Statistical Verification and Optimization of Integrated Circuits

Yu Ben

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2011-31
http://www.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-31.html

April 22, 2011

Copyright © 2011, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Statistical Verification and Optimization of Integrated Circuits

By

Yu Ben

A dissertation submitted in partial satisfaction of the
requirements for the degree of
Doctor of Philosophy
in
Engineering – Electrical Engineering and Computer Sciences
in the
Graduate Division
of the
University of California, Berkeley

Committee in charge:

Professor Costas J. Spanos, Chair
Professor Kameshwar Poolla
Professor David Brillinger

Spring 2011

Statistical Verification and Optimization of Integrated Circuits

Copyright © 2011

by

Yu Ben

Abstract

Statistical Verification and Optimization of Integrated Circuits

by

Yu Ben

Doctor of Philosophy in Engineering – Electrical Engineering and Computer Sciences

University of California, Berkeley

Professor Costas J. Spanos, Chair

Semiconductor technology has gone through several decades of aggressive scaling. The ever shrinking size of both transistors and interconnects is necessitating the interaction between the circuit designers and the foundries that fabricate the IC products. In particular, the designers must take into account the impact of the process variability early in the design stage. This includes both the verification and optimization of the circuit with statistical models characterizing the process variability. This thesis advances three frontiers in the variability-aware design flow.

Yield estimation is a crucial but expensive verification step in circuit design. Existing methods either suffer from computationally intensive rare event probability calculation, or exhibit poor stability. We investigate the problem by combining partial least squares (PLS) regression, a dimension-reduction technique, with importance sampling, a variance-reduction technique. The simulation results show that the method is able to improve the convergence speed by at least an order of magnitude over existing fast simulation methods, and four orders of magnitude faster than Monte Carlo. In addition to PLS-preconditioned importance sampling, several other methods are also investigated, and their properties are compared.

For a quicker verification of the robustness of the circuit, circuit designers often simulate the circuit using corner models. Current corner models are extracted using single transistor performances and cannot incorporate local variability. These facts limit the usage of corner models in deep sub-micron devices and analog applications. We propose to extract the customized corners using PLS regression. The method is tested using ISCAS'85 benchmark circuits. Both the probability density function and cumulative distribution function of the circuit performance predicted by the customized corners agree with Monte Carlo simulation, while delivering two orders of magnitude computational acceleration.

The last frontier concerns robust circuit optimization. Most of the existing methods can only deal with a pessimistic worst-case problem. We choose to solve the ideal yield-constrained circuit optimization problem. To this end, we introduce the idea of robust convex approximation to design automation for the first time. Based on the convex approximation, we propose a sequential method to accommodate a realistic, hierarchical variability model. A simple line-search as an outer-loop algorithm is used to control the

output the method. The optimization result shows that the proposed method is capable of handling circuits of thousands of gates without performance penalties due to overdesign.

Table of Contents

Chapter 1 Introduction ... 1
1.1 Technology scaling and its impact ... 1
1.2 Robust circuit design flow and the outline of the thesis ... 2
1.3 Main contributions of this work ... 3
1.4 References ... 4
Chapter 2 Overview of Statistical Circuit Verification ... 7
2.1 Statistical parametric yield estimation ... 7
2.2 Corner models ... 11
2.3 References ... 12
Chapter 3 Fast Circuit Yield Estimation ... 15
3.1 Problem definition ... 15
3.2 Extrapolation methods ... 17
3.2.1 Extrapolation by biasing operating condition ... 17
3.2.2 Extrapolation using extreme value theory ... 19
3.3 Binary tree-based method ... 26
3.3.1 Method ... 27
3.3.2 Results and Discussion ... 29
3.4 Partial least squares-preconditioned importance sampling method ... 31
3.4.1 Importance Sampling ... 32
3.4.2 Partial Least Squares (PLS) regression ... 34
3.4.3 PLS-preconditioned importance sampling ... 35
3.4.4 Results and Discussion ... 36
3.5 Summary ... 42
3.A The calculation of covariance matrix elements in Eq. (3.22) ... 43
3.6 References ... 45
Chapter 4 Customized Corner Generation and Its Applications ... 47
4.1 Example problems ... 47
4.1.1 Digital logic delay ... 47
4.1.2 The pre-amplifier ... 49

4.2 PLS-based algebraic method ... 50
4.3 Optimization-based methods ... 58
4.3.1 Nonlinear classifier-assisted method ... 58
4.3.2 Direct optimization method ... 60
4.4 Summary ... 61
4.5 References ... 62
Chapter 5 Mathematical Formulation of Optimal Circuit Design ... 63
5.1 Deterministic Circuit Design Problem ... 63
5.2 Robust circuit design ... 64
5.2.1 Yield-constrained design ... 64
5.2.2 Worst-case Design ... 65
5.3 Example: Digital Circuit Sizing ... 66
5.3.1 Deterministic Digital Circuit Sizing ... 67
5.3.2 Yield-constrained Digital Circuit Sizing ... 67
5.4 References ... 67
Chapter 6 Solving the Yield-constrained Design Problem ... 70
6.1 Gradient projection method ... 70
6.1.1 The algorithm ... 72
6.1.2 Results and Discussion ... 72
6.2 Convex Approximation ... 75
6.2.1 Box Approximation ... 76
6.2.2 Ball Approximation ... 77
6.2.3 Bernstein Approximation ... 78
6.2.4 Dynamic Programming Formulation ... 79
6.2.5 Results and Discussion ... 81
6.3 Sequential Geometric Programming ... 83
6.3.1 Scaled Box Approximation ... 83
6.3.2 Sequential Geometric Programming ... 84
6.3.3 Importance Sampling in SGP ... 85
6.3.4 Results and Discussion ... 87
6.4 Summary ... 92
6.5 References ... 93

Chapter 7 Conclusions 95
7.1 Dissertation summary 95
7.2 Future research directions 96
7.3 References 96

Acknowledgements

Like most people, I started my journey of PhD in my 20's. The past 6 years are not just about acquiring knowledge, advancing technology or obtaining a degree. It is also a crucial period that shaped my personality and the view of life. I am particularly grateful that I can have Professor Costas J. Spanos as my PhD advisor. I once wrote in 2007 in an Email to a candidate requesting my comment on our team leader: "his (Costas) wisdom and personality are something I would like to strive for, but may never be able to achieve". Today, those words remain true and I am sure most people in our team including that candidate feel the same way.

I would like to thank another leader of our team, Professor Kameshwar Poolla. His energy and passion always motivate me to move forward even when I was in the most depressed mood. He treats us like his brothers and sisters and makes us feel, in his own words, "he is just another guy in town."

I want to thank Professor Laurent El Ghaoui for his help on the circuit optimization project and serving on my qualifying examination committee, Professor David Brillinger for serving on my dissertation committee, Professor Andrew Packard for serving on my qualifying examination committee, Professor Andrew Neureuther for helping me on my Master project and serving no my master thesis committee.

I also want to thank all BCAM members for their help in the past 6 years. Our incredible staff members, Changrui and Charlotte, have been helping BCAM students navigate their PhD careers for so many years. I simply cannot recall for how many times I bothered them for computer glitches and administrative problems.

I would like to thank all my friends at Berkeley. The list would be too long to fit here. But I want to thank Kenji Yamazoe in particular. He generously provided me suggestions and helps when I was stressed finding research topics for my PhD thesis.

Finally, I would like to express my deepest gratitude to my family. My fiancée Luosha Du witnessed and tolerated all my ups and downs. My mind went radical sometimes, and she was always there correcting me and making me a better person. My parents are very supportive for my PhD. They tried very hard to shield me from the troubles happening thousands miles away at my hometown. Without the support and understanding from my family, I could not have gone so far. This is for them.

Chapter 1 Introduction

1.1 Technology scaling and its impact

The semiconductor industry has gone through nearly fifty years of aggressive scaling [1] since the inception of the integrated circuit [2]. Technology scaling shrinks the size, enhances the speed, reduces the power, and drives down the cost of the integrated circuits. The trend of the changes has been largely exponential and is expected to continue for years to come. In particular, the number of transistors that can be integrated onto a single chip doubles every two years (Fig. 1.1), and this phenomenon is known as the Moore's Law [1], [3].

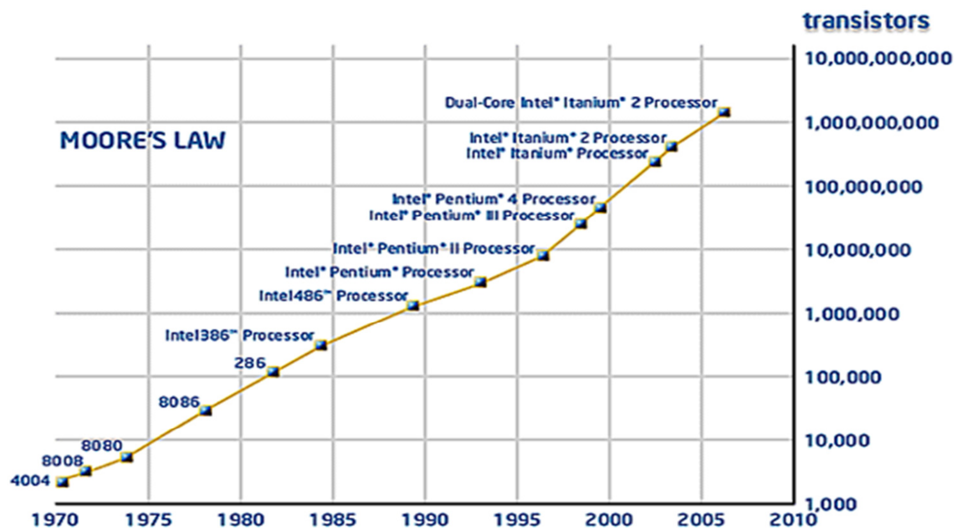

Fig. 1.1: Moore's Law of the integrated circuits [3].

Whereas the technology scaling makes integrated circuits more powerful and affordable, it poses significant challenges to both the design and the manufacturing of ICs. One of the most serious challenges is the variability arising from the manufacturing process [5]. Variability did not get much attention when the size of the transistor was relatively large, but becomes very pronounced as the technology scales toward the deep-submicron regime. For instance, the gate length of the transistor is now as small as 32 nm, and will go under 20 nm in the next few years. Patterning a feature at this scale requires extremely sophisticated lithography projection systems [6] and various correction schemes [7]. However, the molecular granularity of the photoresist always leaves the edge with considerable roughness (Fig. 1.2), and the variability due to this line edge roughness (LER) cannot be removed because of the fundamental structure of the photoresist material [8], [9]. Moreover, even if one can create a perfect pattern, the transistor might still not have the desired performance since the gate area merely accommodates around 100 dopant atoms, and any variation of the dopant at the atomic-scale, which is inevitable, can cause perceptible fluctuation in the overall property of the circuit [10]. This type of fundamental

variability is random and has a short correlation distance and contributes to the so-called intra-die variation [11]. There is another type of variability that characterizes the fluctuation between different dies, wafers or even batches of wafers, and has much slower varying spatial pattern. Parts of the variability can be predicted and modeled in a systematic manner, and others cannot [11]. The former is called systematic variability. But in order to use the model that is capable of characterizing the systematic variability, one must have knowledge of the position of the circuit on the die, the die on the wafer, etc. This information is in general not available to a circuit designer; therefore systematic variability should also be treated as random from the designers' standpoint.

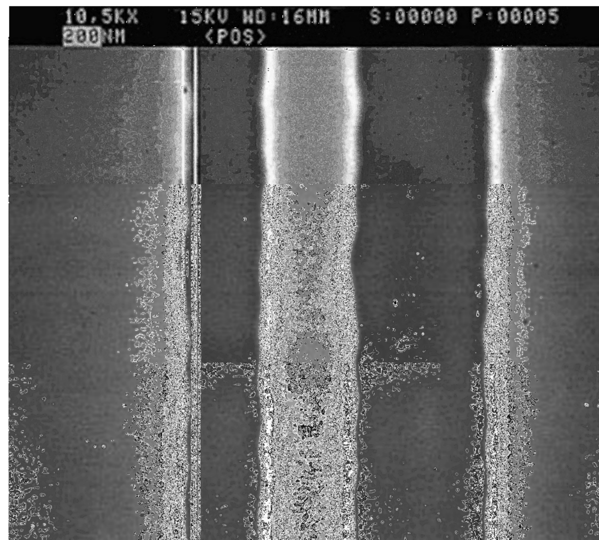

Fig. 1.2: An SEM image showing line edge roughness [9].

1.2 Robust circuit design flow and the outline of the thesis

Because of variability, the manufactured circuit may not have the performance that designers anticipated. It is necessary to take into account variability early in the design stage [12], [13] in order to ensure design robustness in the presence of variability. In order to achieve this, variability must be properly modeled, and the model has to be integrated into the design flow. This thesis is focused on using the existing variability model in the robust circuit design flow.

When a circuit designer is about to design a circuit, the topology or architecture is first chosen based on given specifications and design objectives. The parameters of the circuit components are then tuned to meet specifications. To test if the circuit is robust under the influence of variability, a set of representative worst-case scenarios are summarized as *corner models* [14], and the design is tested (through simulation) on these corner models. If it fails, either the circuit topology or the parameters will be changed. If it passes the corner model test, it will be further evaluated by Monte Carlo simulation to estimate the parametric yield. If the yield is below the desired value, one also goes back to change the

design, and if the yield is satisfactory, the design is ready for tape-out. The entire procedure is summarized in Fig. 1.3.

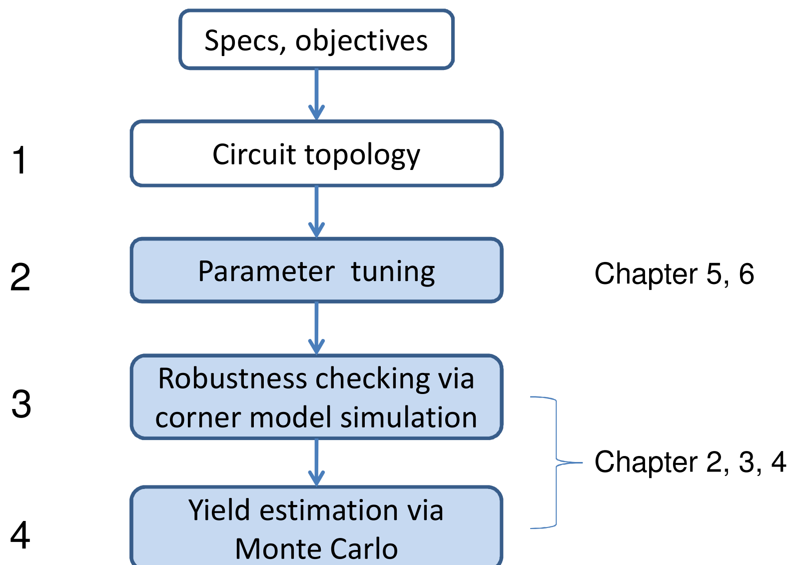

Fig. 1.3: Elements of the circuit design [15].

In the design steps shown in Fig. 1.3, steps 3 and 4 are about the verification of the circuit. In step 3, current state of the art corner model extraction methods are characterized using single transistor performances and cannot account for local intra-die variability [16]. In light of technology scaling, these corner extraction methodologies can no longer give a comprehensive estimate of the impact of variability, and they are particularly problematic in analog circuit design [17]. In step 4, the yield estimation problem is plagued by the inability of Monte Carlo methods in calculating rare event probability [18]. Rare event probability calculation is very important in estimating the failure rate of SRAM cell, which can be as low as 10^{-6} [19]. Existing methods such as importance sampling [20] can help alleviate the problem, but tend to be instable. We will address the challenges in statistical circuit verification in chapters 3 and 4. Before that, we will give an overview of the problem and the existing methods in chapter 2.

Step 1 and 2 in Fig. 1.3 are about the circuit optimization. We will not deal with the choice of the circuit topology in this thesis. Our goal is to optimize the parameters of the circuit keeping in mind the variability without being too conservative. Ideally the optimization should be carried out under a “yield constraint” [21]. Most of the existing circuit optimization methods [22] can only use a worst-case to penalize the optimization problem, and this inevitably gives rise to unnecessarily pessimistic solutions. Recent publications [23], [24] start to look at the yield-constrained problem, but they have to rely on overly simplified probability model. We intend to introduce methods that are flexible yet efficient in dealing with the yield-constrained circuit optimization problem. The proposed methods are given in chapter 6. Before chapter 6, an overview of the formulation and the challenges is given in chapter 5.

1.3 Main contributions of this work

One key contribution is the combination of partial least squares (PLS) regression [25] is with importance sampling in estimating the failure probability of SRAM cell. This novel method shows robust performance and is at least one order of magnitude faster than existing methods, and 10^4 times faster than Monte Carlo [26].

We also extend the definition of customized corner [27] to include both global and local variability. The customized corners found through PLS regression are used to predict the probability density function of the digital logic delay, and the prediction matches that of Monte Carlo, but is two orders of magnitude faster [28].

Another contribution is the introduction of convex approximation, representing the latest development in chance-constrained convex optimization, to the design automation community [29]. Based on the idea of the convex approximation, a sequential optimization method is proposed. The optimized circuit is shown to possess the desired parametric yield. The method is shown to be capable of handling circuit with thousands of gates and flexible in dealing with sophisticated variability models [21].

