential coefficient matrix. We review the features and observations we made for the first-order *StatCap*.

For $W_i$, which is a $K \times K$ matrix, where $K = p(p+3)/2$, the number of nonzero elements in $W_i$ is showed in Table 6.1. From Table 6.1, we can see that matrices $W_i$ for $i = 1, \ldots, K$ are still very sparse. As a result, their tensor products with $P_{1i}$ and $P_{2i}$ will still give rise to the sparse augmented matrix in (6.39).

For the four observations in Section 6.4 regarding the the structure of $W_i, i = p+1, \ldots, K$ and the augmented matrix, we find that all the observations are still valid except for Observation 2. As a result, all the efficient implementation and solving techniques mentioned at the end of the Section 6.4 can be applied to the second order method.

Table 6.1: Number of non-zero element in $W_i$

| | $i = 0$ | $1 \leq i \leq p$ | $p+1 \leq i \leq 2p$ | $2p+1 \leq i \leq K$ |
|---|---|---|---|---|
| # non-zero | K | 2p+2 | p+3 | 2p+4 |

## 6.6 Experimental results

In this section, we compare the results of the proposed first-order and second-order *statCap* methods against the Monte Carlo method and the SSCM method [86], which is based on the spectral stochastic collocation method. The proposed *statCap* methods have been implemented in Matlab 7.4.0. We use Minimum Residue Conjugate Gradient method as the iterative solver. We also implement the SSCM method in Matlab using the sparse grid package [32, 31]. We do not use any hierarchical algorithm to accelerate the calculation of the potential coefficient matrix for both *statCap* and SSCM. Instead, we use analytic formula in [76] to compute the potential coefficient matrices.

All the experimental results are carried out in a Linux system with Intel Quadcore Xeon CPUs with 2.99Ghz and 16GB memory.

We test our algorithm on six testing cases. The more specific running parameters for each testing cases are summarized in Table 6.2. In Table 6.2, $p$ is the number of dominate and independent random variables we get through PCA operation, and $MC\#$ means the times we run Monte Carlo method. The $2 \times 2$ bus are shown in Fig. 6.1, and three-layer metal plane capacitance is shown in Fig. 6.2. In all the experiments, we set standard deviation as 10% of the wire width and the $\eta$, the correlation length, as 200% of the wire width.

Table 6.2: The test cases and the parameters setting

| | 1x1 bus | 2x2 bus | 3-layer | 3x3 bus | 4x4 bus | 5x5 bus |
|---|---|---|---|---|---|---|
| Panel # | 28 | 352 | 75 | 720 | 1216 | 4140 |
| p | 10 | 15 | 8 | 21 | 28 | 35 |
| MC # | 10000 | 6000 | 6000 | 6000 | 6000 | 6000 |

First, we compare the CPU times of the four methods. The results are shown in Table 6.3. In the table, *StatCap(1st/2nd)* refer to the proposed first and second

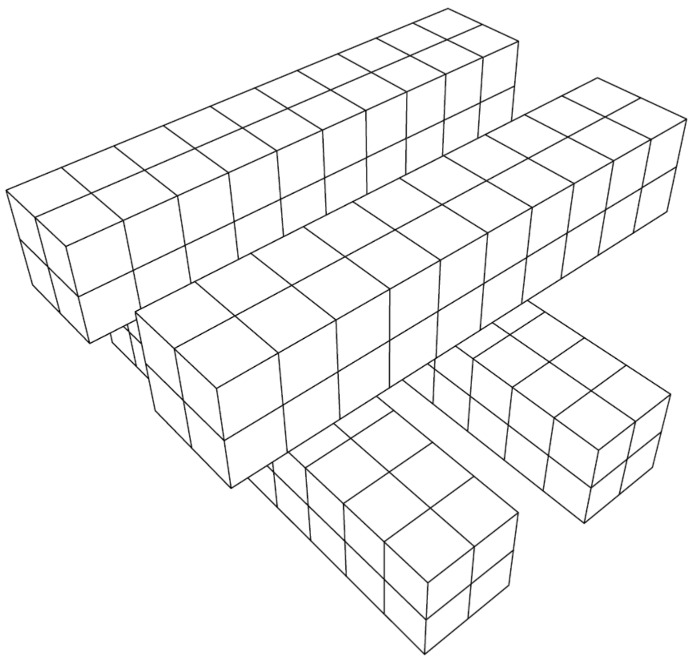

Figure 6.1: A $2 \times 2$ bus

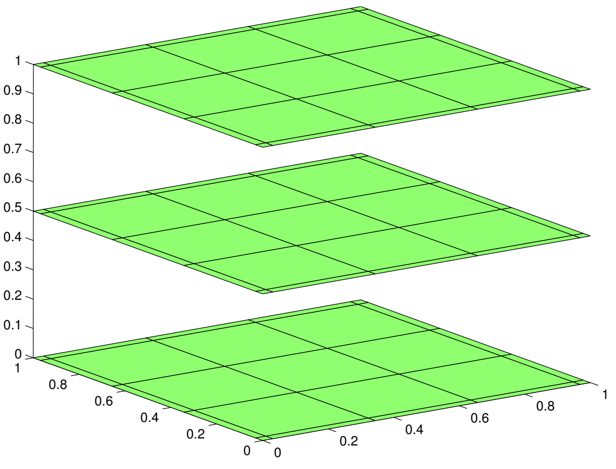

Figure 6.2: 3-layer metal planes

order methods respectively. *SP(X)* means the speed up of the first-order *StatCap* comparing with Monte Carlo or SSCM. All the capacitances are in pico-farad.

Table 6.3: CPU runtime (in seconds) comparison among MC, SSCM and *StatCap(1st/2nd)*

| $1 \times 1$ bus, MC(10000) | | | | | |
|---|---|---|---|---|---|
| MC | SSCM | *StatCap(1st)* | *StatCap(2nd)* | SP(MC) | SP(SSCM) |
| 2764s | 49.35s | 1.55s | 3.59s | 1783 | 32 |

| $2 \times 2$ bus, MC(6000) | | | | | |
|---|---|---|---|---|---|
| MC | SSCM | *StatCap(1st)* | *StatCap(2nd)* | SP(MC) | SP(SSCM) |
| 63059s | 2315s | 122s | 190s | 517 | 19 |

| 3-layer metal plane, MC(6000) | | | | | |
|---|---|---|---|---|---|
| MC | SSCM | *StatCap(1st)* | *StatCap(2nd)* | SP(MC) | SP(SSCM) |
| 16437s | 387s | 4.11s | 6.67s | 3999 | 94 |

| $3 \times 3$ bus, MC(6000) | | | | | |
|---|---|---|---|---|---|
| MC | SSCM | *StatCap(1st)* | *StatCap(2nd)* | SP(MC) | SP(SSCM) |
| $2.2 \times 10^5$s | 7860s | 408s | 857s | 534 | 19 |

| $4 \times 4$ bus, MC(6000) | | | | | |
|---|---|---|---|---|---|
| MC | SSCM | *StatCap(1st)* | *StatCap(2nd)* | SP(MC) | SP(SSCM) |
| –* | $3.62 \times 10^4$ | 1573s | 6855s | 260 | 23 |

| $5 \times 5$ bus, MC(6000) | | | | | |
|---|---|---|---|---|---|
| MC | SSCM | *StatCap(1st)* | *StatCap(2nd)* | SP(MC) | SP(SSCM) |
| –* | – | $1.7 \times 10^4$ | $6.0 \times 10^4$s | – | – |

* – out of memory

It can be seen that both the first and second order *StatCap* are much faster than both SSCM and the Monte Carlo method. And for large testing cases, such as the 5x5 bus case, Monte Carlo and SSCM will run out of memory, but *StatCap* still work well. For all the cases, *StatCap* can deliver about two-orders of magnitude speedup over the SSCM and three orders of magnitude speedup over Monte Carlo method. Notice that both SSCM and *StatCap* use the same random variables after PCA reduction.

We notice that both Monte Carlo and SSCM need to compute the potential coefficient matrices each time the geometry changes. This computation can be significant compared to the CPU time of solving potential coefficient equations. This is one of the reasons that SSCM and MC are much slower than *StatCap*, in which the augmented system only needs to be setup once.

Also SSCM uses the sparse grid scheme to reduce the collocation points in order to derive the orthogonal polynomial coefficients. But the number of collocation points are still in the order of $O(m^2)$ for the second-order Hermit polynomials, where $m$ is the number of variables. Thus it requires $O(m^2)$ solutions for the different geometries. In our algorithm, we also consider the second-order Hermit polynomials. But we only need to solve the augmented system once. The solving process can be further improved by using some advanced solver or acceleration techniques.

Next, we perform the accuracy comparison. The statistics for $1 \times 1$ bus case from the four algorithms are summarized in Table 6.4 and Table 6.5 for the mean value and standard deviation respectively. The parameter settings for each case is listed in Table 6.2. We make sure that SSCM, the first-order and The second-order *StatCap* use the same number of random variables after the PCA operations.

Table 6.4: Capacitance mean value comparison for the $1 \times 1$ bus

|  | MC | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
|---|---|---|---|---|
| C11 | 135.92 | 135.90 | 136.58 | 136.21 |
| C12 | -57.11 | -57.01 | -57.49 | -57.27 |
| C21 | -57.11 | -57.02 | -57.49 | -57.27 |
| C22 | 135.94 | 135.69 | 136.58 | 136.21 |

Table 6.5: Capacitance standard deviation comparison for the $1 \times 1$ bus

|  | MC | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
|---|---|---|---|---|
| C11 | 2.42 | 2.49 | 3.13 | 2.63 |
| C12 | 1.71 | 1.74 | 2.02 | 1.86 |
| C21 | 1.72 | 1.71 | 2.02 | 1.86 |
| C22 | 2.51 | 2.52 | 3.19 | 2.63 |

From these two tables, we can see that first-order *StatCap*, second-order *StatCap* and SSCM give the similar results for both mean value and standard deviation compared with the MC method. For all the other cases, the times we carry out Monte Carlo simulations are as shown in Table 6.3, and the similar experimental results can be obtained. The maximum errors and average errors of mean value and standard deviation for all the testing cases are shown in Table 6.6 and Table 6.7. Compare to the Monte Carlo method, the accuracy of the second-order *StatCap* is better than the first-order *StatCap* method, while from Table 6.3, the speed of second-order *StatCap* keeps in the same order as first-order *StatCap*, and is still much faster than SSCM and Monte Carlo.