1.4 References

[1] G. E. Moore, “Progress in digital integrated electronics,” IEDM, pp. 11-13 (1975)

[2] R. N. Noyce, “Semiconductor device and lead structure,” U. S. Patent #2,981,877 (1961)

[3] T. Ghani, “Challenges and innovations in nano-CMOS transistor scaling,” *Nikkei Presentation*, October 2009. Available online: www.intel.com/technology/silicon/Neikei_Presentation_2009_Tahir_Ghani.ppt

[4] International technology roadmap for semiconductors, Available online: http://www.itrs.net/

[5] K. Qian, B. Nikolic, and C. J. Spanos, “Hierarchical modeling of spatial variability with a 45nm example,” Proc of SPIE, 727505 (2009)

[6] A. K-K. Wong, “Resolution enhancement techniques in optical lithography,” SPIE Publications (2001)

[7] N. B. Cobb, “Fast optical and process proximity correction algorithms for integrated circuit manufacturing,” PhD thesis, UC Berkeley

[8] A. Asenov, S. Kaya and A. R. Brown, "Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness," *IEEE Transactions on Electron Devices*, vol. 50, iss. 5, pp. 1254–1260 (2003)

[9] http://www.microtechweb.com/2d/lw_pict.htm

[10] M. Faiz. Bukhori, S. Roy and A. Asenov, "Simulation of Statistical Aspects of Charge Trapping and Related Degradation in Bulk MOSFETs in the Presence of Random Discrete Dopants," *IEEE Trans. Electron Dev.* vol. 57, iss. 4, pp. 795–803, (2010)

[11] P. Friedberg, W. Cheung, G. Cheng, Q. Y. Tang and C.J. Spanos, “Modeling spatial gate length variation in the 0.2um to 1.15mm separation range,” SPIE Vol. 6521, 652119 (2007)

[12] L. Liebman, "DfM, the teenage years," Proc of SPIE Vol. 6925, 692502 (2008)

[13] M. Orshansky, S. Nassif, D. Boning, "Design for Manufacturability and Statistical Design: A Constructive Approach," Springer (2008)

[14] N. Arora, "MOSFET modeling for VLSI simulation: theory and practice," reprint by World Scientific (2007)

[15] J. M. Rabaey, A. Chandrakasan and B. Nikolic, "Digital integrated circuits: a design perspective," second edition, Prentice Hall (2003)

[16] M. Sengupta, S. Saxena, L. Daldoss, G. Kramer, S. Minehane and Jianjun Cheng, "Application-specific worst case corners using response surfaces and statistical models," IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, pp. 1372, vol. 24 (2005)

[17] P. R. Gray, P. J. Hurst, S. H. Lewis and R. G. Meyer, "Analysis and design of analog integrated circuits," 4th edition, Wiley (2001)

[18] A. C. Davison, "Statistical models," Cambridge University Press, (2008)

[19] L. Dolecek, M. Qazi, D. Sha and A. Chandrakasan, "Breaking the simulation barrier: SRAM evaluation through norm minimization," ICCAD (2008)

[20] R. Kanj, R. Joshi and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," DAC, 69-72 (2006)

[21] Y. Ben, L. El Ghaoui, K. Poolla and C. J. Spanos, "Yield-constrained digital circuit sizing via sequential geometric programming," IEEE International Symposium on Quality Electronic Design, 114 (2010)

[22] M. D. M. Hershenson, S.P. Boyd and T. H. Lee, "Optimal design of a CMOS Op-Amp via geometric programming," IEEE Transactions on Computer-aided design of integrated circuits and systems, Vol. 20, No.1, (2001)

[23] J. Singh, V. Nookala, Z. Luo and S. Sapatnekar, "Robust gate sizing by geometric programming," DAC (2005)

[24] M. Mani and M. Orshansky, "A new statistical optimization algorithm for gate sizing," IEEE International Conference on Computer Design (2004)

[25] S. de Jong, "SIMPLS: an alternative approach to partial least squares regression," Chemometrics and Intelligent Laboratory Systems, 18, 251-263 (1993)

[26] Y. Ben, C. J. Spanos, "Partial least squares-preconditioned importance sampling for SRAM yield estimation," to appear in Design for Manufacturability through Design-Process Integration V, SPIE Advanced Lithography (2011)

[27] C.-H. Lin, M. V. Dunga, D. D. Lu, A. M. Niknejad and C. Hu, "Performance-aware corner model for design for manufacturing," IEEE Transacations on Electron Devices, Vol 56, 595-600 (2009)

[28] Y. Ben and C. J. Spanos, "Estimating the Probability Density Function of Critical Path Delay via Partial Least Squares Dimension Reduction," submitted to ISQED (2011)

[29] Yu Ben, Laurent El Ghaoui, Kameshwar Poolla, Costas J. Spanos, “Digital circuit sizing via robust approximation,” 3rd IEEE International Workshop on DFM&Y (2009)

Chapter 2 Overview of Statistical Circuit Verification

As stated in the Chapter 1, the two crucial verification steps in statistical circuit design are the yield estimation and corner model extraction. In this chapter, we will give an overview of these two topics, describe the challenges thereof, and briefly introduce our approaches. The detailed description of our methods will be given in Chapter 3 (fast yield estimation) and Chapter 4 (customized corner generation).

2.1 Statistical parametric yield estimation

Statistical circuit yield estimation starts with the assumption that the process parameters characterizing variability can be described by random variables. The process parameters can represent threshold voltage, oxide thickness, patterning variabvility, random dopant fluctuation etc. Throughout the thesis, we denote the process parameters with a random vector ξ, of which the distribution has the probability density function $g(\xi)$. The parametric yield is associated with certain circuit performance metric(s). We denote the circuit performance as $f(\xi)$, a function of process parameters ξ. Based on the value of performance $f(\xi)$, one can judge whether the circuit succeeds or fails to meet specifications. We define the region $\mathcal{F}$ in $f(\xi)$-space in a way such that if the value of $f(\xi)$ falls into region $\mathcal{F}$, it represents a failure. Calculating the yield Y is equivalent to calculating the failure probability $1 - Y = P_f = \Pr(\xi: f(\xi) \in \mathcal{F})$. Typically the yield is high and the failure probability is low. It is crucial for most applications that one can accurately assess the failure probability.

The golden standard to estimate failure probability is the Monte Carlo method [1]. In Monte Carlo, a random sequence of length $R, \{\xi_r\}_{r=1}^{R}$ is generated according to the probability density function $g(\xi)$ and $f(\xi)$ is evaluated for all ξ_r's. P_f is estimated as

$$\hat{P}_f = \frac{1}{R}\sum_{r=1}^{R} 1(f(\xi_r) \in \mathcal{F}), \tag{2.1}$$

where $1(f(\xi) \in \mathcal{F})$ is the indicator function of the failure event. The estimate $\hat{P}_f$ is an unbiased estimate of the failure probability P_f. The summands $1(f(\xi_r) \in \mathcal{F})$ are identically independent Bernoulli random variable with probability P_f. The central limit theorem teaches that if the number of samples R is large, the estimate $\hat{P}_f$ is approximately a normal random variable. To characterize a normal random variable, one only needs its mean and variance. The mean of $\hat{P}_f$ equals P_f. The variance of $\hat{P}_f$ is

$$\mathrm{Var}(\hat{P}_f) = \frac{1}{R}(P_f - P_f^2). \tag{2.2}$$

When we obtain through simulation the estimate $\hat{P}_f$, we can infer that the real failure probability P_f is within the interval

$$\left[\hat{P}_f\text{-}2\sqrt{\mathrm{Var}(\hat{P}_f)},\hat{P}_f+2\sqrt{\mathrm{Var}(\hat{P}_f)}\right] \tag{2.3}$$

with 95% confidence [2]. The factor 2 in front of the square root is chosen due to the approximate normality of $\hat{P}_f$. In order to achieve acceptable accuracy, the standard deviation of the estimate must be small compared to the estimate itself. If the standard deviation is required to be less than kP_f, then the required number of runs R is

$$R = \frac{1}{k^2}\frac{1 - P_f}{P_f}. \tag{2.4}$$

There are two important conclusions we can make from Eq. (2.4). The first is that the number of runs (sample size) to achieve a certain accuracy is independent of the dimension of the problem. No matter how high the dimension is, the run number is always given by (2.4). The second conclusion is that when the target failure probability P_f is low, the required sample size can be very high. If the failure probability becomes close to zero, the required sample size will explode. This poses a serious problem for the failure probability analysis of SRAM cell, as an SRAM cell has extremely low failure probability (10^{-5} to 10^{-6}). If the maximum standard error k is set to 10%, then the required sample size is at least at the order of 10^7. Fig. 2.1 explains why Monte Carlo method is plagued by rare event probability simulation: according to the bell-shaped probability density function, most of the samples are categorized as success and are far away from the success/failure boundary. These points are mostly "wasted" since they do not contribute to the information about the failure event. Many methods try to modify the plain Monte Carlo to overcome this difficulty. But before we describe those Monte Carlo variants, we shall introduce methods that are of a completely different style.

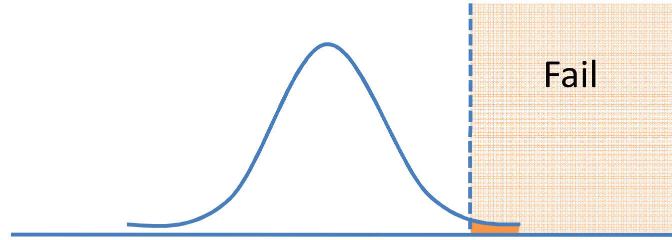

Fig. 2.1: Monte Carlo method in estimating rare event probability.

Unlike the Monte Carlo method relying purely on random simulation, the method proposed by Gu et. al. [3] converts the probability estimation to a geometric problem. The method works on the process parameter space, and the parameters are properly scaled and rotated such that they follow a uniform distribution and are independent from each other. The problem of calculating failure probability is now a problem of calculating the volume of the failure region. Gu et. al. propose to approximate the region with a collection of simplexes (Fig. 2.2). Upon obtaining the coordinates of the vertices, one can calculate the

volume of the simplex, and thus the failure probability. They show through an example of SRAM and ring oscillator that their method can exhibit speedups of 100-1000 times compared to Monte Carlo. Similar idea of exploring the process parameter space can also be found in Ref. [4] with application in statistical timing analysis. These methods abandon the stochastic nature of the problem and treat it as a purely deterministic modeling problem. Although they exhibit superior speed in low dimensional problems, they suffer severely from the curse of dimensionality [5], [6].

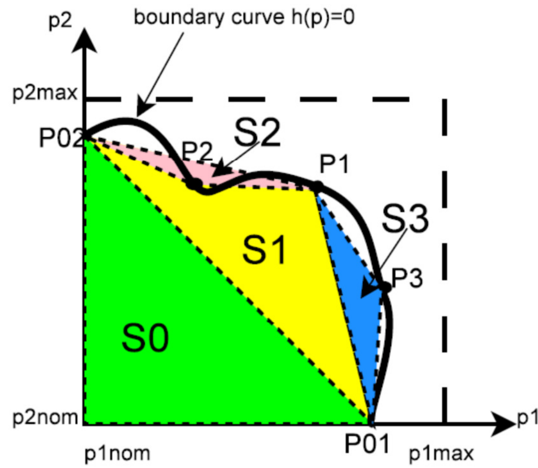

Fig. 2.2: Approximate the failure/success region with simplexes [3].

The third category of methods can be viewed as the combination of the above two, and it generally comes in the form of Monte Carlo with a “smart” sampling strategy. This includes importance sampling [7-9], stratified sampling [10-12], Latin hypercube sampling [13], [14] and statistical blockade [15]. The first three methods all aim to reduce the variance of the estimate by generating the samples based on a new probability density function $h(\xi)$. By contrast, statistical blockade still use the sample from the original distribution, but selectively to simulate them based on the classification result given by a support vector machine classifier [16]. Since it is in general the simulation that consumes most of the time, statistical blockade can save time by not simulating most of the samples. We shall only describe the detail of importance sampling here as we will extend this method in Chapter 3. In importance sampling, the sample is generated based on a new probability density function $h(\xi)$ (Fig. 2.3). If we view the calculation of the failure probability as the integral

$$P_f = \int 1(f(\xi) \in \mathcal{F}) g(\xi) d\xi, \tag{2.5}$$

then it is equivalent to the following expression

$$P_f = \int \left[1(f(\xi) \in \mathcal{F}) \frac{g(\xi)}{h(\xi)} \right] h(\xi) d\xi. \tag{2.6}$$

If we draw samples from the distribution described by $g(\xi)$, the quantity that enters the summation in Monte Carlo is $1(f(\xi) \in \mathcal{F})$. By the same token, if we draw samples from the distribution described by $h(\xi)$, the summand should be the expression within the square bracket $1(f(\xi) \in \mathcal{F})\, g(\xi)/h(\xi)$. Therefore, we have the following estimate for failure probability based on the samples from $h(\xi)$:

$$\hat{P}_f = \frac{1}{R} \sum_{r=1}^{R} w(\xi_r) 1(f(\xi_r) \in \mathcal{F}), \tag{2.7}$$

where $w(\xi_r) = g(\xi_r)/h(\xi_r)$. The variance of the estimate is given by

$$\mathrm{Var}(\hat{P}_f) = \frac{1}{R^2} \left(\sum_{r=1}^{R} \left(w^2(\xi_r) 1(f(\xi_r) \in \mathcal{F}) \right) - R P_f^2 \right). \tag{2.8}$$

The goal of importance sampling is to find the best $h(\xi)$ such that the variance in Eq. (2.8) is minimized. In theory [17], the variance can be reduced to zero with a single sample if we already know the failure probability P_f. But in reality we can never have that knowledge in advance (if we do, there will be no need to estimate it). So, heuristics are used to find a reasonably good $h(\xi)$. A popular norm-minimization heuristic proposed in [8] performs uniform random search in an extended space, screens out those points that give rise to failure events, and finds among them the point with the smallest Euclidean norm. The norm-minimization method is intended to find the failure point that is the most likely to fail. However, relying purely on a haphazard search can easily miss the intended point in a high dimensional space, and a large number of random simulations is needed to determine the shift point.

We will introduce our methods for parametric yield estimation in Chapter 3. The proposed methods extend the frontiers of all three categories as discussed above. For the method based on pure stochastic simulation, we propose to use extrapolation methods based on the assumption that the distribution tail of the concerning quantity has a certain smooth shape, which can be described by a closed-form function. For the method based on exploration of the random parameter space, we propose to model the performance function using binary tree-based algorithms, hoping the logarithmic complexity of the tree algorithm can compensate the exponential complexity incurred by dimensionality. For the hybrid method involving both stochastic simulation and deterministic exploration, we propose to precondition the importance sampling method using machine learning dimension reduction techniques.

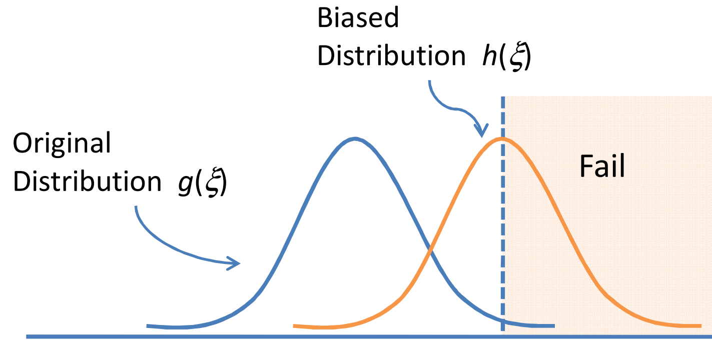

Fig. 2.3: Illustration of importance sampling.

2.2 Corner models

The yield estimation as described in the last section is used in the final verification phase of the circuit design. For a quick check of the circuit robustness against process variation, yield estimation is still overly time-consuming. For the verification task where speed is of major concern, circuit designers opt to use corner models representing the worst-case scenario arising from process variability [18-21] to run circuit simulations and verify the robustness of the circuit.

It is common in industry to take account of the statistical variation using five types of worst case models, each of which is defined by a two letter acronym title describing the relative performance characteristics of the p- and n-channel devices. The letters describe the operation as being typical or nominal (T), fast (F) or slow (S). For instance, "FS" denotes the situation where the n-device is fast and the p-device is slow. Usually, the design is done in typical p, typical n (TT) condition, and checked for all four corners (SS, FF, FS and SF). What is assumed is that given a yield level, the four corners bound all worst possible scenarios of extreme performance. If the circuit can pass the test on these corners, then certain yield is guaranteed. This methodology has been successfully adopted in circuit design practice,, yet it faces two important challenges.

The first challenge is related to the definition and extraction of the corners. Existing process corner models are defined and extracted using the measurement data on individual transistors [18]. For digital logic circuits, the worst-case performance commonly measured by critical delay occurs when single transistors are also at the worst case. But for analog circuits, it is not guaranteed that the worst case will be found at any of these corners [24]. Depending on the metric (band width, common mode rejection ratio, gain etc.), the worst case can be worse that any that the corners predict (Fig. 2.4). This means that even if the circuit passes the four corner cases, it is not guaranteed to have the desired robustness.

The second challenge comes from the fact that local variation due to random dopant fluctuation and line-edge roughness is becoming more pronounced [22], [23]. The corner models as commonly used in the industry can only be used to describe global variation. To account for local variations, designers have to resort to the more expensive Monte Carlo simulation. Direct extension of the corner definition to include local random variables is

impossible as the number of corners increase exponentially with the number of random variables.

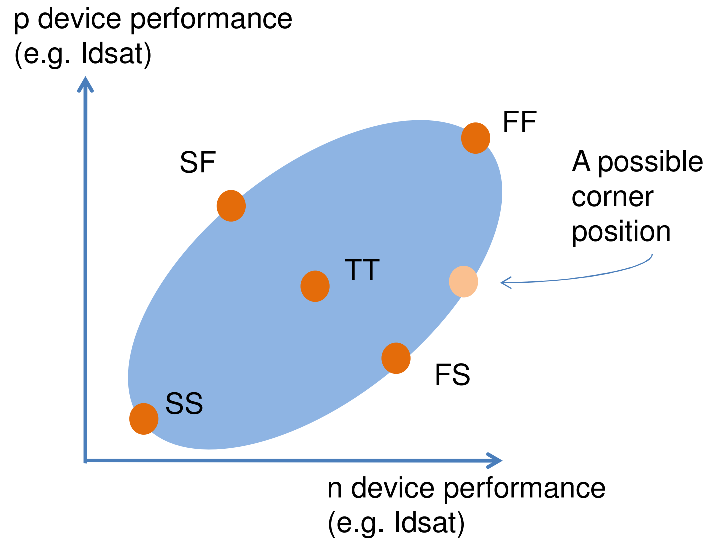

Fig. 2.4: Corner models commonly used in the industry.

To address the first challenge, [25] and [26] propose to define the process corner models based on certain performance metrics. However, the methods proposed in [25] and [26] do not naturally scale to high dimension, hence cannot be used in the situation where local variation must be considered. To the best of our knowledge, [27] is the only work that addresses both challenges. Ref. [27] proposes to use a non-linear operating point to estimate the delay quantiles. The method is limited by the fcat that it needs to pick the critical path in advance since it involves first-order derivative calculation requiring a differentiable delay function. Moreover, that method requires an estimate of sensitivity with respect to every variable, which can be a serious problem if the dimensionality is high. These facts limit the scope of application for the method proposed in Ref. [27].

In this work we propose is a set of methods suitable for different situations. When the dimensionality is high and the nonlinearity of the performance function is weak (as in digital logic delay), we propose an approach that uses partial least squares (PLS) regression [28] to reduce the problem space dimensionality and ultimately use a reduced model to estimate the corner points. When the nonlinearity of the performance function is strong as commonly found in analog circuits, we propose to use nonlinear classifier [29] to describe the contour shape of the function and also use direct optimization to locate the corner points.

2.3 References

[1] A. C. Davison, “Statistical models,” Cambridge University Press, (2008)

[2] R. Kanj, R. Joshi and S. Nassif, “Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events,” DAC, 69-72 (2006)

[3] C. Gu and J. Roychowdhury, "An efficient, fully nonlinear, variability-aware non-Monte-Carlo yield estimation procedure with applications to SRAM cells and ring oscillators," Asia and South Pacific Design Automation Conf., 754-761 (2008)

[4] J. A. G. Jess, K. Kalafala, S. R. Naidu, R. H. J. M. Otten and C. Visweswariah, "Statistical timing for parametric yield prediction of digital integrated circuits," DAC (2003)

[5] C. Gu, private communication

[6] T. Hastie, R. Tibshirani and J. Friedman, "The elements of statistical learning: data mining, inference, and prediction," second edition, Springer (2009)

[7] R. Kanj, R. Joshi and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," DAC, 69-72 (2006)

[8] L. Dolecek, M. Qazi, D. Sha and A. Chandrakasan, "Breaking the simulation barrier: SRAM evaluation through norm minimization," ICCAD (2008)

[9] T. Date, S. Hagiwara, K. Masu and T. Sato, "Robust importance sampling for efficient SRAM yield analysis," IEEE International Symposium on Quality Electronic Design, 15 (2010)

[10] M. Keramat and R. Kielbasa, "A study of stratified sampling in variance reduction techniques for parametric yield estimation," IEEE Transactions on Circuits and Systems, pp 575, Vol. 45 (1998)

[11] D. E. Hocevar, M. R. Lightner, and T. N. Trick, "A study of variance reduction techniques for estimating circuit yields," IEEE Trans. Computer-Aided Design, vol. CAD-2, pp. 180–192, July (1983)

[12] M. L. Stein, "An efficient method of sampling for statistical circuit design," IEEE Trans. Computer-Aided Design, vol. CAD-5, pp. 23–29, Jan (1986)

[13] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in analysis of output from a computer code," Technometrics, vol. 21, no. 2, pp. 239–245, May (1979)

[14] M. Keramat and R. Kielbasa, "Latin hypercube sampling Monte Carlo estimation of average quality index for integrated circuits," Analog Integrated Circuits Signal Processing, vol. 14, nos. 1/2, pp. 131–142, (1997)

[15] A. Singhee and R. A. Rutenbar, "Statistical blockade: a novel method for very fast Monte Carlo simulation of rare circuit events, and its application," Design Automation and Test in Europe (2007)

[16] C. M. Bishop, "Pattern recognition and machine learning," Springer (2006)

[17] R. Srinivasan, "Importance sampling: applications in communications and detection," Springer (2002)

[18] N. Arora, "MOSFET modeling for VLSI simulation: theory and practice," reprint by World Scientific (2007)

[19] A. Nardi, A. Neviani, E. Zanoni, M. Quarantelli and C. Guardiani, “Impact of unrealistic worst case modeling on the performance of VLSI circuits in deep sub-micron CMOS technology,” IEEE Trans. Semicond. Manuf., vol. 12, no. 4, pp. 396–402, Nov (1999)

[20] A. Dharchoudhury and S. M. Kang, “Worst-case analysis and optimization of VLSI circuit performances,” IEEE Trans. Computer-Aided Design, vol. 14, no. 4, pp. 481–492, Apr (1995)

[21] S. R. Nassif, A. J. Strojwas, and S.W. Director, “A methodology for worst case analysis of integrated circuits,” IEEE Trans. Computer-Aided Design, vol. CAD-5, no. 1, pp. 104–113, Jan (1986)

[22] A. Asenov, "Random dopant induced threshold voltage lowering and fluctuations," Electron Devices, IEEE Transactions on, vol. 45, pp. 2505-2513 (1998)

[23] A. R. Brown, S. Kaya, A. Asenov, J. H. Davies and T. Linton, "Statistical Simulation of Line Edge Roughness in Decanano MOSFETs," in Proc. Silicon Nanoelectronics Workshop, Kyoto, Japan, June 10-11 (2001)

[24] X. Li, P. Gopalakrishnan, Y. Xu and L. T. Pileggi, “Robust analog/RF circuit design with projection-based performance modeling,” IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, pp 2, vol. 26 (2007)

[25] C.-H. Lin, M. V. Dunga, D. D. Lu, A. M. Niknejad and C. Hu, “Performance-aware corner model for design for manufacturing,” IEEE Transactions on Electron Devices, Vol 56, 595-600 (2009)

[26] M. Sengupta, S. Saxena, L. Daldoss, G. Kramer, S. Minehane and Jianjun Cheng, “Application-specific worst case corners using response surfaces and statistical models,” IEEE Transactions on Computer-aided Design of Integrated Circuits and Systems, pp. 1372, vol. 24 (2005)

[27] R. Rithe, J. Gu, A. Wang, S. Datla, G. Gammie, D. Buss and A. Chandrakasan, “Non-linear operating point statistical analysis for local variations in logic timing at low voltage,” DATE, 965-968 (2010)

[28] R. Rosipal and N. Kramer, “Overview and recent advances in partial least squares,” Subspace, Latent Structure and Feature Selection, Springer, 34-51 (2006)

[29] B. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in Proc. 5th Annual ACM Workshop on Computational Learning Theory, New York, NY: ACM Press, pp. 144-152 (1992)

Chapter 3 Fast Circuit Yield Estimation

In this chapter we address the challenge in circuit yield estimation, or equivalently, failure probability estimation. The circuit yield estimation is prevalent in all robust circuit analysis and design problems, among which the yield calculation of SRAM cell is considered the most challenging. The failure probability of an SRAM cell is extremely low, but the small failure probability can be a yield limiting factor for a large array of SRAM cells as commonly seen in processors and ASICs. In light of this fact, we will focus on the yield estimation problem for an SRAM cell. We will first introduce the problem definition and formulate the failure probability of an SRAM cell in a way that can be readily handled by the proposed methods. We shall describe three different methods to tackle the yield estimation problem. The first method is based on extrapolation. The second method explores the variability space using binary tree-based methodology. The third method uses partial least squares regression as a dimensionality reduction step for importance sampling. For every method, we will describe the theoretical background, implementation details, and will compare the pros and cons. Among these methods, we find the partial least squares regression-preconditioned importance sampling to be the most suitable for the yield estimation of SRAM cells.

3.1 Problem definition

We are going to illustrate the proposed methods through the failure probability estimation of an SRAM cell. The standard 6-T SRAM cell design, shown in Fig. 3.1(a), has the following parameters: $L_{1-6} = 50$ nm, $W_{1,2,4,5} = 200$ nm, $W_{3,6} = 100$ nm, $V_{dd} = 1$ V and $V_{th0} = 0.3$ V. The circuit is simulated in Hspice with PTM 45 nm low voltage model [1]. The design parameters given above are specified only for the purpose of illustration, and should not be regarded as a good design example.

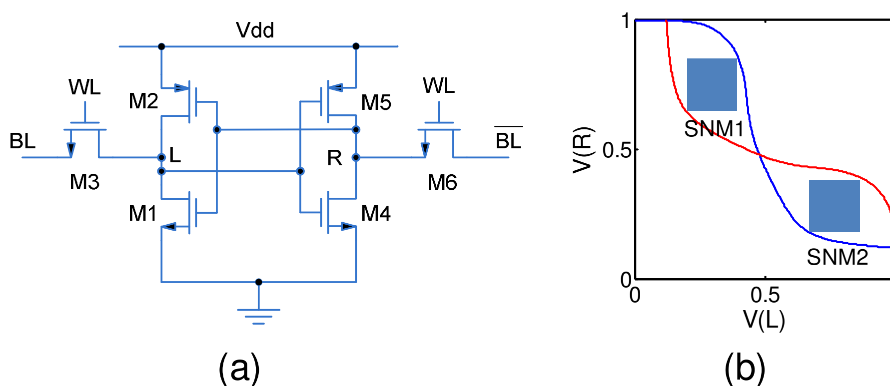

Fig. 3.1: SRAM example. (a) A 6-T SRAM cell schematic. (b) Butterfly curve in read operation. The static noise margin (SNM) is defined as the side length of the maximal inscribed square in either of the two lobes.

The variability is assumed to stem from the threshold voltage fluctuation of every transistor. Due to the hierarchical structure of the variability [2], it is important to consider

both the global variation and the local mismatch. We assume that the threshold voltage fluctuation of the 6 transistors is described by a random vector $\xi \in \mathbb{R}^6$, the component of which is

$$\xi_i = a_g \epsilon_g + a_i \epsilon_i, \tag{3.1}$$

where $i = 1 \dots .6$, a_g and a_i are the amplitudes of the global and local variations, respectively, and ϵ_g, ϵ_i are independent standard normal random variables. The normality assumption is only for the convenience of simulation. We can use any realistic distribution as long as random samples can be easily generated from it. As a result of this formulation, ξ_i's are correlated random variables. In our analysis, as an example we assume that $a_g = a_i = 50$ mV.

The SRAM cell is the most prone to fail during the read operation, when both bit lines (BL) are pre-charged and the word line (WL) is selected [3]. The performance metric of interest is the static noise margin (SNM) during the read operation. It is also a function of ξ. As shown in Fig. 3. 1(b), if the voltage transfer curves of the left and right inverters are plotted on the same graph, the side lengths of the largest inscribing squares that can fit into the two lobes are called $SNM1$ and $SNM2$, respectively. SNM is defined as

$$SNM = \min(SNM1, SNM2). \tag{3.2}$$

Theoretically, cell failure happens when SNM becomes smaller than zero. But in reality we have to guard against additional perturbation induced by supply voltage fluctuation; therefore we define the failure as

$$SNM < SNM_{min}. \tag{3.3}$$

In our example analysis, SNM_{min} is set to 50 mV.

The failure probability that we are interested in is

$$P_f = \Pr(\xi: SNM(\xi) < SNM_{min}). \tag{3.4}$$

Considering the definition of SNM, this definition can be rewritten as

$$\begin{aligned} P_f &= \Pr(\xi: SNM(\xi) < SNM_{min}) \\ &= \Pr(\xi: SNM1(\xi) < SNM_{min}) + \Pr(\xi: SNM2(\xi) < SNM_{min}) \\ &- \Pr(\{\xi: SNM1(\xi) < SNM_{min}\} \cap \{\xi: SNM2(\xi) < SNM_{min}\}). \end{aligned} \tag{3.5}$$

Because the probability that both lobes diminish at the same time is orders of magnitude smaller than the probability that one of them fails [4], the last term in Eq. (3.5) can be eliminated. Taking into account the symmetry between $SNM1$ and $SNM2$, it suffices to calculate

$$P_f = 2\Pr(\xi: SNM1(\xi) < SNM_{min}). \tag{3.6}$$

For the purpose of comparison with the literature, we will use the definition in Eq. (3.6) as used in [4], [5]. However, our method applies to the general definition Eq. (3.4) as well.

Now our focus is to calculate Eq. (3.6), and the method to do so is described in the next 3 sections.

3.2 Extrapolation methods

Extrapolation methods are based on the assumption that the failure probability is a smooth function of a certain parameter, the parameter being either a variable controlled by the user or an artificial threshold defining the success or failure.

We will explore two extrapolation methods in this section. In the first method we change the operating condition by tuning the supply voltage such that the failure probability is high enough to be estimated using common methods (e.g. Monte Carlo). A link is forged between failure probability and supply voltage allowing the real failure probability to be estimated. In the second method, we extend the idea of extreme value theory and treat the failure probability as a function of threshold value delimiting the success and failure.

3.2.1 Extrapolation by biasing operating condition

As we show in section 2.1, the difficulty in estimating the failure probability of an SRAM cell is that the failure event is extremely rare, hence the failure probability is low. If we manually change the operating condition of the SRAM cell, making the failure probability artificially high, then we can adopt common methods such as Monte Carlo to estimate that number. Building upon that estimation, we deduce what the real failure probability is.

For SRAM cells, the operating condition can be altered by changing the supply voltage V_{dd}. At low supply voltage, the SRAM cell is more likely to fail, making the failure probability significantly higher than under nominal conditions (Fig. 3.2). We can use regular Monte Carlo to estimate the failure probability at several different V_{dd} values with low computational cost. Then we extrapolate the real failure probability based on an empirical relation between failure probability and supply voltage. However, building an empirical model directly between failure probability P_f and supply voltage V_{dd} is unlikely to succeed. The reason is that SNM is a highly nonlinear function of V_{dd}, and a pure empirical model is incapable of capturing the nonlinearity. On the other hand, we do have the capability of deducing this relation using a single-run simulation at various V_{dd} levels. Based on these arguments, we propose a two-step procedure as illustrated in Fig. 3.3.

In step 1, we map the supply voltage V_{dd} to the mean value of SNM at that supply voltage:

$$SNM = f(V_{dd}). \tag{3.7}$$

This can be done by simple simulation without variation. If the estimation is conducted experimentally, this step can be carried out by measuring the SNM on sampled cells and calculate the mean value. Since it is the mean value that we need at this step, the sample size does not suffer from the rare event and is generally small.

In step2, we use an empirical model to link the mean value of SNM to failure probability P_f:

$$P_f\big(f(V_{dd})\big) = \exp(\beta_0 + \beta_1 f(V_{dd}) + \beta_2 f^2(V_{dd})), \qquad (3.8)$$

where the parameters β_0, β_1 and β_2 are parameters that can be determined by least-squares fitting.

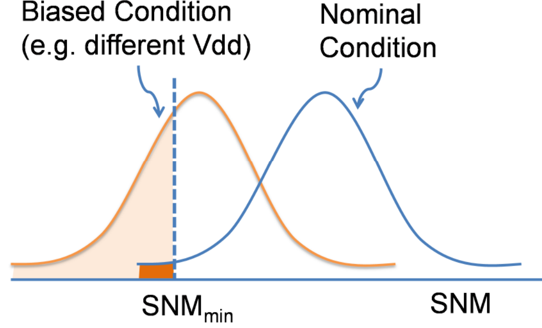

Fig. 3.2: The distribution of SNM under nominal condition and biased operating condition.

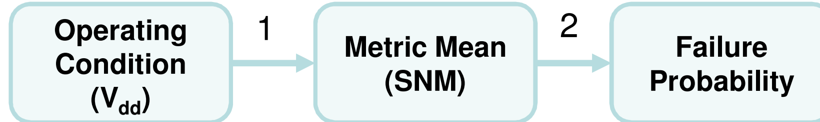

Fig. 3.3: Flowchart for the link between biased condition and nominal condition.