## 6.7 Additional notes

In this appendix section, we detail the derivations for computing derivatives in (6.34), (6.36) and (6.36).

First, we consider the scenario where panel $i$ and panel $j$ are far away (their distance is much larger than the panel area). In this case, the approximations in (6.12) and (6.13) are still valid. From free space Green function, we have (6.15) and (6.16) for the first-order Hermite polynomails and we have the following for the second-order Hermite polynomails:

$$P_{ij,0} = \frac{1}{\mid \vec{x_i} - \vec{x_j} \mid} \tag{6.40}$$

$$\frac{\partial P_{ij}}{\partial \Delta n_i} = -\frac{\vec{r} \cdot \vec{n_i}}{\mid \vec{r} \mid^3} \tag{6.41}$$

$$\frac{\partial P_{ij}}{\partial \Delta n_j} = \frac{\vec{r} \cdot \vec{n_j}}{\mid \vec{r} \mid^3} \tag{6.42}$$

Table 6.6: Error comparison of capacitance mean values among SSCM, and *StatCap*(1st and 2nd order)

| | | | |
|---|---|---|---|
| $1 \times 1$ bus, MC(10000) as standard | | | |
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 0.19% | 0.67% | 0.28% |
| Avg err | 0.14% | 0.57% | 0.24% |
| $2 \times 2$ bus, MC(6000) as standard | | | |
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 0.32% | 0.49% | 1.19% |
| Avg err | 0.15% | 0.24% | 0.89% |
| 3-layer metal plane, MC(6000) as standard | | | |
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 0.30% | 1.84% | 0.81% |
| Avg err | 0.14% | 0.90% | 0.58% |
| $3 \times 3$ bus, MC(6000) as standard | | | |
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 0.33% | 0.81% | 0.43% |
| Avg err | 0.11% | 0.58% | 0.11% |
| $4 \times 4$ bus, SSCM as standard | | | |
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 0 | 0.76% | 0.35% |
| Avg err | 0 | 0.40% | 0.09% |
| $5 \times 5$ bus, *StatCap(2nd)* as standard | | | |
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | – | 0.59% | 0 |
| Avg err | – | 0.28% | 0 |

$$\frac{\partial^2 P_{ij}}{\partial \Delta {n_i}^2} = \frac{3(\vec{r} \cdot \vec{n_i})^2}{|\vec{r}|^5} - \frac{1}{|\vec{r}|^3} \tag{6.43}$$

$$\frac{\partial^2 P_{ij}}{\partial \Delta {n_j}^2} = \frac{3(\vec{r} \cdot \vec{n_j})^2}{|\vec{r}|^5} - \frac{1}{|\vec{r}|^3} \tag{6.44}$$

$$\frac{\partial^2 P_{ij}}{\partial \Delta n_j \Delta n_i} = \frac{-3(\vec{r} \cdot \vec{n_j})(\vec{r} \cdot \vec{n_i})}{|\vec{r}|^5} \tag{6.45}$$

Second, we consider the scenario where panel $i$ and panel $j$ are near each other (their distance is comparable with the panel area). In this case, the approximation

Table 6.7: Error comparison of capacitance standard deviations among SSCM, and *StatCap*(1st and 2nd order)

| $1 \times 1$ bus, MC(10000) as standard | | | |
|---|---|---|---|
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 2.48% | 29.34% | 8.77% |
| Avg err | 2.29% | 23.38% | 7.91% |

| $2 \times 2$ bus, MC(6000) as standard | | | |
|---|---|---|---|
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 14.28% | 12.98% | 25.99% |
| Avg err | 6.11% | 8.51% | 6.04% |

| 3-layer metal plane, MC(6000) as standard | | | |
|---|---|---|---|
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 8.35% | 16.26% | 2.38% |
| Avg err | 3.37% | 5.06% | 0.86% |

| $3 \times 3$ bus, MC(6000) as standard | | | |
|---|---|---|---|
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 23.32% | 21.39% | 11.75% |
| Avg err | 3.33% | 10.35% | 4.38% |

| $4 \times 4$ bus, SSCM as standard | | | |
|---|---|---|---|
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | 0 | 25.7% | 6.68% |
| Avg err | 0 | 16.1% | 3.89% |

| $5 \times 5$ bus, *StatCap(2nd)* as standard | | | |
|---|---|---|---|
| | SSCM | *StatCap(1st)* | *StatCap(2nd)* |
| Max err | – | 17.5% | 0 |
| Avg err | – | 7.92% | 0 |

in (6.12) is no longer accurate and we must consider the general form in (6.29) and (6.30).

Since panel $i$ panel $j$ are perpendicular to $\Delta n_i/\Delta n_j$, for $\frac{\partial P_{ij}}{\partial \Delta n_j}$ and $\frac{\partial^2 P_{ij}}{\partial \Delta {n_j}^2}$, with

(6.29), we have

$$\begin{aligned}
\frac{\partial P_{ij}}{\partial \Delta n_j} &\approx \frac{\partial \frac{1}{s_j}\int_{S_j} G(\overrightarrow{x_i}, \overrightarrow{x_j}) da_j}{\partial \Delta n_j} \\
&= \frac{\partial \frac{1}{s_j}\int_{S_j} \frac{1}{|\overrightarrow{x_i} - \overrightarrow{x_j} + \Delta n_i - \Delta n_j|} da_j}{\partial \Delta n_j} \\
&= \frac{1}{s_j}\int_{S_j} \frac{\partial \frac{1}{|\overrightarrow{x_i} - \overrightarrow{x_j} + \Delta n_i - \Delta n_j|}}{\partial \Delta n_j} da_j \\
&= \frac{1}{s_j}\int_{S_j} \frac{\overrightarrow{r} \cdot \overrightarrow{n_j}}{\mid \overrightarrow{r} \mid^3} da_j \\
&= \frac{\overrightarrow{r} \cdot \overrightarrow{n_j}}{s_j}\int_{S_j} \frac{1}{\mid \overrightarrow{r} \mid^3} da_j
\end{aligned} \tag{6.46}$$

$$\begin{aligned}
\frac{\partial^2 P_{ij}}{\partial \Delta {n_j}^2} &\approx \frac{\partial^2 \frac{1}{s_j}\int_{S_j} G(\overrightarrow{x_i}, \overrightarrow{x_j}) da_j}{\partial \Delta {n_j}^2} \\
&= \frac{\partial^2 \frac{1}{s_j}\int_{S_j} \frac{1}{|\overrightarrow{x_i} - \overrightarrow{x_j} + \Delta n_i - \Delta n_j|} da_j}{\partial \Delta {n_j}^2} \\
&= \frac{1}{s_j}\int_{S_j} \frac{\partial^2 \frac{1}{|\overrightarrow{x_i} - \overrightarrow{x_j} + \Delta n_i - \Delta n_j|}}{\partial \Delta {n_j}^2} da_j \\
&= \frac{1}{s_j}\int_{S_j} \frac{3(\overrightarrow{r} \cdot \overrightarrow{n_j})^2}{\mid \overrightarrow{r} \mid^5} - \frac{1}{\mid \overrightarrow{r} \mid^3} da_j \\
&= \frac{3(\overrightarrow{r} \cdot \overrightarrow{n_j})^2}{s_j}\int_{S_j} \frac{da_j}{\mid \overrightarrow{r} \mid^5} - \frac{1}{s_j}\int_{S_j} \frac{da_j}{\mid \overrightarrow{r} \mid^3}
\end{aligned} \tag{6.47}$$

Similarly, with (6.30), we can further obtain

$$\begin{aligned}
\frac{\partial P_{ij}}{\partial \Delta n_i} &\approx \frac{\partial \frac{1}{s_i}\int_{S_i} G(\vec{x_i},\vec{x_j})da_i}{\partial \Delta n_i} \\
&= \frac{-\vec{r}\cdot\vec{n_i}}{s_i}\int_{S_i}\frac{1}{|\vec{r}|^3}da_i \qquad (6.48)\\
\frac{\partial^2 P_{ij}}{\partial \Delta {n_i}^2} &\approx \frac{\partial^2 \frac{1}{s_i}\int_{S_i} G(\vec{x_i},\vec{x_j})da_i}{\partial \Delta {n_i}^2} \\
&= \frac{3(\vec{r}\cdot\vec{n_i})^2}{s_i}\int_{S_i}\frac{da_i}{|\vec{r}|^5} - \frac{1}{s_i}\int_{S_i}\frac{da_i}{|\vec{r}|^3} \qquad (6.49)
\end{aligned}$$

While for $\frac{\partial^2 P_{ij}}{\partial \Delta n_j \Delta n_i}$, we need to further consider two cases. First, when panel $i$ and panel $j$ are in parallel, we have

$$\frac{\partial^2 P_{ij}}{\partial \Delta {n_i}^2} = \frac{\partial^2 P_{ij}}{\partial \Delta {n_j}^2} = -\frac{\partial^2 P_{ij}}{\partial \Delta n_j \Delta n_i} \qquad (6.50)$$

Second, we consider panel $i$ and panel $j$ are not in parallel. Then we arrive

$$\begin{aligned}
\frac{\partial^2 P_{ij}}{\partial \Delta n_j \Delta n_i} &= \frac{\partial \frac{\partial P_{ij}}{\partial \Delta n_i}}{\partial \Delta n_j} \\
&= \frac{\partial(-\frac{\vec{r}\cdot\vec{n_i}}{s_i}\int_{S_i}\frac{1}{|\vec{r}|^3}da_i)}{\partial \Delta n_j} \\
&= -\frac{\vec{r}\cdot\vec{n_i}}{s_i}\frac{\partial\int_{S_i}\frac{1}{|\vec{r}|^3}da_i}{\partial \Delta n_j} \qquad (6.51)
\end{aligned}$$

Assume the conductors are rectangular geometries. Then two panels should be either in parallel or perpendicular. Since panel $i$ and panel $j$ are not parallel, these two panels will be perpendicular.

Without loss of generality, we assume that panel $i$ is in parallel with $xz$-plane, panel $j$ is in parallel with $yz$-plane. Then easily to see, $\vec{n_i}= (0, 1, 0)$ and $\vec{n_j}= (1, 0, 0)$.

Let $u_{kl}$, $k,l \in \{0,1\}$ denote the four corners of panel $i$, with $(x_{ik}, y_i, z_{il})$ being the Cartesian cooridinates of corner $u_{kl}$, and the center of gravity is $(x_i, y_i, z_i)$. Let $t_{kl}$, $k,l \in \{0,1\}$ denote the four corners of panel $j$, with $(x_j, y_{jk}, z_{jl})$ being the Cartesian cooridinates of corner $t_{kl}$, and the center of gravity is $(x_j, y_j, z_j)$.

After that, (6.51) can be further deduced to

$$
\begin{aligned}
\frac{\partial^2 P_{ij}}{\partial \Delta n_j \Delta n_i} &= \frac{y_j - y_i}{s_i} \frac{\partial \int_{x_i0}^{x_{i1}} \int_{z_{i0}}^{z_{i1}} \frac{dxdz}{|\vec{r}|^3}}{\partial x_j} \\
&= \frac{y_j - y_i}{s_i} \frac{\partial \int_{x_i0-x_j}^{x_{i1}-x_j} (\int_{z_{i0}}^{z_{i1}} \frac{dz}{|\vec{r'}|^3}) dx}{\partial x_j} \\
&= \frac{y_j - y_i}{s_i} (\int_{z_{i0}}^{z_{i1}} \frac{dz}{\left|\vec{r}^-\right|^3} - \int_{z_{i0}}^{z_{i1}} \frac{dz}{\left|\vec{r}^+\right|^3}) \\
&= \frac{y_j - y_i}{s_i} \sum_{k=0}^{1} \sum_{l=0}^{1} \Bigg( \frac{(-1)^{k+l+1}(z_{il} - z_j)}{((x_{ik} - x_j)^2 + (y_i - y_j)^2)} \\
&\times \frac{1}{\sqrt{(x_{ik} - x_j)^2 + (y_i - y_j)^2 + (z_{il} - z_j)^2}} \Bigg) \qquad (6.52)
\end{aligned}
$$

where

$$
\begin{aligned}
\vec{r} &= \sqrt{(x - x_j)^2 + (y_i - y_j)^2 + (z - z_j)^2} \\
\vec{r'} &= \sqrt{(x)^2 + (y_i - y_j)^2 + (z - z_j)^2} \\
\vec{r}^+ &= \sqrt{(x_{i1} - x_j)^2 + (y_i - y_j)^2 + (z - z_j)^2} \\
\vec{r}^- &= \sqrt{(x_{i0} - x_j)^2 + (y_i - y_j)^2 + (z - z_j)^2}
\end{aligned}
$$

## 6.8 Summary

In this chapter, we have proposed a novel statistical capacitance extraction method, called *statCap*, for three-dimensional interconnects considering process variations.

The new method is based on the orthogonal polynomial method to represent the variational geometrical parameters in a deterministic way. We consider both first order and second order variational effects. The new method avoids the sampling operations in the existing collocation-based spectral stochastic method. The new method solves an enlarged potential coefficient system to obtain the coefficients of orthogonal polynomials for capacitances. *statCap* only needs to set up the augmented equation once and can exploit the sparsity and low-rank property to speedup the extraction process. The new *statCap* method can consider second-order perturbation effects to generate more accurate quadratic variational capacitances. Experimental results show that our method is two orders of magnitude faster than the recently proposed statistical capacitance extraction method based on the spectral stochastic collocation method and many orders of magnitude faster than the Monte Carlo method for several practical interconnect structures.

# Chapter 7

# Voltage Binning Technique for Yield Optimization

## 7.1 Introduction

Process-induced variability has huge impacts on the circuit performance and yield in the nanometer VLSI technologies [2]. Indeed, the characteristics of devices and interconnects are prone to increasing process variability as device geometries getting close to the size of atoms. The yield loss from process fluctuations is expected to increase as the transistor size scaling down. As a result, improving yields considering the process variations is critical to mitigate the huge impacts from process uncertainties.

Supply voltage adjustment can be used as a technique to reduce yield loss, which is based on the fact that both chip performance and power consumption depend on supply voltage. By increasing supply voltage, chip performance improves. Both dynamic power and leakage power, however, will become worse at the same time [72]. In contrast, lower supply voltage will reduce the power consumption but make the chip slower. In other words, faster chips usually have higher power consumption and slower chips often come with lower power consumption. Therefore, it is possible to reduce yield loss by adjusting supply voltage to make some failing chips satisfy application constraints.

For yield enhancement, there are also different schemes for supply voltage adjustment. In [72], the authors proposed an adaptive supply voltage method for reducing impacts of parameter variations by assigning individual supply voltage to each manufactured chip. This methodology can be very effective but it requires significant effort in chip design and testing at many different supply voltages. Recently, a new voltage binning technique has been proposed by the patent [33] for yield optimization as an alternative technique of adaptive supply voltage. All manufactured chips are divided into several bins, and a certain value of supply voltage is assigned to each bin to make sure all chips in this bin can work under the corresponding supply voltage. At the cost of small yield loss, this technique is much more practical than the adaptive voltage supply. But only a general idea is given in [33], without details of selecting optimal supply voltage levels. Another recent work [89] provides a statistical technique of yield computation for different voltage binning schemes. From results of statistical timing and variational power analysis, the authors developed a combination of analytical and numerical techniques to compute joint probability density functions (PDFs) of chip yield as a function of inter-die variation in effective gate length $L$ , and solve the problem of computing optimal supply voltages for a given binning scheme.

However, the method in [89] only works under several assumptions and approximations that will cause accuracy loss in both yield analysis and optimal voltage binning scheme. The statistical model for both timing and power analysis used in [89] are simplified by integrating all process variations other than inter-die variation in $L$ to one random variable following Gaussian distribution. Indeed, the intra-die variations has a huge impact on performance and power consumption [5, 65]. And other process variations (gate oxide thickness, threshold voltage, etc) have different distributions and should not be simplified to only one Gaussian distribution. Furthermore, this technique cannot predict the number of voltage bins needed under certain yield requirement before solving the voltage binning problem.

In general, voltage binning for yield improvement becomes an emerging technique but with many unsolved issues. In this paper we propose a new voltage binning scheme to optimize yield. The new method first computes the set of working supply voltage segments under timing and power constraints from either the measurement of real chips or Monte Carlo based SPICE simulations on a chip with process variations. Then on top of the distribution of voltage segment lengths, we propose a formulate to predict the upper bound of bin number needed under uniform binning scheme for the yield requirement. Furthermore, we frame the voltage binning scheme as a set-cover problem in graph theory and solve it by a greedy algorithm in an incremental way. The new method is not limited by the number or types of process variabilities involved as it should be based on actual measured results. Furthermore, the new algorithm can be easily extended to deal with a range of working supply voltages for dynamic voltage scaling under different operations modes (like lower power and high performance modes).

Experimental results on a number of benchmarks under 45nm technology show that the proposed method can correctly predict the upper bound on the number of bins required. The optimal binning scheme can lead to significant saving for the number of bins compared to the uniform one to achieve the same yield with very small CPU cost.

## 7.2 Problem formulation

### 7.2.1 Yield estimation

A "good" chip needs to satisfy two requirements:

1) timing slack is positive $S > 0$ under working frequency.

2) power does not exceed the limit $P < P_{lim}$.

For a single voltage supply, the definition of parametric chip yield is the percentage of manufactured chips satisfying these constraints. Specifically, we compute yield for a given voltage level by direct integration in the space of process parameters:

$$Y = \underset{S>0, P<P_{lim}}{\int \cdots \int} f(\Delta\vec{X}_1, \ldots, \Delta\vec{X}_n) d\Delta\vec{X}_1 \ldots d\Delta\vec{X}_n \tag{7.1}$$

where $f(\Delta\vec{X}_1, \Delta\vec{X}_2, \ldots, \Delta\vec{X}_n)$ is the joint PDF of $\Delta\vec{X}_1$ to $\Delta\vec{X}_n$, which represents the process variations. Also there exists spatial correlation in the intra-die part of variation. Existing approach in [89] ignores the intra-die variation in process parameters, which means only one random variable for inter-die variation is considered. And all other variations except inter-die variation in $L_{eff}$ are integrated into one Gaussian random variable. In this way, the multi-dimensional integral in (7.1) can be modeled numerically as a 2 or 3 dimensional integral. However, the spatial correlation can have significant impacts on both statistical timing and statistical power of a circuit [8, 65], thus impacts on yield analysis also.

### 7.2.2 Voltage binning problem

We first define voltage binning scheme as in [89].

**Definition 7.2.1.** *A voltage binning scheme is a set of supply voltage levels* $\vec{V} =$

*$\{V_1, V_2, \ldots, V_k\}$, a set of corresponding bins $\vec{U} = \{U_1, U_2, \ldots, U_k\}$, which is also a partitioning of all chips, and a binning algorithm $B$, which distributes manufactured chips among the bins.*

The binning algorithm $B$ assigns chips to bins so that any chip in bin $U_i$ meets both the performance and power constraints at supply voltage level $V_i$ corresponding to $U_i$. The yield loss is constituted by chips which fail to be assigned to any bin in $\vec{U}$.

The definition of a voltage binning scheme depends on two factors: the bin voltage levels $\vec{V}$ and the binning algorithm $A$. Different binning algorithm will result in different yield even given the same bin voltage levels $\vec{V}$. However, in the optimization process, the focus is the binning algorithms which can produce the maximum possible yield. That is to say, in an optimal binning algorithm, there exists at least one voltage bin for any "good" chip (the chips satisfies performance and power constraints). In this way, the yield loss under bin voltage levels $\vec{V}$ will reach the maximum value.

Therefore, the problem of computing optimal voltage binning scheme can be formulated as follows:

$$\max_{\vec{V}} Y; \quad s.t. \qquad V_{min} \le V_i \in \vec{V} \le V_{max} \tag{7.2}$$

where $Y$ is the total yield under the optimal voltage binning scheme with supply voltage levels $\vec{V} = \{V_1, V_2, \ldots, V_k\}$.