We apply the above method to the SRAM cell as described in section 3.1. The biased conditions are controlled by the supply voltage V_{dd}. We let V_{dd} take values 0.5, 0.6, 0.7 and 0.8 V, and evaluate the failure probability at every V_{dd} level. In accordance with the two-step method, the mean SNM values are simulated at these four supply voltage levels, and also at the nominal $V_{dd} = 1V$ (upper panel of Fig. 3.4). The failure probability as a function of mean $SNM = f(V_{dd})$ is plotted in the lower panel of Fig. 3.4. These four points are used to fit the formula in Eq. (3.8) to find parameters β_0, β_1 and β_2. Upon fitting Eq. (3.8), we predict the failure probability at nominal operating condition. The final relation between failure probability P_f and V_{dd} is summarized in Fig. 3.5, where the solid line is the model given by Eq. (3.8), the squares are Monte Carlo simulation at lower supply voltages and the diamond is obtained by 2×10^6 Monte Carlo at nominal condition. The error bar around the diamond indicates the 95% confidence interval of the Monte Carlo estimation. We can see that the model as shown in solid line gives an accurate estimation at the nominal operating condition. It is interesting to note that, if we made the extrapolation directly on $P_f \sim V_{dd}$ relation, we would roughly have a straight line in Fig. 3.5 based on the first four points. An extrapolation using the straight line would significantly underestimate the failure probability. By contrast, we can achieve an accurate estimate by adopting the two-step strategy as described above.

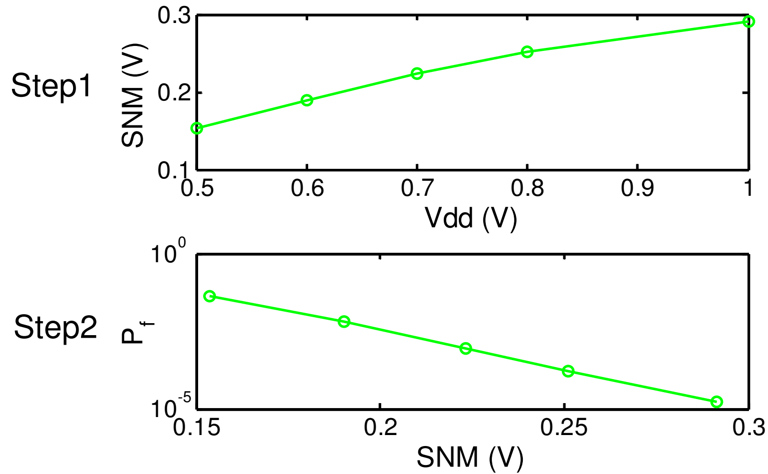

Fig. 3.4: Two-step extrapolation in SRAM failure probability estimation.

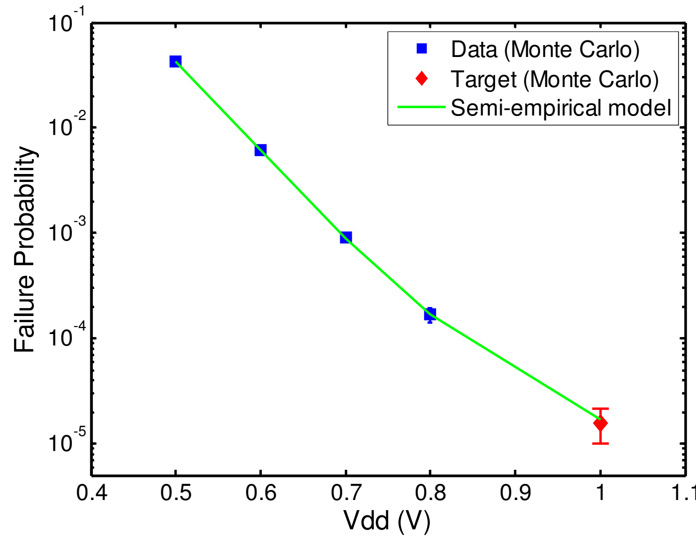

Fig. 3.5: Failure probability as a function of supply voltage V_{dd}.

Compared to other methods to be introduced later, this method has a unique advantage in that it can be used in simulation as well as when actual measurements are involved. This is because the operation condition is changed by supply voltage, which can also be varied experimentally, and all the parameters that are present in the model can also be obtained by measurements. In other methods, there are always some parameters that are beyond experimenters' control, and can only be handled in simulation.

3.2.2 Extrapolation using extreme value theory

The method we will explore in this section is based on extreme value theory (EVT). The original theory of EVT was established around 1975 [6], [7], but it did not gain the attention of design automation researchers until recently [8], [9]. The basic idea of EVT is as follows.

Suppose we have a random variable X, and two real numbers x and u with u fixed, then the probability for $X \geq x + u$ given $X \geq u$ is a function of x. We denote it as $F_u(x)$:

$$F_u(x) = \Pr(X \geq x + u | X \geq u). \tag{3.9}$$

As we increase the value of u, EVT teaches that the function will converge to a parametric function $1 - G_{a,k}(x)$:

$$\lim_{u\to\infty} \sup_{x\geq 0} \left| F_u(x) - \left(1 - G_{a,k}(x)\right) \right| = 0. \tag{3.10}$$

where $G_{a,k}(x)$ is the cumulative distribution function of the Generalized Pareto Distribution (GPD):

$$G_{a,k}(x) = \begin{cases} 1 - \left(1 + \frac{kx}{a}\right)^{-\frac{1}{k}}, & k \neq 0 \\ 1 - e^{-\frac{x}{a}}, & k = 0. \end{cases} \tag{3.11}$$

The relation holds for almost all commonly-seen distributions [10]. Among the two expressions of $G_{a,k}(x)$, it is the most common to use the exponential parametric form. If we expand the conditional probability in Eq. (3.9), we find

$$\Pr(X \geq x + u) = \Pr(X \geq u) F_u(x). \tag{3.12}$$

This shows that the probability for X to be greater than a threshold that is x extra larger than u equals the probability for $X \geq u$ times an exponential function of x provided u is large enough.

To apply the theory to SRAM yield estimation, we use M_0 in place of SNM_{min} for the sake of simplicity. Then the quantity of interest is

$$\Pr(SNM < M_0). \tag{3.13}$$

Compared to Eq. (3.12), what we are trying to estimate is the probability in the lower rather that the upper tail of the distribution. So we will set a *small* threshold M, for which the probability $\Pr(SNM < M)$ is easy to estimate, and then estimate Eq. (3.13) using the relation derived from Eq. (3.12):

$$\Pr(SNM < M_0) = \Pr(SNM < M) \exp\left(-\alpha(M - M_0)\right), \tag{3.14}$$

where α is a fitting parameter. If we use multiple M's and calculate the corresponding probabilities $\Pr(SNM < M)$, then we can estimate the parameter α in Eq. (3.14), hence make the estimation of the failure probability at the nominal threshold M_0 (Fig. 3.6). But for Eq. (3.14) to be valid M should be small and close to the tail of the distribution. To this end, one normally adopts an arbitrary criterion of choosing the largest value of M: M_{upper} such that $\Pr\left(SNM < M_{upper}\right) \leq 0.01$ [8], [9]. On the other hand, if M is too small, then the failure probability at that point can be inaccurate or expensive to estimate. We denote the smallest value of M as M_{lower} and will discuss its choice later in this section. As shown in

Fig. 3.7, multiple M values are picked between M_{lower} and M_{upper}, and the failure probabilities are estimated using regular Monte Carlo for every M to fit Eq. (3.14).

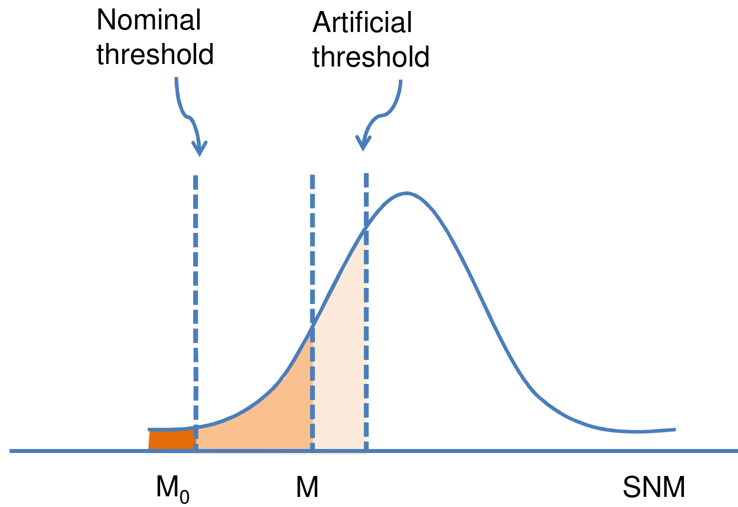

Fig. 3.6: Change the threshold M such that the probability $\Pr(SNM < M)$ is large and easy estimate.

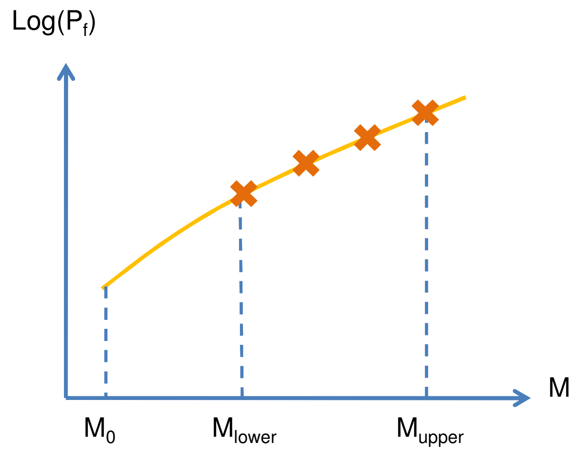

Fig. 3.7: Use multiple threshold points to estimate parameter α and the real failure probability $\Pr(SNM < M_0)$.

There are two issues that need some discussion. The first is about the validity of Eq. (3.14). The relation given by Eq. (3.14) is true when M approaches negative infinity. In reality, however, we cannot use an infinite number and we must use a number that is reasonably low. To the best of our knowledge, there is no known general method giving guidance to choose this number, and most people resort to arbitrary criterion, such as the probability being less than 0.01. Therefore, the arbitrary criterion of choosing M_{upper} cannot guarantee the exact exponential relation. In fact, we observe that the failure probability is a super-exponential function of M (Fig. 3.8). In light of this fact, we propose

to use quadratic-exponential model to account for this deviation from exponential characteristics of the tail of the distribution:

$$P_f = \Pr(SNM < M) = \exp(\beta_0 + \beta_1 M + \beta_2 M^2), \qquad (3.15)$$

where the parameters β_0, β_1 and β_2 are parameters that can be determined using least-squares fitting. We call the model in Eq. (3.15) the extended EVT in comparison to the original theory of EVT. Both extended EVT and original EVT methods are used in the SRAM example. We use 10 evenly distributed M values between 0.084 and 0.14, and estimate the quantity $\Pr(SNM < M)$ for these M values using the first 1×10^5 samples from a larger 1×10^6 sample pool. The results are plotted in Fig. 3.9. We can observe that the original EVT extrapolation gives an estimation of the failure probability at $M_0 = 0.05$ that is higher than that given by the extended EVT. If we compare the real failure probability as shown in Fig. 3.8 (around 1.6×10^{-5}), we can see that the extended EVT gives better estimation than the original EVT, because it accounts for the fact that the M values we use in extrapolation are not infinitely far into the tail of the distribution.

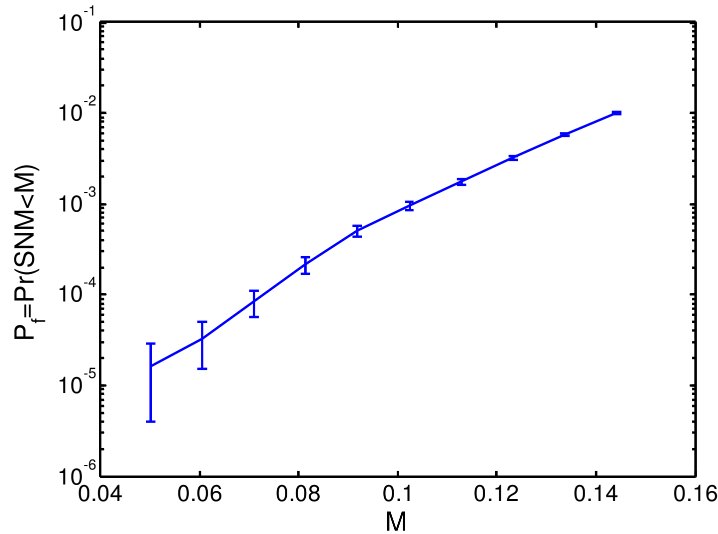

Fig. 3.8: The failure probability $P_f = \Pr(SNM < M)$ as a function of M as estimated by 1×10^6 Monte Carlo. The error bars indicate the standard error.

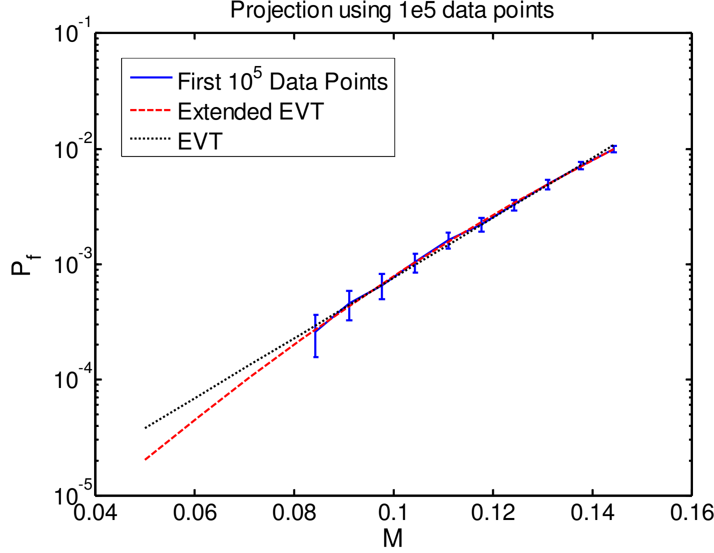

Fig. 3.9: Extrapolation given by extended EVT and the original EVT. The goal is to find the failure probability at $M = 0.05$. The original EVT overestimates the failure probability.

The second issue that must be addressed is the choice of the number of M values. It appears that if we use more points, we can infinitely reduce the prediction error. Since we use different M as threshold values on a single data set to count the number of points smaller than that value, it would seem as if we can improve the accuracy by merely involving more threshold values. Obviously this cannot be true, and we are going to show that the correlation between the failure probability estimation for different M's is the reason that prevents us from using arbitrarily large number of points. For the discussion of this issue, we denote the exponent $\beta_0 + \beta_1 M + \beta_2 M^2$ in Eq. (3.15) as y. Suppose the estimation of P_f can be expressed as a summation of mean value $\bar{P_f}$ and an error term ΔP_f, then y can be expressed as

$$y = \bar{y} + \epsilon = \log P_f \approx \log \bar{P}_f + \frac{\Delta P_f}{\bar{P}_f}. \tag{3.16}$$

The error term ϵ is related to the error in P_f by

$$\epsilon = \Delta P_f / \bar{P}_f. \tag{3.17}$$

It shows that the absolute error of the exponent y is the standard error of the failure probability estimation.

The mean value of y is a linear combination of $\beta = (\beta_0\ \beta_1\ \beta_2)^T$, and can be expressed by the standard linear model

$$\bar{y} = (1\ M\ M^2)\begin{pmatrix}\beta_0\\ \beta_1\\ \beta_2\end{pmatrix}. \tag{3.18}$$

If we have multiple M's, we can stack the row vector $\left(1\ M\ M^2\right)$ and call it matrix X following the convention of standard linear model [10]. The complete model for estimating β is

$$y = X\beta + \epsilon, \tag{3.19}$$

of which the least square solution is

$$\hat{\beta} = (X^T X)^{-1} X^T (y + \epsilon). \tag{3.20}$$

The purpose of this fitting is to predict the value of y_p at M_0. Based on the above formulation, if we assign $x_p = (1\ M_0\ M_0^2)$, we have

$$y_p = x_p \hat{\beta} = x_p (X^T X)^{-1} X^T y + x_p (X^T X)^{-1} X^T \epsilon, \tag{3.21}$$

and the prediction error is

$$\epsilon_p = \sqrt{x_p \left(X^T X\right)^{-1} X^T Cov(\epsilon, \epsilon^T) X \left(X^T X\right)^{-1} x_p^T}. \tag{3.22}$$

Considering the relation in Eq. (3.17), we find that the quantity given by Eq. (3.22) is the standard error of the P_f estimation. If the covariance matrix $Cov(\epsilon, \epsilon^T)$ has only diagonal elements, then we can reduce the standard error without limit by increasing the number of M's. However, the off-diagonal elements of the covariance matrix are significant compared to that of diagonal elements as shown in the calculation given in the appendix 3.A of this chapter. This fact limits the extent of improvement in terms of accuracy. We shall use Eq. (3.22) to evaluate the accuracy of the method.