We would like to mention one special type of voltage binning in which we have an infinite number of voltage bins with all possible voltage levels. This binning scheme allows the supply voltage to be individually tailored for each chip to meet timing and power constraints. It is obvious that the yield in this case is the maximum possible yield, named as $Y_{max}$, which should be an upper bound of yield for any other voltage binning scheme. As a result, for optimal solution, $k_{opt}$ should be the minimum number of bins that make $Y_{k,opt} = Y_{max}$.

## 7.3 Proposed new voltage binning method

**Algorithm**: NEW VOLTAGE BINNING ALGORITHM

**Input**: Timing and power constraints, measured data of timing and power from $N$ manufactured chips.

**Output**: Optimal voltage binning scheme and the corresponding number of bins $k_{opt}$.

```
1. Map measured data to a set of $V_{dd}$ segments $\mathcal{S} = \{S_j\}$, in which
   $S_j = [V_{low,j}, V_{high,j}]$ represents the $V_{dd}$ range at which the $j^{th}$ chip
   satisfies timing and power constraints.
2. Keep only the valid $V_{dd}$ segments $\mathcal{S}_{val}$ ($V_{low} \leq V_{high}$).
3. Calculate voltage levels and corresponding bins for optimal binning
   scheme:
   $\mathcal{S}_{left} = \mathcal{S}_{val}$; $i = 1$
   while $\mathcal{S}_{left}$ is not empty
       $\mathcal{S}_{tmp} = \mathcal{S}_{left}$
       GREEDY-SET-COVER($\mathcal{S}_{tmp}$) $\rightarrow \mathcal{S}_{left}, V_i$
       $U_i$ = chips covered by $V_i$; i ++
   $k_{opt} = i - 1$
```

Figure 7.1: The algorithm sketch of the proposed new voltage binning method.

In this section, we present a new voltage binning scheme, which not only gives the good solution for a given set of voltage levels, but also computes the minimum number of bins required. Fig. 7.1 presents the overall flow of the proposed method and highlights the major computing steps. Basically, Step 1 and 2 compute the valid voltage segment for each chip. Step 3 determinates the voltage levels and the chip assignments to the resulting bins. This is done by a greedy-based set covering method. In Fig. 7.1, $\mathcal{S}_{left}$ denotes the set of uncovered voltage segments left in the complete set of valid voltage segments $\mathcal{S}_{val}$. $V_i$ is the $i^{th}$ supply voltage level, and chips assigned to $U_i$ can meet both the power and timing constraints at supply voltage $V_i$.

The algorithm in Step 3 tries to find the voltage level one at a time such that it can cover as many chips as possible in a greedy fashion (a chip is covered if its valid $V_{dd}$ segment contains the given voltage level). The algorithm stops when all the chips are covered, and the number of levels seen so far ($k_{opt}$) will be the minimum number of bins that can reach the maximum possible yield $Y_{max}$. In the new algorithm, we can also provide a formulation to predict the minimum number of bins required under the uniform binning scheme from the distribution of length of valid $V_{dd}$ segment, which can serve as a guideline for the number of bins required.

### 7.3.1 Voltage binning considering valid segment

For a chip, the working supply voltage range (segment) $[V_{low}, V_{high}]$ actually can be considered as a knob to do the trade-off between the power and timing of the circuit. As we know, supply voltage affects power consumption and timing performance in opposite ways. Reducing supply voltage will decrease the dynamic power and leakage power, which is often considered the most effective technique for low power design. On the other hand, propagation delay will increase as supple voltage decreases [73]. Fig. 7.2 shows the mean delay and power consumption as functions of supply voltage, which show such trends clearly. As a result, given the power consumption bound and the timing constraint for a chip, $V_{low}$ is mainly decided by timing and $V_{high}$ is mainly determined by power constraint. Since process variation leads to different timing performances and power consumptions, the valid $V_{dd}$ segment $[V_{low}, V_{high}]$ will

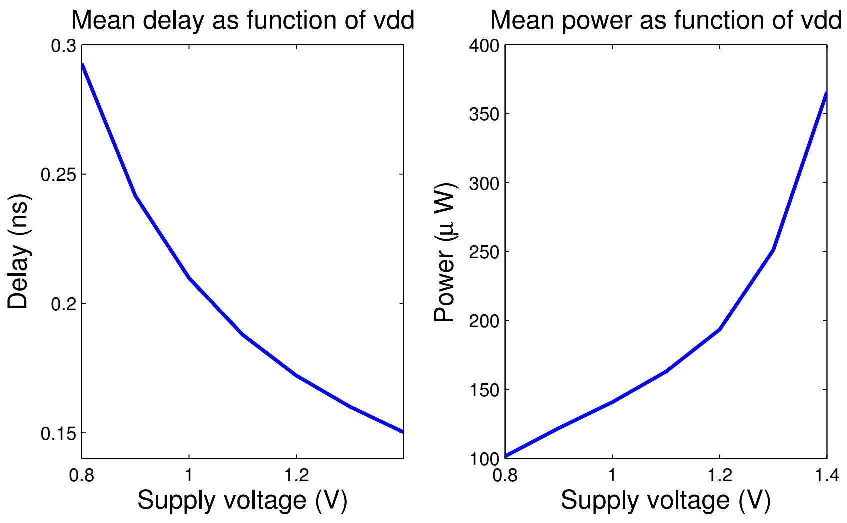

Figure 7.2: The delay and power change with supply voltage for C432.

be different for each chip. As a result, the measured timing and total power data from a chip can be mapped onto corresponding working $V_{dd}$ segments, which is the Step 1 in Fig. 7.1. For some chips, we may have $V_{low} > V_{high}$ (invalid segment), which means that these chips will fail on any supply voltage. So we call them "bad" chips.

Suppose there are $N$ sampling chips from testing, and $n_{bad}$ are bad chips. Obviously, the maximum of possible yield via voltage binning scheme only will be

$$Y_{max} = (N - n_{bad})/N, \tag{7.3}$$

We then define the set of valid segments $\mathcal{S}_{val} = [V_{low}, V_{high}]$ by removing the bad chips from the sampling set and only keeping the valid segments (Step 2 in Fig 7.1). Then the voltage binning scheme problem in (7.2) can be framed into a set-cover problem. Take Fig. 7.3 for instance, there are $n_{val} = 13$ horizontal segments between $V_{min}$ and $V_{max}$ (each corresponds a valid $V_{dd}$ segment), and the problem becomes using minimum number of vertical lines to cover all the horizontal segments. In this case, three voltage levels can cover all the $V_{dd}$ segments of these 13 chips. We also notice that one chip can be covered by more than one voltage level. In this case, it can be assigned to any voltage level containing it. The problem is well known in graph theory with known efficient solutions. This valid voltage segment model has many

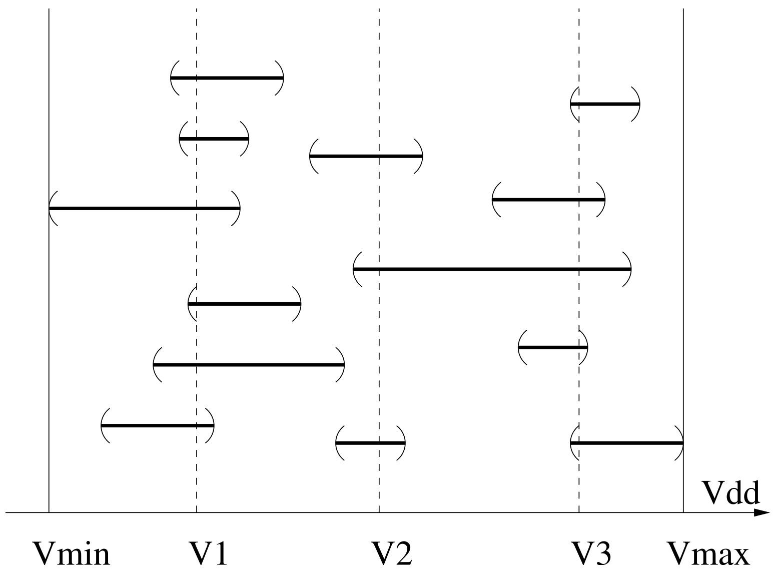

Figure 7.3: Valid voltage segment graph and the voltage binning solution.

benefits compared with other yield analysis model for voltage binning:

1. Distribution of length of valid supply voltage segment can provide information about the minimum number for uniform binning under certain yield requirement (e.g. to achieve 99% for $Y_{max}$, more details in 7.3.2.)
2. The model can also be used when the allowed supply voltage level for one voltage bin is an interval or a group of discrete values for voltage scaling mechanism instead of a scalar (details in Section 7.3.3). To the best knowledge of the authors, this proposed method is the first one working for this case.

## 7.3.2 Bin number prediction under given yield requirement

The distribution of valid $V_{dd}$ segment length (defined as $len = V_{high} - V_{low}$) can be a guide in yield optimization when there is a lower bound requirement for yield. And it works for both uniform binning and optimal binning. Notice that the optimal binning can always has an equal or better yield than the uniform binning. Actually the experiment result part shows that the number of bins needed for optimal voltage binning is much smaller than the prediction from the distribution of *len*. Fig. 7.4

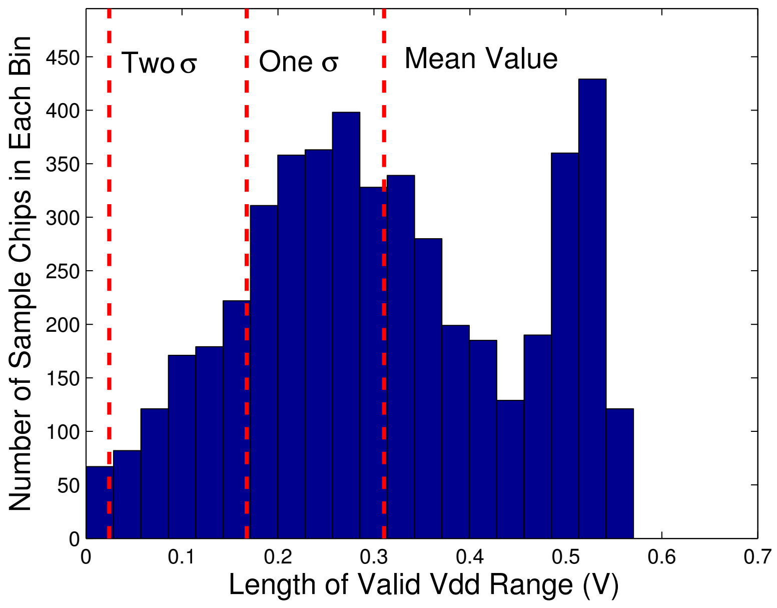

Figure 7.4: Histogram of the length of valid supply voltage segment *len* for C432.

shows the histogram of valid supply voltage length, *len*, for testing circuit C432. From which we can see that it is hard to tell which type of random variable it belongs to. However, it is quite simple to get the numerical probability density function (PDF) and cumulative distribution function (CDF) from measured data of testing samples, as well as the mean value and standard deviation.