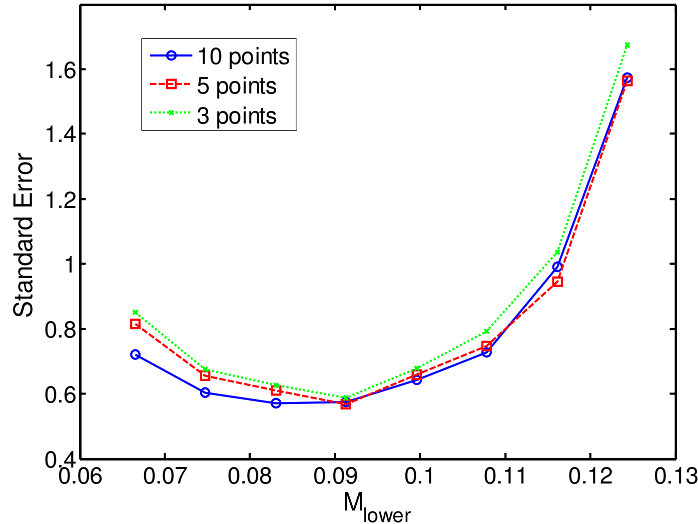

Fig. 3.10: Standard error of failure probability estimation at different M_{lower}. The artificial thresholds (M) are evenly distributed between M_{lower} and M_{upper}. The parameter M_{upper} is chosen such that Pr(SNM<M_{upper}) =0.01.

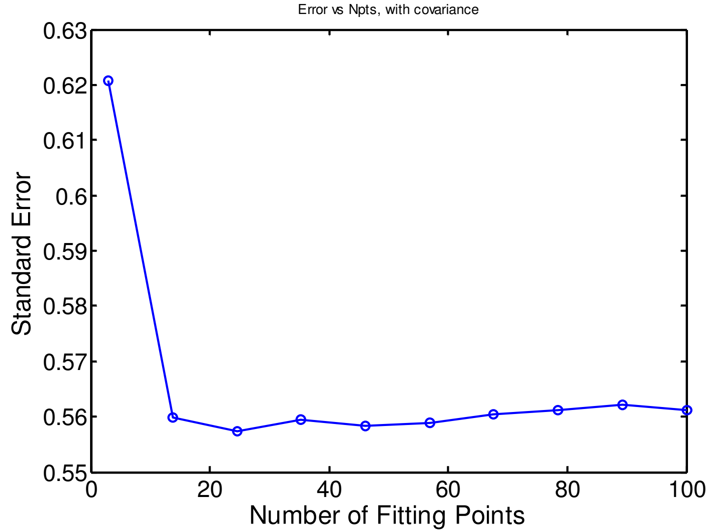

Fig. 3.11: Standard error of failure probability estimation given by different number of fitting points. M_{lower} is chosen to be the 0.09 as indicated by Fig. 3.10.

The first parameter being tested is the smallest value of M used in parameter fitting. This is the parameter M_{lower} in Fig. 3.7. The largest value M_{upper} is fixed such that $\Pr(SNM < M_{upper}) = 0.01$. The values of M are evenly distributed between M_{lower} and M_{upper}. The resulting standard errors for estimating $\Pr(SNM < M_0)$ using 10, 5 and 3 values of M are shown in Fig. 3.10. What we observe is that there is an optimal M_{lower} such that the standard prediction error is minimized. For M_{lower} value that is too high, the predicted M_0 is too far from the data region; by contrast, if M_{lower} is too low, the standard error of the data used in fitting is large. Therefore, there is tradeoff of choosing the value of M_{lower}.

The second parameter being tested is the number of M's or fitting points. The smallest value of M is fixed at 0.09 which is an optimal value as indicated by Fig. 3.10. The number of M's is scanned from 3 to 100, and the corresponding standard errors for estimating $\Pr(SNM < M_0)$ are plotted in Fig. 3.11. We can clearly observe that beyond a certain point, more data will not contribute to error reduction.

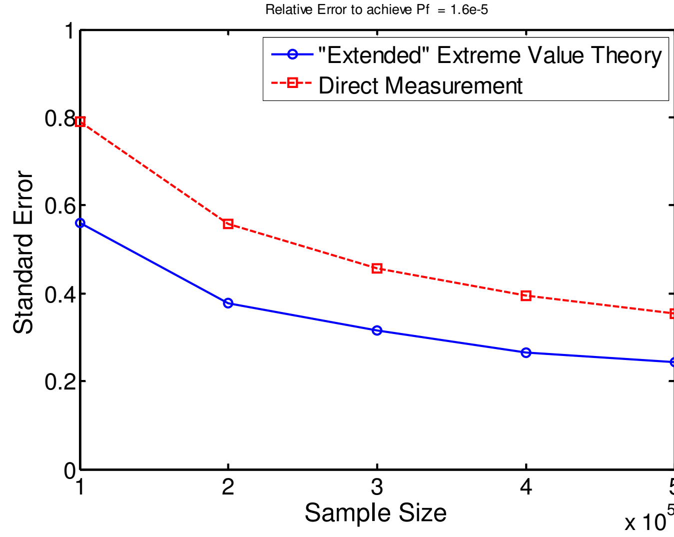

Fig. 3.12: Comparison of standard error as a function of sample size given by extended EVT and direct measurement.

We compare the efficiency of extended EVT method with direct measurement by looking at the standard error that can be achieved with certain sample size. Direct measurement refers to the methodology where one counts the number of failures out of a large sample, and it is essentially the Monte Carlo method. The results are shown in Fig. 3.12. For a given sample size, the extended EVT method can reduce the standard error by up to 50%. For a given standard error, the extended EVT method can shrink the sample size by 40%.

In summary, the extrapolation method builds upon the traditional Monte Carlo methods by imposing certain smoothness on the probability function. The method, more efficient than Monte Carlo as shown in the last paragraph, overlooks information that can be obtained from exploring the process parameter space. Therefore the improvement in efficiency is limited. A closer look at the function shape in the process parameter space is the topic of the next section.

3.3 Binary tree-based method

As stated in section 2.1, the Monte Carlo method wastes too many simulation runs away from the failure point, leading to its inefficiency in estimating the small failure probability. Ideally, one should not spend too much computational resource in the region that is determined to be either success or failure; instead one should focus on the boundary between success and failure.

In statistical learning theory, tree-based algorithms have been proven a successful technique for classification and regression [11]. By segmenting the feature space into smaller but pseudo-uniform regions, one can extract useful information and end up with a useful model. Similarly, we would expect the same method can quickly discard regions that are not important, and focus on the important boundary regions. The purpose of this section is to test the binary tree-based method in the context of failure probability estimation. We shall use a generic performance function $f(\xi)$ in this section in place of the static noise margin SNM. The failure is defined as $f(\xi) > M$, where M is the threshold.

3.3.1 Method

The first step of applying the binary tree-based method is to transform the original ξ space to ξ' space, such that every entry of ξ' is a uniformly-distributed random variable, and they are independent of each other. The transformation is possible if we rotate and scale the original ξ space properly [12]. In the newly transformed space, ξ' can take value from a high dimensional unit box of which the volume is 1. Within this box, there is a region or regions that correspond to the failure event, and the volume of that failure region is the failure probability we are interested in. For instance, in Fig. 3.13, the entire space is indicated by the large square. The failure region is the part to the top right of the curve. The calculation of the failure probability amounts to calculating the volume of the upper-right region of the box as defined by the curve.

The binary tree-based method starts with the entire square, splits it in half, and judges among the resulting split boxes whether they are worth further splitting. If the box is found contained in either the success or the failure region, it will be considered inactive and left un-processed. If it is not, the box will be split. In the end, the failure region will be approximated by a collection of boxes, among which a large proportion is devoted to mimic the boundary shape. This process is analogous to a binary tree, where the final boxes are the leaves of the tree. Therefore, the method is called the binary tree-based method.

It is understood that the complexity of the boundary grows exponentially with the dimension. But the algorithmic complexity of the tree-based method can be logarithmic. So we hope that logarithmic complexity of the tree algorithm can compensate for the exponential complexity due to dimensionality of the yield estimation problem.

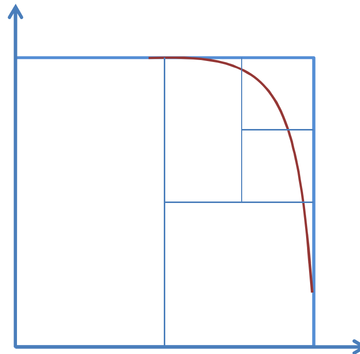

Fig. 3.13: Illustration of the binary tree-based method.

The complete flow of the binary tree-based method is summarized in Fig. 3.14. For every box, we assign a flag “splitornot” indicating the smallest number of splits that must be carried out before the box becomes inactive. The default value is 2, meaning every box must split at least twice to avoid mistakes due to lack of resolution. We start with a candidate box to split, evaluate the gradient direction and split along the Cartesian coordinate giving rise to the largest change in function value $f(\xi')$. The resulting 2 new boxes will be evaluated as described in the next paragraph. The splitting procedure updates the estimation of the failure probability. If the fractional change compared to the estimation before splitting is larger than a parameter Tol, then we reset a flag “splitornot” for the new boxes to default; if not, the newly generated boxes are considered one step

closer to being inactive by setting the flag to the value of its parent box minus one. Eventually, either all boxes are labeled inactive, or the fractional change is less than Tol, and the algorithm is terminated.

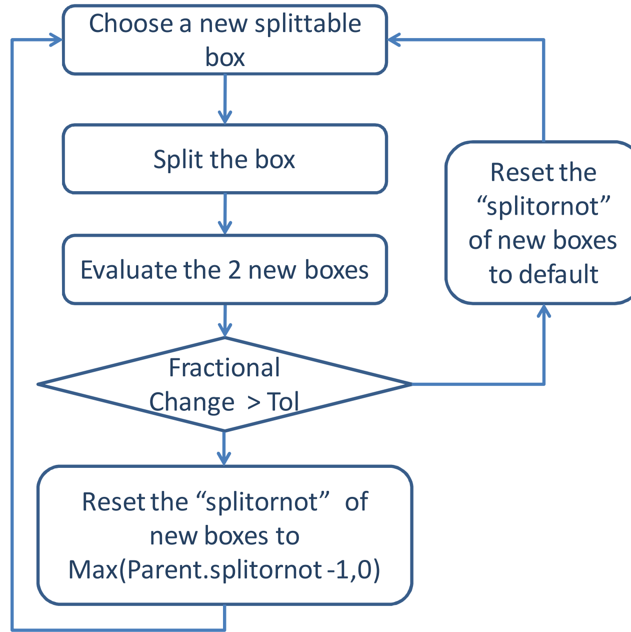

Fig. 3.14: Flow diagram of the tree-based algorithm.

A key step is the process used to evaluate the box. The evaluation procedure is shown in Fig. 3.15. First, the function $f(\xi')$ is evaluated at extreme points as shown in the left panel of Fig. 3.15. We do not pick the corner points of the box to avoid the exponential explosion of the number of corner points in high dimension. Once the function values are calculated, a linear model is fitted using least squares

$$f(\xi') = a^T \xi' + b, \tag{3.23}$$

where a and b are fitting parameters. Next, we solve the following two optimization problems

$$\begin{aligned} f_1 &= \min_{\xi' \in Box} f(\xi') \\ f_2 &= \max_{\xi' \in Box} f(\xi'). \end{aligned} \tag{3.24}$$

Both the objective function and the constraints in the above two optimization problems are linear functions; so they are recognized as linear programming (LP) problems, and we can solve it very quickly using off-the-shelf algorithms [13]. Using the function values f_1 and f_2, a judgment is made about whether the box is contained or not in either the success or failure region. If it is contained, we simply calculate the volume of the box and update the failure probability estimation. If it is not, we estimate the failure volume of this box based on a linear approximation:

$$V_f(Box) = \frac{f_2 - M}{f_2 - f_1} Vol(Box), \tag{3.25}$$

to update the total failure probability estimation.

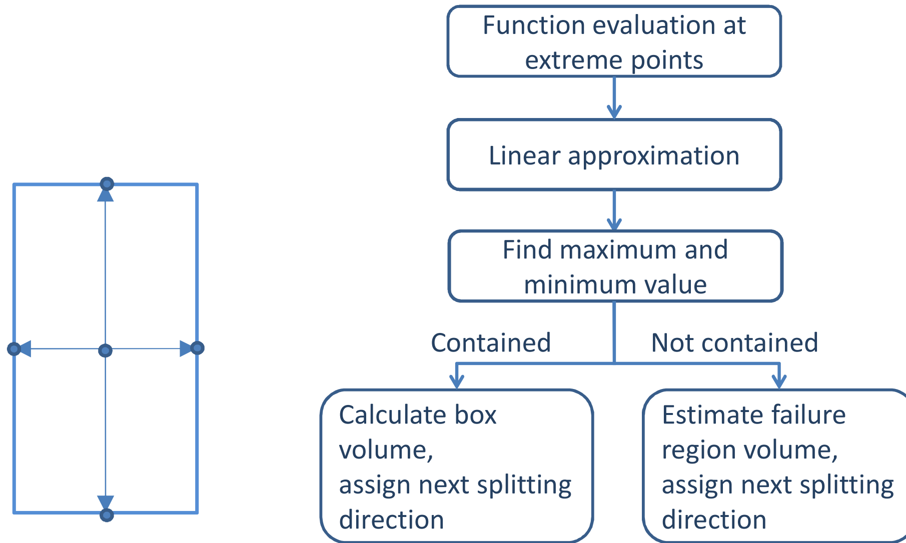

Fig. 3.15: Extreme points of a box and the flow diagram of evaluating the box.

3.3.2 Results and Discussion

We test the above method on a simple two-dimensional function

$$f(x, y) = 2x^2 + y^2 + 0.5x + 0.5y. \tag{3.26}$$

The threshold M is set to be 3.5. The parameter Tol is set to 0.02. The final box layout and the P_f estimate-step trajectory are shown in Fig. 3.16. The P_f estimate given by binary tree-based method is 0.0124, and the number of function evaluations is 230. By contrast, the P_f estimate by Monte Carlo with the same accuracy is 0.0120, and it needs 204531 function evaluations to reach the standard error of 0.02.

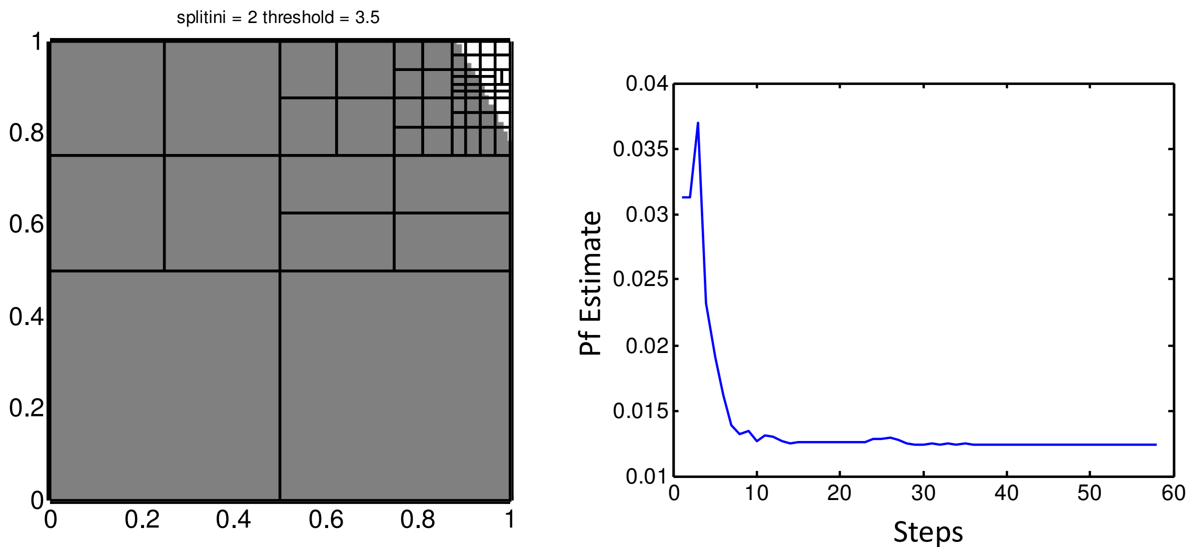

Fig. 3.16: Box splitting result (left) and P_f estimate trajectory (right).

To see how the method scales with dimensionality, we use two test functions. The first test function is a linear function of the form

$$f(\xi') = \alpha^T \xi' + \beta, \tag{3.27}$$

where the coefficients α and β are randomly chosen. The threshold M is set to a value such that the target P_f is around 0.1. The fractional change tolerance Tol equals 0.02. The number of function evaluations in order to achieve the desired accuracy ($Tol = 0.02$) is plotted in Fig. 3.17. What is also shown is the number of function evaluations for the Monte Carlo method to achieve standard error of 0.02. We can observe in Fig. 3.17 that the binary tree-based method is much more efficient than Monte Carlo on low-dimensional problems. But its computational cost grows quickly as dimension grows. When the dimensionality is beyond 6, Monte Carlo is already a faster method than the binary tree-based method.