Suppose the yield requirement is $Y_{req}$, and the allowed supply voltages for testing is in $[V_{min}, V_{max}]$. For the uniform voltage binning scheme, there is $k$ bins, and the set of supply voltage levels is $\vec{V} = \{V_1, V_2, \ldots, V_k\}$. Since the voltage binning scheme is uniform,

$$V_i - V_{i-1} = \Delta V \quad \text{const.} \quad (i = 2, 3, \ldots k). \tag{7.4}$$

For the uniform voltage binning scheme, we have the following observations:

**Observation 1.** If there are $k$ bins in $[V_{min}, V_{max}]$, then

$$\Delta V = (V_{max} - V_{min})/(k + 1). \tag{7.5}$$

**Observation 2.** For a $V_{dd}$ segment $[V_{low}, V_{high}]$ with a length $len = V_{high} - V_{low}$, if $len > \Delta V$, there must exist at least one $V_{dd}$ level in the set of supply voltage levels $\vec{V} = \{V_1, V_2, \ldots, V_k\}$ that can cover $[V_{low}, V_{high}]$. Now we have the following results:

**Proposition 7.3.1.** *For the yield requirement $Y_{req}$, the upper bound for voltage binning numbers $k_{up}$ can be determined by*

$$k_{up} = \frac{V_{max} - V_{min}}{F^{-1}(1 - Y_{req})} - 1, \tag{7.6}$$

*where $F^{-1}(len)$ is the inverse function of cumulative distribution function (CDF) of $len$.*

(7.6) basically says that the upper bound for the numbers of voltage bins in uniform scheme can be predicted from the yield requirement and the distribution of $len$.

**Proof sketch for Proposition 7.3.1**:

If the chip satisfies the yield requirement $Y_{req}$,

$$1 - F(\Delta V) \leq Y_{req} \quad (Observation 2). \tag{7.7}$$

For the upper bound for voltage binning numbers $k_{up}$, the corresponding $\Delta V_{min}$ can be calculated by

$$\Delta V_{min} = \frac{V_{max} - V_{min}}{k_{up} + 1} \quad (Observation 1). \tag{7.8}$$

From (7.7) and (7.8),

$$Y_{req} = 1 - F(\Delta V_{min}) = 1 - F\left(\frac{V_{max} - V_{min}}{k_{up} + 1}\right). \tag{7.9}$$

which is equivalent form of (7.6). Q.E.D.

Notice that the optimal binning always has a better or equal yield compared to uniform binning using same number of bins. Therefore, if the uniform voltage binning scheme with $k$ bins already satisfies the yield requirement, $k$ bins must be enough for the optimal voltage binning scheme. So the histogram for the length of valid $V_{dd}$ segment can be used to estimate the upper bound for the number of bins needed for a certain yield requirement for both uniform and optimal voltage binning schemes. And this process can be done right after mapping measured power and timing data to working $V_{dd}$ segments.

### 7.3.3 Yield analysis and optimization

The whole voltage binning algorithm for yield analysis and optimization is given in Fig. 7.1. After the yield analysis and optimization, supply voltage levels $\vec{V} = \{V_1, V_2, \ldots, V_{k,opt}\}$ , the corresponding set of bins $\vec{U} = \{U_1, U_2, \ldots, U_{k,opt}\}$ can be calculated up to $k_{opt}$, where $Y_{k,opt} = Y_{max}$ already.

There are many algorithms for solving the set-cover problem in Step 3. By choosing optimal set-cover algorithm, the global optimal solution can be obtained. In this case, the decision version of set-covering problem will be NP-complete. In this paper we use a greedy approximation algorithm as shown in Fig. 7.5, which can easily be implemented to run in polynomial time and achieve a good enough approximation of optimal solution. Notice that the greedy approximation is not necessary and any algorithm for set-cover can be used in Step 3, which is not a limitation for our valid supply voltage segment model. The solution found by GREEDY-SET-COVER is at most a small constant times larger than optimal [13], which is found already satisfactory as shown in the experimental results. Besides, the greedy algorithm can guarantee that each voltage level will cover the most segments corresponding to uncovered testing chips, which means this algorithm is incremental. As a result, if only $k-1$ bins is needed, we can stop the computation at $k-1$ instead of $k$. And when the designer needs more voltage bins, the computation doesn't need to be start all over again. Actually the benefit of incremental voltage binning scheme is very useful for circuit design. Since when the number of bins increase from $k-1$ to $k$, the existing $k-1$ voltage levels will be the same.

```
Algorithm: GREEDY-SET-COVER($\mathcal{S}$).
Input: $\mathcal{S}$.
Output: $\mathcal{C}$.

1. Select an supply voltage value $V_g$ value which covers most voltage
   segments in $\mathcal{S}$
2. $\mathcal{C} \leftarrow \emptyset$
3. for i = 1 : size($\mathcal{S}$)
     if $V_g \in S_i$
       $\mathcal{C} \leftarrow \mathcal{C} + S_i$
4. return $\mathcal{C}$
```

Figure 7.5: The flow of greedy algorithm for covering most uncovered elements in $\mathcal{S}$.

We remark that the proposed method can be easily extended to deal with a group of discrete values $V_{g,1}, V_{g,2}, \ldots$ for dynamic voltage scaling under different operation modes instead of a single voltage. For example, if the $i^{th}$ supply voltage level $V_i$

contains two discrete values, $V_s$ and $V_h$, which are the supply voltages for saving-power mode and high-performance mode, respectively (anything in between also works for the selected chips). Set-cover algorithm in Fig. 7.5 now will use a range $V_g$ (defined by users) to cover the voltage segments instead of a single voltage level. Such extension is very straightforward for the proposed method.

## 7.4 Experimental results

In this section, the proposed voltage binning technique for yield analysis and optimization was verified on circuits in the ISCAS85 benchmark set with constraints on timing performance and power consumption. The circuits were synthesized with Nangate Open Cell Library. The technology parameters come from the 45nm FreePDK Base Kit and PTM models [59]. The proposed method has been implemented in Matlab 7.8.0. All the experimental results are carried out in a Linux system with quad Intel Xeon CPUs with 2.99Ghz and 16GB memory.

### 7.4.1 Setting of process variation

For each type of circuit in the benchmark, 10000 Monte Carlo samples are generated from process variations. In this paper, effective gate length $L$ and gate oxide thickness $T_{ox}$ are considered as two main sources of process variations. According to [1], the physical variation in $L$ and $T_{ox}$ should be controlled within +/-12%. So the $3\sigma$ values of variations for $L$ and $T_{ox}$ were set to 12% of the nominal values, of which inter-die variations constitute 20% and intra-die variations, 80%. $L$ is modeled as sum of spatial correlated sources of variations, and $T_{ox}$ is modeled as an independent source of variation. The same framework can be easily extended to include other parameters of variations. Both $L$ and $T_{ox}$ are modeled as Gaussian parameters. For the correlated

$L$, the spatial correlation was modeled based on the exponential models [77].

The power and timing information as a function of supply voltage for each testing chip is characterized by using SPICE simulation. Under $45nm$ technology, typical supply voltage range is $0.85V - 1.3625V$ [25]. Since that, $V_{dd}$ is varied between 0.8 volt and 1.4 volt in this paper, which is enough for $45nm$ technology.

We remark that practically the power and timing information can be obtained from measurements. As a result, all the sources of variability of transistors and interconnects including inter-die and intra-die variations with spatial correlations will be considered automatically.

### 7.4.2 Prediction of bin numbers under yield requirement

As mentioned in 7.3.2, the proposed valid segment model can be used to predict the number of bins needed under yield requirement before voltage binning optimization. Table 7.1 shows the comparison between the predicted number and the actual number

Table 7.1: Predicted and actual number of bins needed under yield requirement.

| *Circuit* | $Y_{req}$ | Predicted | Real for Uni. | Real for Opt. |
|---|---|---|---|---|
| C432 | 99% | 25 | 23 | 4 |
| | 97% | 10 | 9 | 3 |
| | 95% | 7 | 6 | 3 |
| C1908 | 99% | 27 | 12 | 7 |
| | 97% | 11 | 6 | 3 |
| | 95% | 7 | 3 | 3 |
| C2670 | 99% | 8 | 4 | 3 |
| | 97% | 5 | 3 | 2 |
| | 95% | 3 | 2 | 1 |
| C7552 | 99% | 30 | 12 | 5 |
| | 97% | 9 | 4 | 3 |
| | 95% | 6 | 3 | 2 |

needed under yield requirement for the testing chips. In this table $Y_{req}$ means the lower bound requirement for yield optimization (normalized by $Y_{max}$). Column 3 is the predicted number of bins; and columns 4 and 5 are the actual bin numbers found for the uniform and optimal voltage binning schemes, respectively. This table validates the upper bound formulation for the needed number of bins in 7.3.2. From this table, we can see that the predicted value is always the upper bound of actual number of bins needed, which can be applied as a guide for yield requirement in optimization. Table 7.1 also shows that the optimal voltage binning scheme can significantly reduce the number of bins compared with the uniform voltage binning schema under the same yield requirement. When yield requirement is 99% of the optimal yield, the optimal voltage binning scheme can reduce 52% bin count on average.

### 7.4.3 Comparison between uniform and optimal voltage binning schemes

Experiments for both uniform and the optimal voltage binning schemes with different number of bins are used to verify the proposed voltage binning technique. Table 7.2 shows the results, where $Y_{max}$ is the maximum chip yield which can be achieved when $V_{dd}$ is adjusted individually for each manufactured chip, *VB* stands for voltage binning schemes used and $k_{opt}$ is the minimum number of bins to achieve $Y_{max}$. From Table 7.2, we can see that the yield of optimal VB always increases with the number of bins, with $Y_{max}$ as the upper bound. And the voltage binning can significantly improve yield compared with simple supply voltage. Column 8 in Table 7.2 shows that the number of bins needed to achieve $Y_{max}$ in optimal voltage binning schemes is only 1.88% of number of bins needed in the uniform scheme on average, which means that optimal voltage binning schemes is much more economic in order to reach the best possible yield.