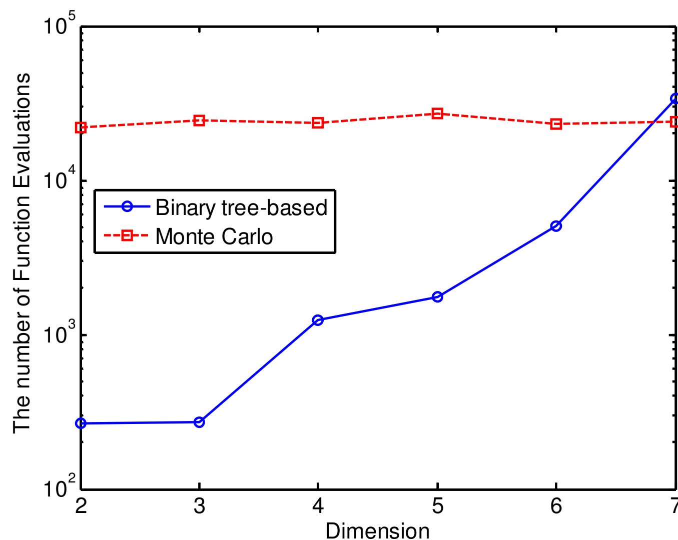

Fig. 3.17: The number of function evaluations as a function of dimension for linear test function.

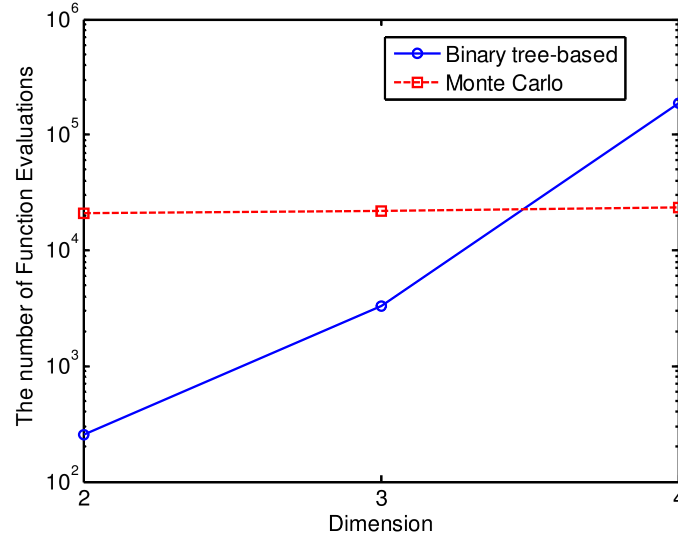

Fig. 3.18: The number of function evaluations as a function of dimension for quadratic test function.

The second test function is a quadratic function of the form

$$f(\xi') = \xi'^T A \xi' + b^T \xi' + c, \tag{3.28}$$

where the coefficients A, b and c are randomly chosen. Similar to the linear test case, the threshold M is set to a value such that the target P_f is around 0.1. The fractional change tolerance Tol equals 0.02. The number of function evaluations in order to achieve the desired accuracy for both binary tree-based method and Monte Carlo method is plotted in Fig. 3.18. For a slightly more complicated function as that in Eq. (3.28), the efficiency of the binary tree-based method deteriorates quickly as the dimensionality increases, whereas the necessary number of function evaluations for Monte Carlo to achieve the same accuracy remains almost constant.

In summary, the binary tree-based method has an advantage in low dimensional problems. However, it suffers the curse of dimensionality in high dimensions. On the contrary, the Monte Carlo method is plagued by small probability simulation, but is immune to dimension. It would be better to combine the deterministic search in the random parameter space as found in binary tree-based method with stochastic simulation. This is the topic of the next section.

3.4 Partial least squares-preconditioned importance sampling method

The technique of importance sampling [10] is a perfect example of combining the deterministic search and stochastic simulation. It has been adopted by several authors [4], [5], [14] to estimate failure probability. Most of the popular importance sampling implementations begin with a preconditioning step to find a shift point, construct around the shift point a shifted version of the original probability density function, and perform Monte Carlo simulation based on the new density function. The simulation result is translated back to the original distribution via the formulae as stated in importance

sampling theory. The odds of success rely on the quality of the shift point. Existing methods use random search or stratified search in the parameter space to find the minimal-norm point [4], [5]. Provided the search space is extremely "sparse" compared to the "point" that we hope to find, the preconditioning step in existing methods cannot guarantee the quality of the shift point. As shown in the following text, should the choice be bad, the resulting importance sampling can take a long time to converge. The remedy for the existing framework is to increase the sample size in the preconditioning step. However, this will almost fail for sure as dimension increases since the required sample size will grow exponentially.

Given the problems faced by the existing approach, a better preconditioning step is needed to aid the importance sampling in yield estimation. We propose to make use of partial least square (PLS) regression [15] to shrink the search space. We collapse the search space onto the line determined by the rotating vector in PLS regression. Then a systematic line search is used to find a boundary point. That point is used as the shift point to construct the biased density function and carry out the subsequent importance sampling simulation. Since we use PLS in the preconditioning step, the method is called the PLS-preconditioned importance sampling. It will be illustrated through example that the proposed method is much more efficient and stable than existing approaches.

In this section, we will first introduce the basic idea of importance sampling and point out the crucial step that awaits systematic solution. We then lay out the general settings of the seemingly irrelevant PLS regression. Finally the two concepts are combined to introduce the PLS-preconditioned importance sampling. In describing the method, we will use general notation, but will refer them to the quantities defined in section 3.1.

3.4.1 Importance Sampling

Suppose that the random variable is $\xi \in \mathbb{R}^N$, of which the probability density function is $g(\xi)$. In our example, ξ is a (6×1) vector denoting the threshold voltage fluctuation of every transistor within the SRAM cell. The failure event is determined by a circuit performance metric, a scalar function of ξ: $f(\xi)$. In our example, $f(\xi) = SNM1(\xi)$. If the value of $f(\xi)$ falls into region $\mathcal{F}$, it is considered a failure. Our objective is to evaluate $P_f = \Pr(\xi: f(\xi) \in \mathcal{F})$. In the SRAM example, this quantity is defined in Eq. (3.6).

We have discussed the methods of Monte Carlo and importance sampling in Chapter 2. For the purpose of illustrating the principle of the proposed method, we re-state here some of the key features. In Monte Carlo, a random sequence of length R, $\{\xi_r\}_{r=1}^{R}$ is generated according to $g(\xi)$, and $f(\xi)$ is evaluated for all ξ_r's. P_f is estimated as

$$\hat{P}_f = \frac{1}{R}\sum_{r=1}^{R} 1(f(\xi_r) \in \mathcal{F}), \tag{3.29}$$

where $1(f(\xi) \in \mathcal{F})$ is the indicator function of the failure event. The variance of the estimate is

$$\mathrm{Var}(\hat{P}_f) = \frac{1}{R}\left(P_f - P_f^2\right). \tag{3.30}$$

In order to achieve acceptable accuracy, the standard deviation of the estimate must be small compared to the estimate itself. If the standard deviation is required to be less than kP_f, then the required number of runs R is

$$R = \frac{1}{k^2}\frac{1 - P_f}{P_f}. \tag{3.31}$$

It is evident from Eq. (3.31) that the number of runs to achieve certain accuracy will increase dramatically as P_f becomes close to zero. The reason is that most of the runs are "wasted" in the region where we are not interested in.

Importance sampling [16] as a variance reduction technique was invented to circumvent this difficulty by biasing the original distribution such that there are sufficient number of trials falling into the important region, which is the region close to the success/failure boundary, defined by

$$\{\xi : f(\xi) = F\}. \tag{3.32}$$

In the SRAM example, the boundary is $\{\xi : SNM1(\xi) = SNM_{min}\}$. In importance sampling, the random sequence $\{\xi_r\}_{r=1}^{R}$ is generated according to the biased density function $h(\xi)$, and the failure probability is estimated by

$$\hat{P}_f = \frac{1}{R}\sum_{r=1}^{R} w(\xi_r) 1(f(\xi_r) \in \mathcal{F}), \tag{3.33}$$

where $w(\xi_r) = g(\xi_r)/h(\xi_r)$. The variance of the estimate is given by

$$\mathrm{Var}(\hat{P}_f) = \frac{1}{R^2}\left(\sum_{r=1}^{R}\left(w^2(\xi_r) 1(f(\xi_r) \in \mathcal{F})\right) - RP_f^2\right). \tag{3.34}$$

If we want to achieve the same accuracy as that in Eq. (3.31), we can stop the simulation when the following criterion is met:

$$\frac{\sqrt{\mathrm{Var}(\hat{P}_f)}}{\hat{P}_f} \leq k. \tag{3.35}$$

The introduction of the biased distribution $h(\xi)$ is a double-edge sword. Chosen wisely, it can reduce the variance given by Eq. (3.34), thus reducing the number of runs required by Eq. (3.35). On the other hand, if the choice is bad, the performance of importance sampling can be worse than plain Monte Carlo.

A good biased distribution $h(\xi)$ can be constructed by shifting the center of the original density function to the success/failure boundary defined by Eq. (3.32) such that the sampling points according to the biased distribution can provide ample information about

the import region. This operation is hard to implement in high dimensional space because the boundary is a complicated high dimensional surface, and finding a point on that surface as the center of the new biased distribution requires careful investigation.

A popular norm-minimization method proposed in [4] is to perform uniform random search in a reasonably large space, screen out those points that give rise to failure events, and find among them the point with the smallest Euclidean norm. The norm-minimization method is intended to find the failure point that is the most likely to fail. But relying purely on haphazard search can easily miss the intended point in high dimensional space, and large amount of random simulations are needed to determine the shift point.

We propose an alternative method to identify the shift point, aided by PLS regression as will be explained in the next two subsections.

3.4.2 Partial Least Squares (PLS) regression

PLS regression [15] is a method for modeling relations between two sets of variables. In practice, variables often exhibit correlation between each other because they depend on certain common latent factors. Projecting the data onto these latent structures reveals important factors, reduces the dimensionality of the problem, and enhances robustness. Principal component analysis (PCA) [17] is a well-established method to resolve internal correlation among explanatory variables. However, this overlooks the external relation between explanatory and response variables. This is where PLS comes into play: it projects both explanatory and response variables onto a few leading factors (score vectors), such that the relation between explanatory and response variable can be best explained by the relation between the first few score vectors.

The general setting of PLS regression consists of two data sets. One set of data, denoted as $x \in \mathbb{R}^N$, is the explanatory variable vector; the other set $y \in \mathbb{R}^D$ is the response variable vector. In our analysis, each value of the explanatory variable vector x is an instantiation of the random variable vector ξ. The response variable y takes value of the circuit performance metric $f(x)$, therefore in this specific case it is a scalar with $D = 1$. Suppose that we have n data samples from each data set, we can then organize the explanatory variables into matrix $\mathbf{X} \in \mathbb{R}^{n \times N}$, and organize the response variables into matrix $\mathbf{Y} \in \mathbb{R}^{n \times D}$

$$\mathbf{X} = \begin{bmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{bmatrix} \text{ and } \mathbf{Y} = \begin{bmatrix} y_1^T \\ y_2^T \\ \vdots \\ y_n^T \end{bmatrix}. \tag{3.36}$$

For the convenience of analysis, the mean value of each variable is subtracted from both matrices. PLS decomposes the zero-mean matrices $\mathbf{X}$ and $\mathbf{Y}$ as

$$\begin{aligned} \mathbf{X} &= \mathbf{T}\mathbf{P}^T + \mathbf{E} \\ \mathbf{Y} &= \mathbf{U}\mathbf{Q}^T + \mathbf{F} \end{aligned} \tag{3.37}$$

where $\mathbf{T} = (t_1, t_2, \dots t_p)$ and $\mathbf{U} = (u_1, u_2, \dots u_p)$ are $(n \times p)$ matrices of p score vectors, $\mathbf{P} \in \mathbb{R}^{N \times p}$ and $\mathbf{Q} \in \mathbb{R}^{D \times p}$ are matrices of loading vectors, and $\mathbf{E}$ and $\mathbf{F}$ are residual matrices. Ordinary least squares regression is then carried out in order to determine the score

vectors t_i and u_i. Regression on score vectors is better than that on the original data matrices in the sense that score vectors t_i and u_i are the choice that has the largest covariance among the linear combinations of columns of **X** and **Y**. In fact, in most implementations of PLS (e.g. NIPALS [18] and SIMPLS [19]), the score vectors are determined by iteratively updating the rotating vectors $r \in \{v|v \in \mathbb{R}^N, \|v\| = 1\}$ and $s \in \{v|v \in \mathbb{R}^D, \|v\| = 1\}$, such that

$$|cov(t,u)| = |cov(\mathbf{X}r, \mathbf{Y}s)| \tag{3.38}$$

is maximized. Once the score vectors t and u are determined, the loading vectors are determined by regressing the data matrices against them:

$$p = \mathbf{X}^T t/(t^T t) \text{ and } q = \mathbf{Y}^T u/(u^T u). \tag{3.39}$$

The overall procedure is iterative and it is repeated after the original data matrices **X** and **Y** are updated by *deflating* the data matrices:

$$\mathbf{X}' = \mathbf{X} - tp^T \text{ and } \mathbf{Y}' = \mathbf{Y} - uq^T. \tag{3.40}$$

The remaining data matrices **X'** and **Y'** are used to find the next score vectors until certain convergence criteria are met.

In our analysis, we will use only the first score vectors and denote them as t, u without subscript. Among all linear combinations of the column vectors in **X**, t gives the best explanation of the variance in **Y**.

3.4.3 PLS-preconditioned importance sampling

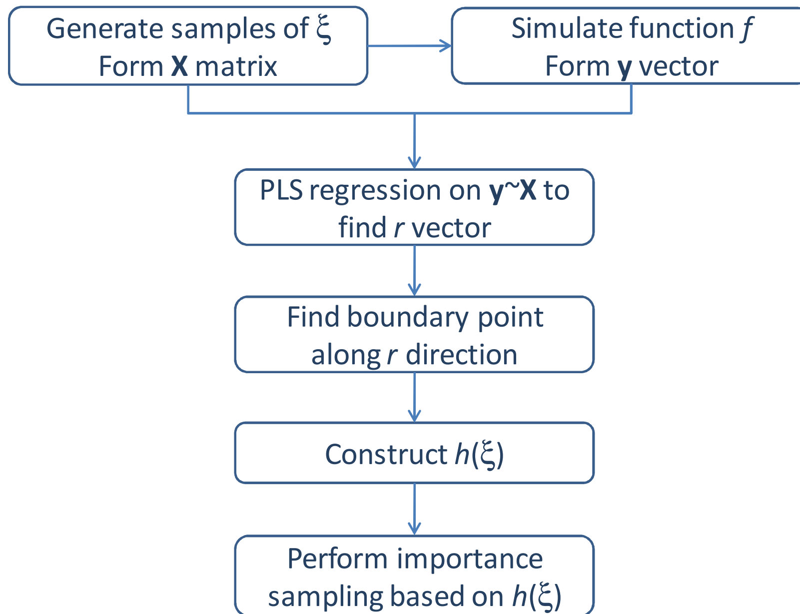

Fig. 3.19: Flow chart of PLS –preconditioned importance sampling.

Once we completed the PLS regression, we have found the rotating vector r that combines the columns of $\mathbf{X}$ matrix to obtain leading score vector t. The vector r projects x into the direction parallel to r. That direction is the best direction in x space to explain the covariance between x and y. *We postulate that a good choice of shift point for importance sampling lies on that direction, and it will be justified by simulation results.*

Since we have fixed the direction, we can perform line search along it to find the boundary point. This can be done by using any line search algorithm. Upon obtaining the boundary point on the r direction, we center the biased distribution $h(\xi)$ on that point and perform importance sampling simulation.

In our simulation we use the Newton-Raphson method in line search. Suppose the distance along direction r is represented by scalar s, then s is updated in every iteration according to

$$s_{(n+1)} = s_{(n)} - \frac{f\left(s_{(n)}\frac{r}{\|r\|}\right) - F}{\frac{\partial f\left(s_{(n)}\frac{r}{\|r\|}\right)}{\partial s_{(n)}}}. \tag{3.41}$$

where F is the parameter to define boundary in Eq. (3.32). Notice that in doing the line search to find the boundary point, we do not intend to solve equation $f\left(s_{(n)}r/\|r\|\right) = F$ with high accuracy, because we are not interested in the solution to that equation itself and the solution is only used as a base point for subsequent simulation. A modest accuracy requirement can be employed as the stopping criterion. In our simulation, we use

$$\left|f\left(s_{(n)}\frac{r}{\|r\|}\right) - F\right| \leq 0.02F. \tag{3.42}$$

The line search typically reaches the convergence criterion in less than 5 steps.

Following the above discussion, the PLS-preconditioned importance sampling is summarized in Fig. 3.19.

3.4.4 Results and Discussion

We compare our method with the norm-minimization method proposed in Ref [4] by calculating the failure probability defined in Eq. (3.6): $\Pr(\xi: SNM1(\xi) < SNM_{min})$. The accuracy k as defined in Eq. (3.35) is 0.1.

In a norm-minimization method, 3000 uniform random sampling points are generated on $[-6a_g, 6a_g]$ for every ξ_i. According to Eq. (3.1), the standard deviation σ of ξ_i is $\sqrt{2}a_g$; so the interval is around $[-4\sigma, 4\sigma]$ for every ξ_i. For all sampling points, simulation is performed in order to find SNM, and all failure points are sorted out based on simulation results. The Euclidean norm should be used with modification since the underlying random variables ξ_i are not independent. Suppose the covariance matrix of ξ is Σ, a more appropriate choice of norm is

$$\|\xi\|_\Sigma = \sqrt{\xi^T \Sigma^{-1} \xi} \tag{3.43}$$

Thus, among the failure points, the one with the smallest $\|\cdot\|_\Sigma$ norm is picked as the shift point for importance sampling. Since both the search for the point with minimum norm and importance sampling itself are stochastic simulations, we repeat the entire procedure (norm minimization plus the subsequent importance sampling) 50 times in order to have a meaningful comparison with our method. As will be shown in a moment, sometimes this method can result in very bad choice of the shift point, and the resulting importance sampling run number to achieve the accuracy $k = 0.1$ is prohibitively large. Therefore, we manually stop the importance sampling process whenever the run number exceeds 1×10^5.

In PLS-preconditioned importance sampling, we first draw 500 sampling points from the original distribution $g(\xi)$. These points are used in PLS regression to determine the rotation vector r. Once the rotation direction is determined, a line-search is performed to find the boundary point as described in section 3.4.3. The line-search normally takes less than 5 steps to converge. After the boundary point along the r direction is determined, we use that point as the shift point to perform importance sampling simulation. As in norm-minimization approach, the whole procedure is also random. Therefore, it makes sense to repeat it 50 times as well.

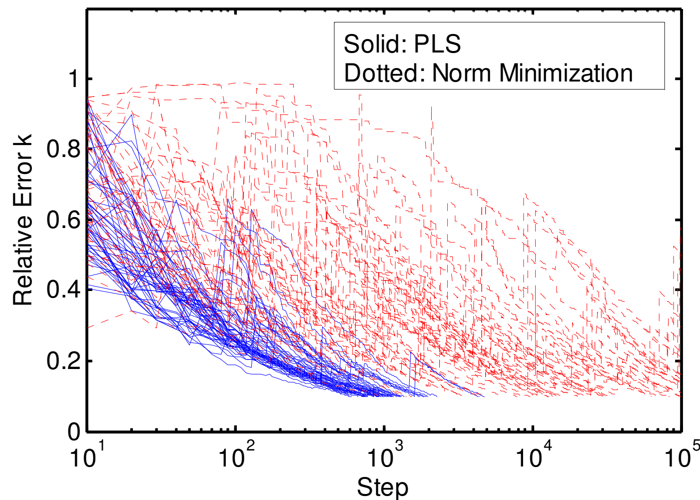

Fig. 3.20: Relative error trajectories in importance sampling. 50 experiments using PLS-preconditioned method are shown in solid lines. 50 experiments using norm-minimization method are shown in dotted lines.

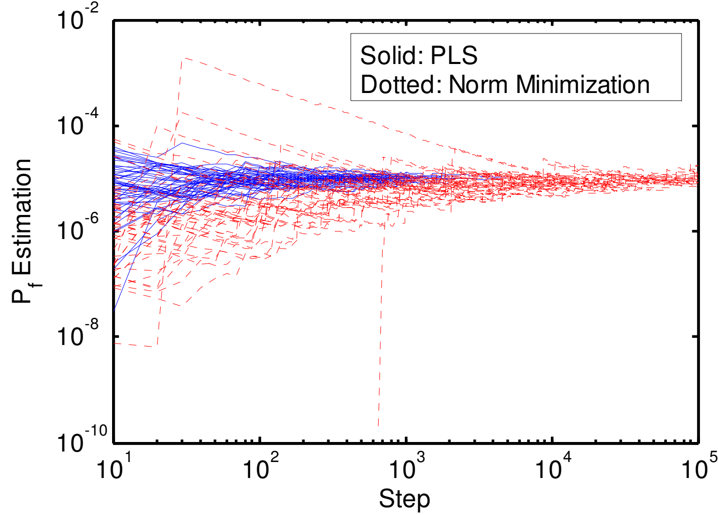

Fig. 3.21: Failure probability estimation trajectories in importance sampling. 50 experiments using PLS-preconditioned method are shown in solid lines. 50 experiments using norm-minimization method are shown in dotted lines.

The relative error k as defined in Eq. (3.35) recorded at every step in importance sampling simulation are shown in Fig. 3.20. The solid lines represent the trajectories given by PLS-preconditioned importance sampling, and the dotted lines are given by the norm-minimization method. The number of simulations in order to achieve $k = 0.1$ in PLS-preconditioned importance sampling is around 10^3, whereas that of norm-minimization method varies significantly: some of them are comparable or even better than PLS-preconditioned method, but some fail to reach the target accuracy in less than 10^5 runs.

The trajectories of P_f estimation (Fig. 3.21) show that during importance sampling simulation, the PLS-preconditioned method oscillates in a smaller range, and converges to the correct failure probability ($\sim 10^{-5}$) quickly. The quick convergence benefits from the high quality of the shift point picked during the preconditioning process. By contrast, the shift point given by norm-minimization method is chosen in a purely random fashion, thus failing to guarantee a fast convergence.

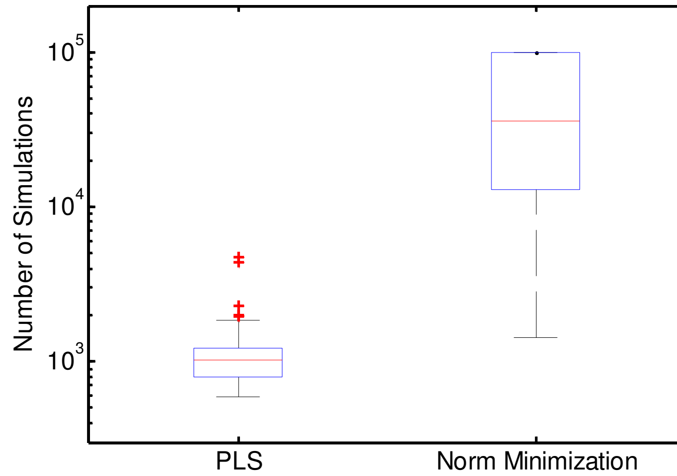

Fig. 3.22: Box plots of the number of simulations to reach $k = 0.1$. The plots use the data from 50 experiments for both methods. The number for norm-minimization method is not shown because the simulation is manually halted if it fails to converge within 10^5 runs.

The comparison is further summarized in Fig. 3.22 and Fig. 3.23. The box-plots in Fig. 3.22 compare the number of simulations needed to achieve the same accuracy $k = 0.1$, and Fig. 3.23 compares the P_f estimations given by the two approaches. The two methods give essentially the same estimation of failure probability. But for the simulation runs to reach convergence, the number is much smaller and the spread is narrower in PLS-preconditioned method. Notice that the simulations runs in the preconditioning step for the PLS-preconditioned method (500 in the example), and the simulation runs in norm-minimizing step for norm-minimization step (3000 in the example) are excluded. If included, they can further enlarge the benefit gained by PLS-preconditioned method. It is true that one could enhance the quality of norm-minimizing step by increasing the number of samples used in the preconditioning step. Even though this approach can work to some extend in this example ($\xi \in \mathbb{R}^6$), it will be troublesome if we are dealing with a space of which the dimension is much higher than 6, where it becomes exponentially harder to identify a good shift point as the dimension increases. The PLS-based method will not suffer as much in high dimensional space. In fact, in other fields [14], PLS is widely used as a dimension-reduction technique to screen out import factors from a large population of variables at the cost of very few experimental runs.

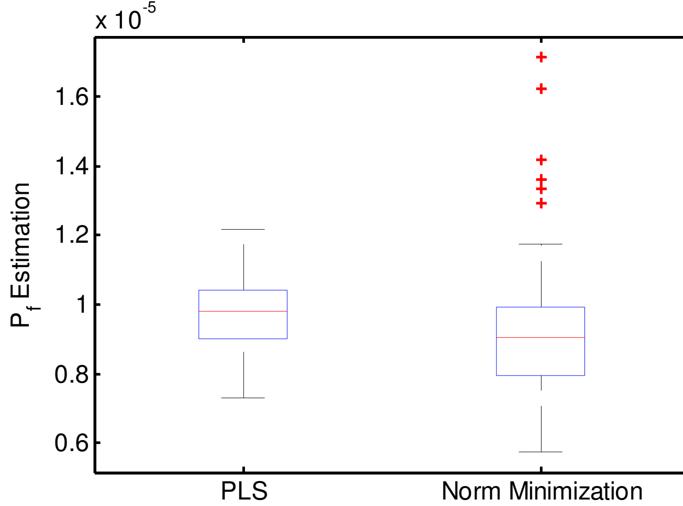

Fig. 3.23: Box plots of P_f estimation. The plots use the data from 50 experiments for both methods. Both methods give roughly the same estimation of probability of failure$P_f \sim 10^{-5}$.

Finally, we note that to estimate failure probability as slow as 10^{-5} with 10% standard error, it takes Monte Carlo 10^7 simulation runs according to Eq. (3.31). The partial least squares-preconditioned importance sampling method exhibits 10^4 speed up compared to plain Monte Carlo.

We devote the last part of this section to further understanding why PLS regression can help pick a good shift point.

Suppose the random data ξ exhibits itself as data cloud shown in Fig. 3.24(a). The shape of the cloud is determined by the level sets of probability density function $g(\xi)$. The level sets of the scalar function $f(\xi)$ are plotted as solid lines in ξ space. Thus the points on the same line have identical function values. Suppose the center line represents the value that distinguishes success and failure $f(\xi) = F$. Therefore any point on this line is a boundary point. According to the large deviation theory [20], when a rare event happens, it happens in the most likely manner. Translated to the geometry exhibited in Fig. 3.24(a), it means the point that is the most likely to fail is the tangential point of the scalar function level set with respect to the level set of the probability density function. That is the point where we should center the biased density function $h(\xi)$ for importance sampling. This choice of ideal shift point is the same as in norm-minimization method. However, unlike the norm-minimization method which scouts the space randomly, PLS regression can find this point in a systematic manner by carrying out line search in the rotating direction.

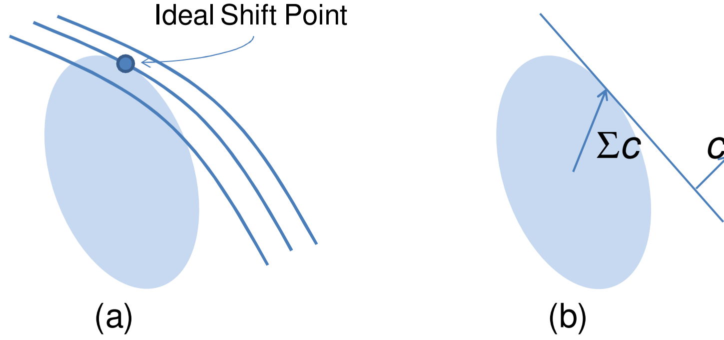

Fig. 3.24: Illustration of ideal shift point selection in ξ space. (a) The tangential point between function level set and density function level set is the ideal shift point. (b) For data with covariance matrix Σ and linear function with coefficient c, the tangential point lies in the direction parallel to vector Σc.

We give now a mathematical proof of our claim that the tangential point mentioned in the last paragraph lies on the r direction given by PLS regression. We will prove in a simplified case where the performance metric is a linear function ξ as illustrated in Fig. 3.24(b). Though strict mathematical proof cannot be made for more general cases, this proof can give some insight, and we resort to simulation results in the last section as a practical justification in general situation.

Proposition: Suppose $f(\xi) = c^T\xi$ and $\xi \sim \mathcal{N}(0, \Sigma)$, we have hyper-planes $P_\beta = \{\xi : c^T\xi = \beta\}$ ellipsoidal surface $E_\alpha = \{\xi : \xi^T\Sigma^{-1}\xi = \alpha^2\}$; then the tangential point between P_β and E_α lies on the direction given by the r direction in PLS regression (Fig. 3.24(b)).

Proof: Following the setup of PLS regression, we have ξ's stacked in $\mathbf{X}$, and n function values stacked in $y \in \mathbb{R}^{n\times 1}$. Therefore we have $y \equiv \mathbf{X}c$ due to our assumption. As illustrated in section 3.4.2, the goal of PLS regression is to find the rotation vector r and s, such that the covariance between rotated response and explanatory variables is maximized. Since y is now a column vector instead of a matrix, the rotation on y does not have any rotating effect; thus it suffices to consider r only. The maximization criterion is formally stated as

$$\begin{aligned} r &= \arg\max_{\|r\|=1}\big(\mathrm{cov}(y, \mathbf{X}r)\big) \\ &= \arg\max_{\|r\|=1}(y^T\mathbf{X}r) \\ &= \arg\max_{\|r\|=1}(c^T\mathbf{X}^T\mathbf{X}r). \end{aligned} \tag{3.44}$$

The solution to the above problem is

$$r = \frac{\mathbf{X}^T\mathbf{X}c}{\|\mathbf{X}^T\mathbf{X}c\|} \tag{3.45}$$

Suppose without loss of generatlity that $\mathbf{X}$ is a zero-centered matrix, then the covariance matrix of ξ is $\Sigma \propto \mathbf{X}^T\mathbf{X}$. So the direction determined by r is parallel to Σc.

Now we will show that the tangential point is also on the direction parallel to Σc. The definition of tangency implies that the normal direction of ellipsoidal surface E_α is the same as that of the hyper-plane P_β. The normal direction of of any point ξ on E_α is

$$\nabla_\xi(\xi^T\Sigma^{-1}\xi - \alpha^2) = 2\Sigma^{-1}\xi. \tag{3.46}$$

This quantity should be proportional to c, therefore we have the following equation

$$2\Sigma^{-1}\xi = \gamma c, \tag{3.47}$$

where γ is a constant. Solving this can give the solution for the tangential point

$$\xi = \frac{\gamma}{2}\Sigma c, \tag{3.48}$$

showing that the tangential point is along the direction determined by Σc. This coincides with the r direction given by PLS regression. Q.E.D.

In reality, the function $f(\xi)$ is seldom a linear function of ξ. But PLS regression can still capture the interaction between $f(\xi)$ and the distribution of ξ. This is illustrated in our SRAM example as shown in Fig. 3.25. It shows the contour lines of $SNM1$ as a function of threshold voltage fluctuation. Only the $\Delta Vth1 - \Delta Vth2$ plane is shown here due to the visualization limitation. The dotted line shows the principal component direction, which corresponds to the major axis of the ellipsoid. The direction given by the r vector in PLS regression is shown in solid line. It is deflected away from the principal component direction toward the function gradient direction. Considering the above discussion, this is a better choice in terms of finding the shift point for importance sampling.

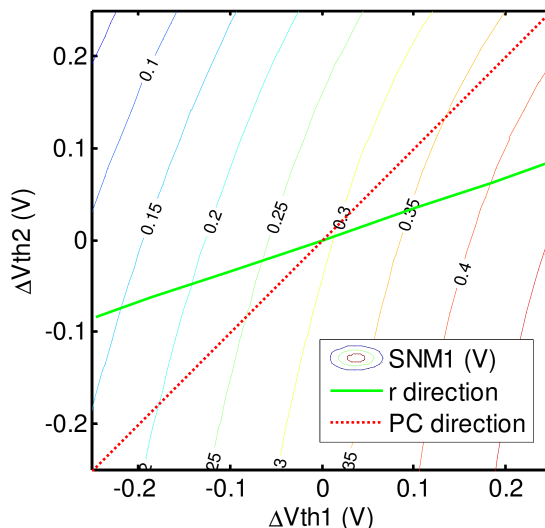

Fig. 3.25: Contours of $SNM1$ on $\Delta Vth1 - \Delta Vth2$ plane. The solid line is the direction given r vector. The dotted line represents the direction of the first principal component direction if one carries out PCA on the data.

3.5 Summary

Yield estimation plays an important role in yield-based statistical circuit analysis. Among various yield calculation methods, the Monte Carlo method is purely based upon stochastic simulation. By imposing additional smoothness on the failure probability, we have the extrapolation methods described in section 3.2. Compared to plain Monte Carlo, the extrapolation methods can reduce the sample size by a factor of 4. The improvement is not very significant compared to the methods introduced later in this chapter, but it has a unique advantage of being able to be used in measurements.

Binary tree-based method abandons the stochastic approach. It explores the geometry of the space defined by the random process variables. Though it is shown to be very effective in low dimensional problems, it suffers from the curse of dimensionality.

Combining both stochastic simulation and deterministic exploration, importance sampling inherits Monte Carlo's immunity to dimensionality, and overcomes the difficulty of Monte Carlo in rare event simulation by biasing the original distribution. Existing methods of obtaining the bias distribution relie on haphazard search in the parameter space, and the resulting performance can vary significantly from run to run. More often than not, the bad choice of shift point causes degrades the performance of importance sampling. We propose a PLS-preconditioned method to address this problem. The method starts with PLS regression to find the most important direction in parameter space, reducing the search space to a single line. Then a simple line search is done in this direction to find the boundary point. The biased distribution is constructed around that point, and is used in subsequent importance sampling simulation. We illustrate the effectiveness of the proposed method through an example of SRAM cell. It is shown that the PLS-preconditioned importance sampling is much more stable than existing method. In average, it is at least one order or magnitude faster than existing importance sampling techniques, and is 10^4 faster than Monte Carlo.