Table 7.2: Yield under uniform and optimal voltage binning schemes (%).

| *Circuit* | $Y_{max}$ | VB | 1 bin | 2bins | 5bins | 10bins | $k_{opt}$ |
|---|---|---|---|---|---|---|---|
| C432 | 96.66 | Uni. | 60.19 | 79.04 | 90.52 | 94.36 | 4514 |
| | | Opt. | 80.08 | 88.68 | 96.42 | 96.66 | 10 |
| C1908 | 98.06 | Uni. | 71.80 | 91.46 | 95.20 | 97.04 | 437 |
| | | Opt. | 89.18 | 92.88 | 97.18 | 98.06 | 21 |
| C2670 | 90.15 | Uni. | 81.12 | 87.13 | 89.74 | 89.95 | 1205 |
| | | Opt. | 85.77 | 88.34 | 89.83 | 90.08 | 13 |
| C7552 | 93.46 | Uni. | 73.94 | 86.38 | 91.40 | 92.34 | 1254 |
| | | Opt. | 87.22 | 90.30 | 92.64 | 93.26 | 18 |

Fig 7.6 compares the yields from uniform and optimal voltage binning schemes with the number of bins from 1 to 10 for C432. This figure shows that the optimal binning scheme always provides higher yield than the uniform binning scheme. For optimal voltage binning scheme, the yield increasing speed is slower down as the bin number increases since we use greedy algorithm. For other testing circuits, similar phenomenon is observed from the yield results.

### 7.4.4 Sensitivity to frequency and power constraints

For very strict power or frequency constraints, voltage binning can provide more opportunities to improve yield. Figure 7.7 shows the changes in parametric yield for C432 with and without voltage binning yield optimization due to the changes in frequency and power consumption requirements, where $P_{norm}$ is normalized power constraint and $f_{norm}$ is normalized frequency constraint. By analyzing this figure, we can see that parametric yield is sensitive to both performance and power constraints. As a result, yield can be substantially increased by binning supply voltage to a very small amount of levels in the optimal voltage binning scheme. For example, without voltage binning technique, the yield will fall down 0% when constraints become 20% stricter, while the voltage binning technique can keep the yield as high as 80% under

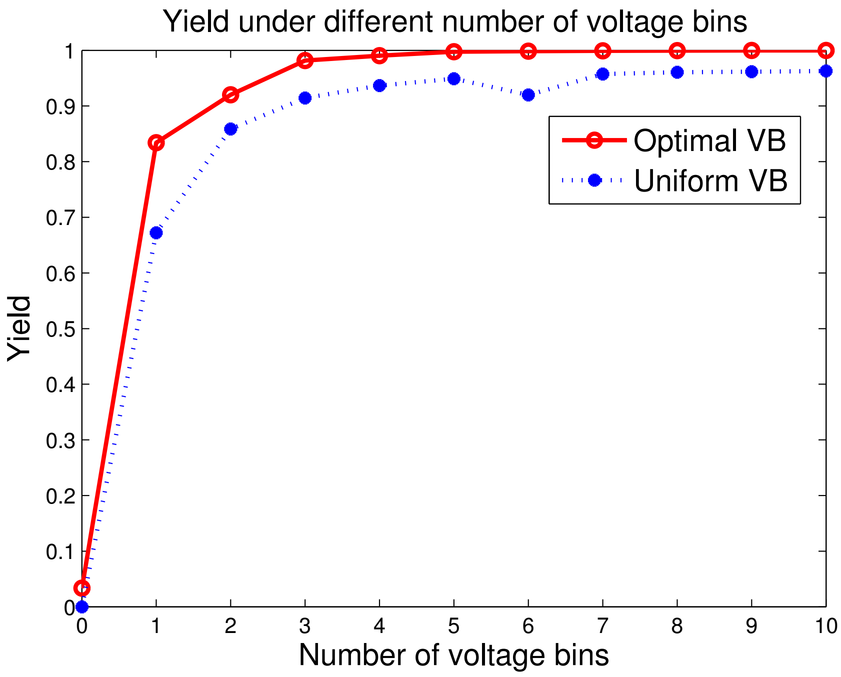

Figure 7.6: Yield under uniform and optimal voltage binning schemes for C432.

the same situation.

### 7.4.5 CPU times

Table 7.3 compares the CPU times among different voltage binning schemes and different number of bins. Since the inputs of our algorithm in Fig. 7.1 are the measured data for real chips practically, the time cost of measuring data is not counted in the time cost of the voltage binning method. But in this paper, the timing and power data is generated from SPICE simulation. There are three steps in our proposed method as shown in Fig. 7.1. It is easy to see that the time complexity of Step 1 and 2 are both $O(N)$, where $N$ is the number of MC sample points. From [13], Step 3 can run within $O(N^2 ln(N))$ time. Therefore, the speed of our voltage binning algorithm is not related to the size of circuits. Table 7.3 confirms that binning technique is insignificant even for the case of 10 bins, and the time cost is not increasing with the

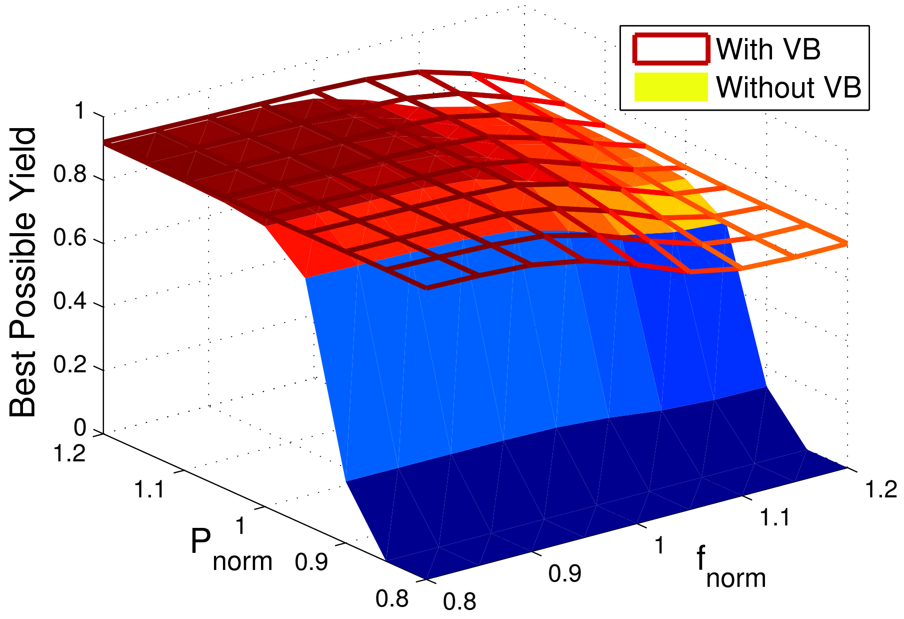

Figure 7.7: Maximum achievable yield as function of power and performance constraints for C2670.

number of gates on chip.

Table 7.3: CPU time comparison(s).

| *Circuit* | VB | 1 bin | 2bins | 5bins | 10bins |
|---|---|---|---|---|---|
| C432 | Uni. | 0.0486 | 0.0571 | 0.0866 | 0.1374 |
| | Opt. | 0.0747 | 0.0786 | 0.0823 | 0.0827 |
| C1908 | Uni. | 0.0551 | 0.0749 | 0.1237 | 0.2037 |
| | Opt. | 0.0804 | 0.0840 | 0.0874 | 0.0901 |
| C2670 | Uni. | 0.0347 | 0.0371 | 0.0425 | 0.0504 |
| | Opt. | 0.0686 | 0.0696 | 0.0711 | 0.0704 |
| C7552 | Uni. | 0.0476 | 0.0565 | 0.0925 | 0.1493 |
| | Opt. | 0.0775 | 0.0791 | 0.0802 | 0.0812 |

## 7.5 Summary

In this chapter, we have proposed a new voltage binning technique to improve the yield of chips. First, we have proposed formulation to predict the maximum number of bins required under the uniform binning scheme from the distribution of valid $V_{dd}$ segment length. We then developed an approximation of optimal binning scheme based on greedy-based set-cover solution to minimize the number of bins and keep the corresponding voltage levels incremental. The new method is also extendable to deal with a range of working supply voltages for dynamic voltage scaling operation. Experimental results on some benchmarks on 45nm technology show that the proposed method can correctly predict the upper bound on the number of bins required. The proposed optimal binning scheme can lead to significant saving for the number of bins compared to the uniform one to achieve the same yield with very small CPU cost.

# Chapter 8

# Conclusion and Future Works

This chapter concludes the dissertation. The first section summaries the research contributions for statistical analysis techniques we proposed for nano-scale VLSI design in the dissertation. Then the second section discusses how our work can be extended in the future.

## 8.1 Summary of research contributions

As technology sizes shrink down to nanometer regime, circuit integration increases, and the variational consideration of process has to be assessed in various VLSI design steps to ensure robust circuit design. In order to efficiently generate high yield chips, we must have a reliable statistical model in the first place.

In this dissertation, we have presented several novel statistical modeling methodologies for VLSI design automation. Then based on our models, we have proposed fast and accurate approach for statistical analysis for full-chip leakage power, and 3D capacitance extraction. An voltage binning technique was also proposed for yield improvement and optimization.

### 8.1.1 Fast and accurate full-chip statistical analysis of leakage power

Chapter 4 presented a method for analyzing the full-chip leakage current distributions. Compared to existing approaches, no grid-based partitioning and approximation were required. Instead, the spatial correlations were naturally handled by orthogonal decompositions. The proposed method was very efficient and it becomes linear in the presence of strong spatial correlations. Experimental results showed that the proposed method is about $10\times$ faster than the recently proposed method [9] with constant better accuracy.

In Chapter 5 an improved linear-time algorithm for full-chip statistical analysis of leakage powers was proposed, which worked well in the presence of general spatial correlation (strong or weak). In this algorithm, a new statistical leakage characterization in SCL was put forward for fast full-chip statistical leakage estimation. The most promising feature of this technique was $O(N)$ time complexity, where $N$ was the number of grids on chip. The numerical examples showed that the proposed algorithm was 1000X faster than a recently proposed grid-based method [9] with similar accuracy and many orders of magnitude times speedup over the Monte Carlo method. Further more, an incremental version was proposed, which provided about 10X further speedup. So we ended up with 10,000X compared to [9], and the incremental analysis could achieve more speedup over the full leakage analysis for larger problem sizes.

### 8.1.2 Efficient non-linear 3-D statistical capacitance extraction method

Next, in Chapter 6, we have proposed a efficient statistical capacitance extraction method for interconnect conductors considering process variations. In the new method called *StatCap*, orthogonal polynomials were used to represent the statistical processes. The chapter showed how the variational potential coefficient matrix was represented in a first-order form using Taylor expansion and orthogonal decomposition. Then an augmented potential coefficient matrix, which consisted of the coefficients of the polynomials, was derived. After that, corresponding augmented system was solved to obtain the variational capacitance values in the orthogonal polynomial form. Then this chapter proposed a method to extend *StatCap* to the second-order form to give more accurate results without loss of efficiency compared to the linear models. We showed the derivation of the analytic second-order orthogonal polynomials for the variational capacitance integral equations.

To the best knowledge of the author, this was the first time that closed-form formulas for second-order potential coefficient matrix was proposed, which provided more accurate extraction result with tiny cost of additional runtime. Also, techniques such as setting up the augmented equation once and exploiting the sparsity and low-rank property were used to speedup the extraction process. Experimental results showed that *statCap* was two orders of magnitude faster than the recently proposed statistical capacitance extraction method based on the spectral stochastic collocation approach [86] and many orders of magnitude faster than the Monte Carlo method for several practical conductor structures.

### 8.1.3 Yield optimization for nano-scale VLSI considering statistical behavior

Chapter 7 proposed a yield optimization technique using voltage binning method to improve yield of chips. Voltage binning technique tried to assign different supply voltages to different chips in order to improve the yield. The chapter introduced a novel "valid voltage segment concept", which was determined by the timing and power constraints of chips. Different from previous voltage binning technique, the new concept of valid voltage segment enabled a series of efficient technique for yield analysis and improvement.

Using this novel concept, we have proposed formulation to predict the maximum number of bins required under the uniform binning scheme from the distribution of valid $V_{dd}$ segment length, and then developed an approximation of optimal binning scheme based on greedy-based set-cover solution. The nature of greedy algorithm could keep the corresponding voltage levels incremental while we were trying to minimize the number of bins. The new method was also extendable to deal with a range of working supply voltages for dynamic voltage scaling operation. Experimental results on some benchmarks on 45nm technology showed that the proposed voltage binning technique could correctly predict the upper bound on the number of bins required, and the proposed optimal binning scheme can lead to significant saving for the number of bins compared to the uniform one to achieve the same yield with ignorable runtime.

## 8.2 Future research topics

Our work in statistical related VLSI research can be extended in several following directions.

First, we can extend our fast full-chip statistical leakage analysis methodology to run-time leakage reduction or simulation. Although the model of leakage power used in this is idle-time leakage, the proposed method can be extended to leakage computation under the run-time scenario with leakage reduction. In this dissertation, we discussed about it, and further experiment needs to be done.

Second, for 3D capacitance extraction, we have put less emphasis on the acceleration techniques during the extraction processes such as the multiple-pole scheme [47] and the hierarchical methods [66, 81], where the key idea is using the more sophisticated iterative solvers such as general minimal residue (GMRES) [61]. The reason is that this is not the focus area where our major contributions are made. We believe those existing acceleration techniques can significantly speedup the proposed method as they did for the deterministic problem. This is especially the case for the hierarchical approach [66]. The number of panels (thus the random variables) can be considerably reduced and the interaction between panels are constant. These are the areas for our future investigations.

Third, for yield analysis and optimization, there are a lot of new problems lies in process variation related issues. In this dissertation, we focus on the voltage binning scheme. However, there are several different methods to predict and improve yield, and extend to more variation. Increasing chip yield rate considering statistical issues is and will be a continually emerging topic in the future.

# Bibliography

[1] International technology roadmap for semiconductors (ITRS) 2008 edition, 2008. http://public.itrs.net.

[2] International technology roadmap for semiconductors(ITRS), 2009 update. http://public.itrs.net.

[3] A. Abdollahi, F. Fallah, and M. Pedram. Runtime mechanisms for leakage current reduction in CMOS VLSI circuits. In *Proc. Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 213–218, Aug. 2002.

[4] A.A. Abu-Dayya and N.C. Beaulieu. Comparison of methods of computing correlated lognormal sumdistributions and outages for digital wireless applications. In *Proc. IEEE Vehicular Technology Conference*, volume 1, pages 175–179, June 1994.

[5] K. Agarwal, D. Blaauw, and V. Zolotov. Statistical timing analysis for intra-die process variations with spatial correlations. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 900–907, Nov. 2003.

[6] S. Bhardwaj, S. Vrudhula, and A. Goel. A unified approach for full chip statistical timing and leakage analysis of nanoscale circuits considering intradie process variations. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27(10):1812–1825, Oct. 2008.

[7] S. Borkar, T. Karnik, and V. De. Design and reliability challenges in nanometer technologies. In *Proc. Design Automation Conf. (DAC)*, pages 75–75, 2004.

[8] H. Chang and S. Sapatnekar. Statistical timing analysis under spatial correlations. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(9):1467–1482, Sept. 2005.

[9] H. Chang and S. S. Sapatnekar. Full-chip analysis of leakage power under process variations, including spatial correlations. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 523–528, 2005.

[10] H. Chen, S. Neely, J. Xiong, V. Zolotov, and C. Visweswariah. Statistical modeling and analysis of static leakage and dynamic switching power. In *Power and Timing Modeling, Optimization and Simulation: 18th International Workshop, (PATMOS)*, pages 178–187, Sep. 2008.

[11] R. Chen, L. Zhang, V. Zolotov, C. Visweswariah, and J. Xiong. Static timing: back to our roots. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 310–315, Jan. 2008.

[12] C. Chiang and J. Kawa. *Design for Manufacturability.* Springer, 2007.

[13] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, second edition, 2001.

[14] S. G. Duvall. Statistical circuit modeling and optimization. In *Intl. Workshop Statistical Metrology*, pages 56–63, Jun 2000.

[15] J. Fan, N. Mi, S. X.-D. Tan, Y. Cai, and X. Hong. Statistical model order reduction for interconnect circuits considering spatial correlations. In *Proc. Design, Automation and Test In Europe. (DATE)*, pages 1508–1513, 2007.

[16] R. Fernandes and R. Vemuri. Accurate estimation of vector dependent leakage power in presence of process variations. In *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, pages 451–458, Oct. 2009.

[17] George F. Fishman. *Monte Carlo, concepts, algorithms, and Applications.* Springer, 1996.

[18] P. Friedberg, Y. Cao, J. Cain, R. Wang, J. Rabaey, and C. Spanos. Modeling within-die spatial correlation effects for process design co-optimization. In *Proceedings of the 6th International Symposium on Quality of Electronic Design*, pages 516–521, 2005.

[19] R. Ghanem. The nonlinear Gaussian spectrum of log-normal stochastic processes and variables. *Journal of Applied Mechanics*, 66:964–973, December 1999.

[20] R. G. Ghanem and P. D. Spanos. *Stochastic Finite Elements: A Spectral Approach.* Dover Publications, 2003.

[21] G. H. Golub and C. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, 3rd edition, 1996.

[22] K. R. Heloue, N. Azizi, and F. N. Najm. Modeling and estimation of full-chip leakage current considering within-die correlation. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 93–98, 2007.

[23] T. Hill and P. Lewicki. *STATISTICS: Methods and Applications.* StatSoft, 2007.

[24] A. Hyvarinen, J. Karhunen, and E. Oja. *Independent Component Analysis*. Wiley, 2001.

[25] Intel pentium processor e5200 series specifications. http://ark.intel.com/Product.aspx?id=37212.

[26] A. Iserles. *A First Course in the Numerical Analysis of Differential Equations*. Cambridge University, 3 edition, 1996.

[27] H. Jiang, M. Marek-Sadowska, and S. R. Nassif. Benefits and costs of power-gating technique. In *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, pages 559–566, Oct. 2005.

[28] R. Jiang, W. Fu, J. M. Wang, V. Lin, and C. C.-P. Chen. Efficient statistical capacitance variability modeling with orthogonal principle factor analysis. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 683–690, 2005.

[29] I. T. Jolliffe. *Principal Component Analysis*. Springer-Verlag, 1986.

[30] T. Karnik, S. Borkar, and V. De. Sub-90nm technologies-challenges and opportunities for CAD. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 203–206, San Jose, CA, Nov 2002.

[31] Andreas Klimke. Sparse Grid Interpolation Toolbox – user's guide. Technical Report IANS report 2006/001, University of Stuttgart, 2006.

[32] Andreas Klimke and Barbara Wohlmuth. Algorithm 847: spinterp: Piecewise multilinear hierarchical sparse grid interpolation in MATLAB. *ACM Transactions on Mathematical Software*, 31(4), 2005.

[33] M. W. Kuemerle, S. K. Lichtensteiger, D. W. Douglas, and I. L. Wemple. Integrated circuit design closure method for selective voltage binning. In *U.S. Patent 7475366*, Jan. 2009.

[34] A. Labun. Rapid method to account for process variation in full-chip capacitance extraction. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 23:941–951, June 2004.

[35] Peng Li and Weiping Shi. Model order reduction of linear networks with massive ports via frequency-dependent port packing. In *Proc. Design Automation Conf. (DAC)*, pages 267–272, 2006.

[36] T. Li, W. Zhang, and Z. Yu. Full-chip leakage analysis in nano-scale technologies: Mechanisms, variation sources, and verification. In *Proc. Design Automation Conf. (DAC)*, pages 594–599, June 2008.

[37] X. Li, J. Le, P. Gopalakrishnan, and L. Pileggi. Asymptotic probability extraction for non-normal distributions of circuit performance. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 2–9, 2004.

[38] X. Li, J. Le, L. Pileggi, and A. Strojwas. Projection-based performance modeling for inter/intra-die variations. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 721–727, 2005.

[39] Y. Lin and D. Sylvester. Runtimie lekaage power estimation technique for combinational circuits. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 660–665, Jan. 2007.

[40] Y. Liu, S. Nassif, L. Pileggi, and A. Strojwas. Impact of interconnect variations on the clock skew of a gigahertz microprocessor. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 168–171, 2000.

[41] MCNC benchmark circuit placements. http://vlsicad.ucsd.edu/GSRC/bookshelf/Slots/Placement/.

[42] N. Mi, J. Fan, S. X.-D. Tan, Y. Cai, and X. Hong. Statistical analysis of on-chip power delivery networks considering lognormal leakage current variations with spatial correlations. *IEEE Trans. on Circuits and Systems I: Fundamental Theory and Applications*, 55(7):2064–2075, Aug. 2008.

[43] N. Mi, S. X.-D. Tan, Y. Cai, and X. Hong. Fast variational analysis of on-chip power grids by stochastic extended krylov subspace method. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 27(11):1996–2006, 2008.

[44] N. Mi, S. X.-D. Tan, P. Liu, J. Cui, Y. Cai, and X. Hong. Stochastic extended Krylov subspace method for variational analysis of on-chip power grid networks. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 48–53, 2007.

[45] S. Mukhopadhyay, S. Member, A. Raychowdhury, and K. Roy. Accurate estimation of total leakage in nanometer-scale bulk cmos circuits based on device geometry and doping profile. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(3):363–381, March 2005.

[46] Saibal Mukhopadhyay and Kaushik Roy. Modeling and estimation of total leakage current in nano-scaled CMOS devices considering the effect of parameter variation. In *Proc. Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 172–175, 2003.

[47] K. Nabors and J. White. Fastcap: A multipole accelerated 3-d capacitance extraction program. *IEEE TCAD*, pages 1447–1459, November 1991.

[48] K. Narbos and J. White. FastCap: a multipole accelerated 3D capacitance extraction program. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 10(11):1447–1459, 1991.

[49] Siva Narendra, Vivek De, Shekhar Borkar, Dimitri A. Antoniadis, and Anantha P. Chandrakasan. Full-chip subthreshold leakage power prediction and reduction techniques for sub-0.18-$\mu$m CMOS. *IEEE J. Solid-State Circuits*, 39(2), 2004.

[50] S. Nassif. Delay variability: sources, impact and trends. In *Proc. IEEE Int. Solid-State Circuits Conf.*, pages 368–369, San Francisco, CA, Feb 2000.

[51] S. Nassif. Design for variability in DSM technologies. In *Proc. Int. Symposium. on Quality Electronic Design (ISQED)*, pages 451–454, San Jose, CA, Mar 2000.

[52] S. Nassif. Model to hardware correlation for nm-scale technologies. In *Proc. IEEE International Workshop on Behavioral Modeling and Simulation (BMAS)*, Sept 2007. Keynote speech.

[53] Nangate open cell library. `http://www.nangate.com/`.

[54] E. Novak and K. Ritter. Simple cubature formulas with high polynomial exactness. *Constructive Approximation*, 15(4):449–522, Dec 1999.

[55] M. Orshansky, L. Milor, and C. Hu. Characterization of spatial intrafield gate cd variability, its impact on circuit performance, and spatial mask-level correction. In *IEEE Trans. on Semiconductor Devices*, volume 17, pages 2–11, Feb 2004.

[56] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. on Numerical Analysis*, 12(4):617–629, September 1975.

[57] A. Papoulis and S. Pillai. *Probability, Random Variables and Stochastic Processes*. McGraw-Hill, 2001.

[58] M. Pedram and S. Nazarian. Thermal modeling, analysis and management in VLSI circuits: principles and methods. *Proc. of IEEE, Special Issue on Thermal Analysis of ULSI*, 94(8):1487–1501, 2006.

[59] Predictive Technology Model. `http://www.eas.asu.edu/~ptm/`.

[60] R. Rao, A. Srivastava, D. Blaauw, and D. Sylvester. Statistical analysis of subthreshold leakage current for VLSI circuits. *IEEE Trans. on Very Large Scale Integration (VLSI) Systems*, 12(2):131–139, Feb 2004.

[61] Y. Saad and M. H. Schultz. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. on Sci and Sta. Comp.*, pages 856–869, 1986.

[62] B. E. A. Saleh and M. C. Teich. *Fundamentals of Photonics.* Wiley, 1991.

[63] S. Shah, P. Gupta, and A. Kahng. Standard cell library optimization for leakage reduction. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 983–986, July 2006.

[64] R. Shen, N. Mi, S. X-.D. Tan, Y. Cai, and X. Hong. Statistical modeling and analysis of chip-level leakage power by spectral stochastic method. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 161–166, Jan. 2009.

[65] R. Shen, S. X-.D. Tan, and J. Xiong. A linear algorithm for full-chip statistical leakage power analysis considering weak spatial correlation. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 481–486, Jun. 2010.

[66] W. Shi, J. Liu, N. Kakani, and T. Yu. A fast hierarchical algorithm for 3-d capacitance extraction. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 21(3):330–336, March 2002.

[67] R. Singhal, A. Balijepalli, A. Subramaniam, F. Liu, S. Nassif, and Y. Cao. Modeling and analysis of non-rectangular gate for post-lithography circuit simulation. In *Proc. Design Automation Conf. (DAC)*, pages 823–828, June 2007.

[68] A. Srivastava, R. Bai, D. Blaauw, and D. Sylvester. Modeling and analysis of leakage power considering within-die process variations. In *Proc. Int. Symp. on Low Power Electronics and Design (ISLPED)*, pages 64–67, Aug. 2002.

[69] Ashish Srivastava, Dennis Sylvester, and David Blaauw. *Statistical Analaysis and Optimization for VLSI: Timing and Power.* Springer, 2005.

[70] Sheldon X.-D. Tan and L. He. *Advanced Model Order Reduction Techniques in VLSI Design.* Cambridge University Press, 2007.

[71] Radu Teodorescu, Brian Greskamp, Jun Nakano, Smruti R. Sarangi, Abhishek Tiwari, and Josep Torrellas. A model of parameter variation and resulting timing errors for microarchitects. In *Workshop on Architectural Support for Gigascale Integration (ASGI)*, Jun 2007.

[72] J. W. Tschanz, S. Narendra, R. Nair, and V. De. Ectiveness of adaptive supply voltage and body bias for reducing impact of parameter variations in low power and high performance microprocessors. *IEEE J. Solid-State Circuits*, 38(5):826–829, May 2003.

[73] M. Vratonjic, B. R. Zeydel, and V. G. Oklobdzija. Circuit sizing and supply-voltage selection for low-power digital circuit design. In *Power and Timing Modeling, Optimization and Simulation: 18th International Workshop, (PATMOS)*, pages 148–156, 2006.

[74] S. Vrudhula, J. M. Wang, and P. Ghanta. Hermite polynomial based interconnect analysis in the presence of process variations. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 25(10), 2006.

[75] J. M. Wang, B. Srinivas, D. Ma, C. C.-P. Chen, and J. Li. System-level power and thermal modeling and analysis by orthogonal polynomial based response surface approach (OPRS). In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 727–734, Nov. 2005.

[76] D.R. Wilton, S.M. Rao, A.W. Glisson, D.H. Schaubert, O.M. Al-Bundak, and C.M. Butler. Potential integrals for uniform and linear source distributions on polygonal and polyhedral domains. *IEEE Trans. on Antennas and Propagation*, AP-32(3):276–281, March 1984.

[77] J. Xiong, V. Zolotov, and L. He. Robust extraction of spatial correlation. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 26(4), 2007.

[78] D. Xiu and G. Karniadakis. The Wiener-Askey polynomial chaos for stochastic differential equations. *SIAM J. Scientific Computing*, 24(2):619–644, Oct 2002.

[79] D. Xiu and G. Karniadakis. Modeling uncertainty in flow simulations via generalized polynomial chaos. *J. of Computational Physics*, 187(1):137–167, May 2003.

[80] H. Xu, R. Vemuri, and W. Jone. Run-time active leakage reduction by power gating and reverse body biasing: An energy view. In *Proc. IEEE Int. Conf. on Computer Design (ICCD)*, pages 618–625, Oct. 2008.

[81] Shu Yan, Vivek Sarim, and Weiping Shi. Sparse transformation and preconditioners for 3-d capacitance extraction. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 24(9):1420–1426, 2005.

[82] Z. Ye and Z. Yu. An efficient algorithm for modeling spatially-correlated process variation in statistical full-chip leakage analysis. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 295–301, Nov. 2009.

[83] W. Yu, C. Hu, and W. Zhang. Variational capacitance extraction of on-chip interconnects based on continuous surface model. In *Proc. IEEE/ACM Design Automation Conference (DAC)*, pages 758–763, July 2009.

[84] W. Zhang, W. Yu, Z. Wang, Z. Yu, R. Jiang, and J. Xiong. An efficient method for chip-level statistical capacitance extraction considering process variations with spatial correlation. In *Proc. Design, Automation and Test In Europe. (DATE)*, pages 580–585, Mar. 2008.

[85] Ying Zhou, Zhuo Li, Yuxin Tian, Weiping Shi, and Frank Liu. A new methodology for interconnect parasitics extraction considering photo-lithography effects. In *Proc. Asia South Pacific Design Automation Conf. (ASPDAC)*, pages 450–455, Jan. 2007.

[86] H. Zhu, X. Zeng, W. Cai, J. Xue, and D. Zhou. A sparse grid based spectral stochastic collocation method for variations-aware capacitance extraction of interconnects under nanometer process technology. In *Proc. Design, Automation and Test In Europe. (DATE)*, pages 1514–1519, Mar 2007.

[87] Z. Zhu and J. White. FastSies: a fast stochastic integral equation solver for modeling the rough surface effect. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 675–682, 2005.

[88] Zhenhai Zhu, J. White, and A. Demir. A stochastic integral equation method for modeling the rough surface effect on interconnect capacitance. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 887–891, 2004.

[89] V. Zolotov, C. Viweswariah, and J. Xiong. Voltage binning under process variation. In *Proc. Int. Conf. on Computer Aided Design (ICCAD)*, pages 425–432, Nov. 2009.