3.A The calculation of covariance matrix elements in Eq. (3.22)

Suppose we have a random sample $\{S_i\}_{i=1}^{N}$, and threshold values T_1 and T_2, from which two binomial random variables can be defined:

$$P_1 = \frac{1}{N}\sum_i 1(S_i \leq T_1)$$

$$P_2 = \frac{1}{N}\sum_i 1(S_i \leq T_2).$$

The mean values of these two random variables are $E(P_1) = \overline{P_1}$ and $E(P_2) = \overline{P_2}$. To calculate the covariance, one needs to calculate $E(P_1 P_2)$:

$$E(P_1P_2) = E\left(\frac{1}{N^2}\sum_i\sum_j 1(S_i \le T_1)1(S_j \le T_2)\right)$$
$$= E\left(\frac{1}{N^2}\sum_i 1(S_i \le T_1)1(S_i \le T_2) + \frac{1}{N^2}\sum_i\sum_{j\ne i} 1(S_i \le T_1)1(S_j \le T_2)\right)$$
$$= \frac{1}{N^2}\sum_i \min(\overline{P_1}, \overline{P_2}) + \frac{1}{N^2}\sum_i\sum_{i\ne j}\overline{P_1}\,\overline{P_2} = \frac{\min(\overline{P_1}, \overline{P_2})}{N} + \frac{N(N-1)\overline{P_1}\,\overline{P_2}}{N^2}$$
$$= \frac{\min(\overline{P_1}, \overline{P_2}) + (N-1)\overline{P_1}\,\overline{P_2}}{N}.$$

Therefore we can calculate the covariance between P_1 and P_2 by

$$Cov(P_1, P_2) = E(P_1P_2) - E(P_1)E(P_2) = \frac{\min(\overline{P_1}, \overline{P_2}) - \overline{P_1}\,\overline{P_2}}{N}.$$

Now, consider

$$y_i = \log P_i \approx \log \overline{P_i} + \frac{\Delta P_i}{\overline{P_i}}$$

So the error term $\epsilon_i = \frac{\Delta P_i}{\overline{P_i}}$, and the covariance between ϵ_i and ϵ_j is

$$Cov(\epsilon_i, \epsilon_j) = \frac{1}{\overline{P_i}\,\overline{P_j}} Cov(\Delta P_i, \Delta P_j) = \frac{\min(\overline{P_i}, \overline{P_j}) - \overline{P_i}\,\overline{P_j}}{N\overline{P_i}\,\overline{P_j}} \approx \frac{\min(\overline{P_i}, \overline{P_j})}{N\overline{P_i}\,\overline{P_j}}.$$

Compare this result with the variance of ϵ_i

$$Var(\epsilon_i) = \frac{\overline{P_i} - \overline{P_i}^2}{N\overline{P_i}^2} \approx \frac{1}{N\overline{P_i}},$$

We can see that they are comparable. An example of the covariance matrix elements given by 10 M points is shown in Fig. 3.26.

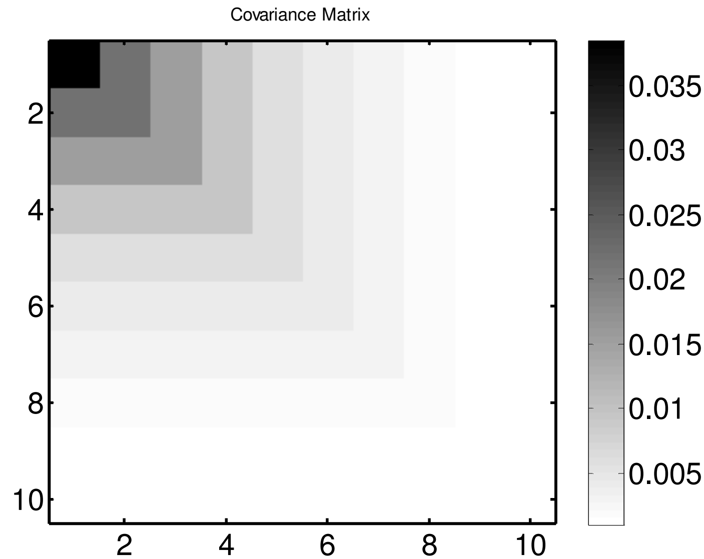

Fig. 3.26: Covariance matrix of failure probability estimations at 10 different values of M

3.6 References

[1] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm design exploration," IEEE International Symposium on Quality Electronic Design, 717-722 (2006)

[2] P. Friedberg, W. Cheung, G. Cheng, Q. Y. Tang and C.J. Spanos, "Modeling spatial gate length variation in the 0.2um to 1.15mm separation range," SPIE Vol. 6521, 652119 (2007)

[3] J. M. Rabaey, A. Chandrakasan and B. Nikolic, "Digital integrated circuits: a design perspective," second edition, Prentice Hall (2003)

[4] L. Dolecek, M. Qazi, D. Sha and A. Chandrakasan, "Breaking the simulation barrier: SRAM evaluation through norm minimization," ICCAD (2008)

[5] T. Date, S. Hagiwara, K. Masu and T. Sato, "Robust importance sampling for efficient SRAM yield analysis," IEEE International Symposium on Quality Electronic Design, 15 (2010)

[6] A. A. Balkema and L. De Haan, "Residual life time at great age," The Annals of Probability, vol. 2, no. 5, pp. 792-804 (1974)

[7] J. Pickands, "Statistical inference using extreme order statistics," The Annals of Statistics, Vol.3, 119-131 (1975)

[8] A. Singhee and R. A. Rutenbar, "Statistical blockade: a novel method for very fast Monte Carlo simulation of rare circuit events, and its application," Design Automation and Test in Europe (2007)

[9] A. Kumar, J. Rabaey and K. Ramchandran, "SRAM supply voltage scaling: a reliability perspective," IEEE International Symposium on Quality Electronic Design, 782 (2009)

[10] A. C. Davison, "Statistical models," Cambridge University Press, (2008)

[11] T. Hastie, R. Tibshirani and J. Friedman, "The elements of statistical learning: data mining, inference, and prediction," second edition, Springer (2009)

[12] C. Gu and J. Roychowdhury, "An efficient, fully nonlinear, variability-aware non-Monte-Carlo yield estimation procedure with applications to SRAM cells and ring oscillators," Asia and South Pacific Design Automation Conf., 754-761 (2008)

[13] D. G. Luenberger, "Linear and nonlinear programming," second edition, Addison-Wesley (1984)

[14] R. Kanj, R. Joshi and S. Nassif, "Mixture importance sampling and its application to the analysis of SRAM designs in the presence of rare failure events," DAC, 69-72 (2006)

[15] R. Rosipal and N. Kramer, "Overview and recent advances in partial least squares," Subspace, Latent Structure and Feature Selection, Springer, 34-51 (2006)

[16] R. Srinivasan, "Importance sampling: applications in communications and detection," Springer (2002)

[17] W. f. Massy, "Principal component regression in exploratory statistical research," Journal of the American Statistical Association, Vol 60, 234-256 (1965)

[18] H. Wold, "Estimation of principal components and related models by iterative least squres," in P. R. Krishnaiah (Editor), "Multivariate analysis," Academic Press, New York, 391-420 (1966)

[19] S. de Jong, "SIMPLS: an alternative approach to partial least squares regression," Chemometrics and Intelligent Laboratory Systems, 18, 251-263 (1993)

[20] J. A. Bucklew, "Large deviation techniques in decision, simulation, and estimation," Wiley (1990)

Chapter 4 Customized Corner Generation and Its Applications

In this chapter we propose several methods to address the two challenges in corner model extraction described in chapter 2: inclusion of the local variability and customized definition of the corner for specific performance. The first method, based on PLS algebraic computation, is able to find the customized corner model for digital logic circuit in an efficient manner, but it is less useful for analog circuits. For analog and other applications with high nonlinearity, we propose to use optimization-based methods. These methods are slower than their algebraic counterparts, and are more suitable for analog applications since they are generally small in scale. Next we describe two sample problems. They are the circuits that the proposed methods will be tested on. The methods are then introduced and the results are discussed at the end.

4.1 Example problems

4.1.1 Digital logic delay

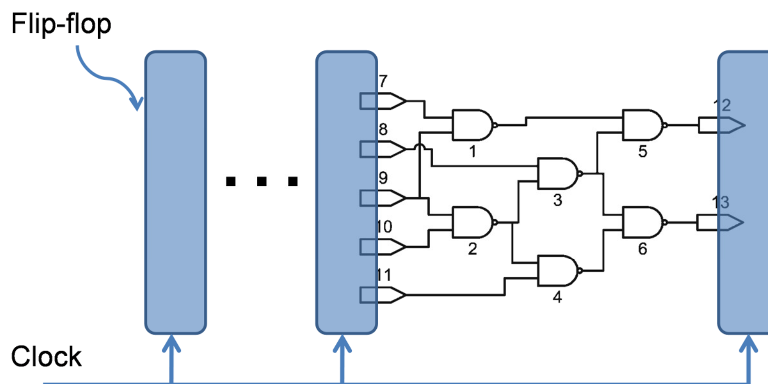

Figure 4.1: Digital logic blocks.

As shown in Fig. 4.1, the logic block can be viewed as a collection of gates. The logic circuit blocks are often placed between flip-flops, which are in turn controlled by clock signals. The data signal must arrive at the other end of the circuit block before the next clock cycle. Thus there are two important numbers associated with the digital logic block: one is the longest time it takes for the signal to traverse the block; it is called the critical delay $D_{critical}$; the other one is the maximum allowable delay determined by the clock frequency and the setup time of the flip flop; and we denote it as D_{max}. In order to for the circuit to work, the critical delay must be less than D_{max}. The performance of interest is the critical delay. We will not discuss the parameter D_{max} until chapter 5.

We describe the gate delay using the RC delay model [2], [3]: when a signal is about to propagate through a gate, it is essentially charging or discharging a capacitor through a non-linear resistor. We approximate this as a RC network composed of an equivalent linear resistor and an equivalent capacitor (Fig. 4.2). The equivalent resistor represents the driving strength of the gate; large gate size corresponds to small resistance. We will fix the gate sizing in our analysis in this chapter and the equivalent resistance of gate i is a constant:

$$R_i^{eq} = R_i. \tag{4.1}$$

The constant R_i depends on gate type, threshold voltage etc. The equivalent capacitance is the sum of all caps connected to this gate:

$$C_i^{eq} = C_i^{int} + \sum_{j \in FO(i)} C_j^{in}, \tag{4.2}$$

where C_i^{int} is intrinsic capacitance of gate i, C_j^{in} is the input capacitance of gate i, and $FO(i)$ denotes the indices of the gates that connect to the output of gate i, or the fan-out gate i. The delay of a single gate can be expressed by

$$D_i(x) = 0.69 R_i^{eq} C_i^{eq} \tag{4.3}$$

The constants that will be used in our example are summarized in Table 4.1. In Chapter 5 and 6, we will extend the model (4.1)-(4.3) to include the dependence on gate sizes.

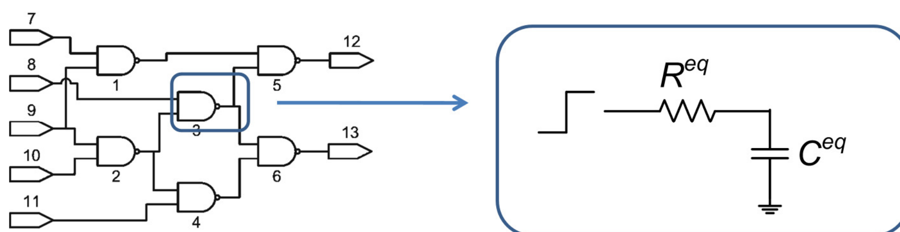

Figure 4.2: RC delay model.

Gate type	C^{in} (fF)	C^{int} (fF)	R (kΩ)	a
INV	3	3	0.48	3
NAND2	4	6	0.48	8
NOR2	5	6	0.48	10

Table 4.1: Circuit parameters [3]. The constant of the last column will be used in Chapters 5 and 6.

Because a digital logic block can be viewed as a directed acyclic graph, one can adopt either the path-based (depth-first) algorithm or the block-based (breadth-first) algorithm to find the critical delay of the circuit [4]. A complete enumeration of all paths for large-scale circuit is problematic since the number of paths grows exponentially with the size of the circuit. By contrast, a block-based algorithm can be used to calculate critical delay at linear computational cost.

We assume that the variability arising from manufacturing impacts the delay through threshold voltage fluctuation ξ. The threshold voltage of gate i is $V_{th,i} = V_{th,i}^{0} + \xi_i$, where $V_{th,i}^{0}$ is the nominal threshold voltage and is assumed to be 0.3 V. The unit-size resistance can be related to the threshold voltage by the α-power law [5]:

$$R_i \propto \left(V_{dd} - V_{th,i}\right)^{-\alpha}, \tag{4.4}$$

where V_{dd} is the supply voltage, typically around 1 V; α is a parameter that typically equals 1.3; $V_{th,i}$ is the threshold voltage of gate *i*. The variability in threshold voltage is described by the same hierarchical model as in (3.1):

$$\xi_i = a_g \epsilon_g + a_i \epsilon_i, \tag{4.5}$$

where a_g and a_i are the amplitudes of the global and local variations respectively, and ϵ_g, ϵ_i are independent standard normal random variables. In our example in this chapter, we assume $a_g = 50$ mV and $a_i = a_g$.

In summary, the critical delay is the performance metric of concern. We use a block-based method to evaluate critical delay of unit-sized logic circuit. The delay of every gate is given by Eq. (4.3). The variability is assumed to be the threshold voltage fluctuation and is modeled as Eq. (4.5). It is linked to the resistance in Eq. (4.3) by (4.4).

4.1.2 The pre-amplifier

The second example is a pre-amplifier circuit that has wide applications in signal processing, communication and power electronics [6]. As shown in Fig. 4.3, the circuit has 6 transistors (M1 – M6) and 4 load resistors. The output of the circuit (node 5) is driving a load capacitor of 10fF. The design parameters are shown on the schematics. The design is used to illustrate the methods presented here, and should be deemed as an optimal design example. The circuit is modeled using Hspice with 45nm PTM transistor compact model [7].

Similar to the SRAM problem examined in Chapter 3, the variability is assumed to come from threshold voltage fluctuation. We use a 6×1 vector ξ to characterize the threshold voltage change, of which every entry is the threshold change of each transistor, and is described using the same hierarchical model as in Eq. (4.5).

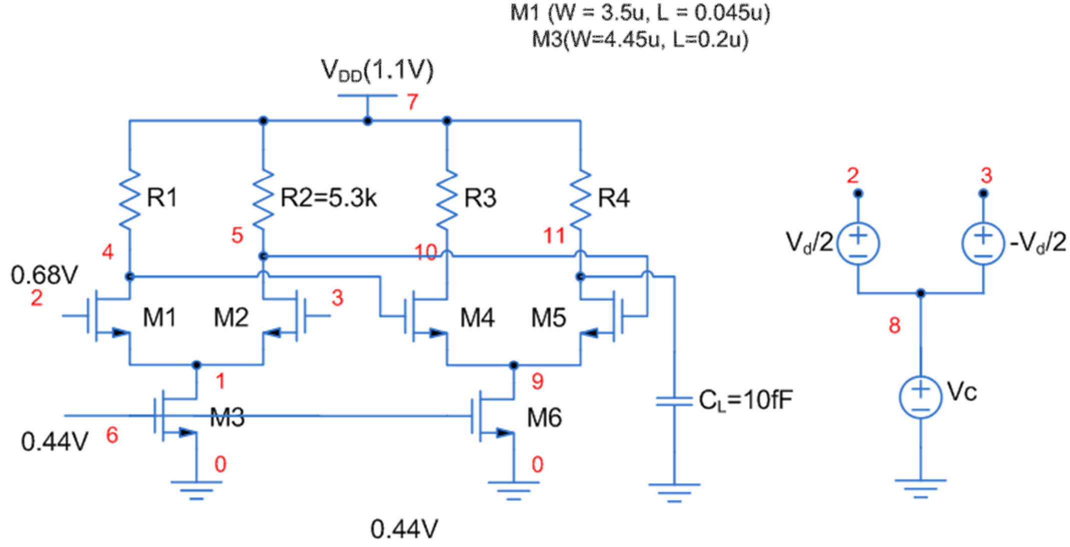

Fig. 4.3: The schematics of the pre-amplifier [6].

We will consider two performance metrics in our analysis. The first is the open loop differential gain. Under the small-signal model [6], if the differential input amplitude is V_{di}, and the differential output amplitude is V_{do}, then the open loop differential gain is defined as $A_d = V_{do}/V_{di}$, and expressed in units of dB:

$$A_d(dB) = 20\log_{10}(A_d). \tag{4.6}$$

A related quantity is the common mode gain. If the common mode input amplitude is V_{ci}, and the common mode output amplitude is V_{co}, then the common mode gain is defined as $A_c = V_{co}/V_{ci}$, and this quantity is also normally expressed in dB:

$$A_c(dB) = 20\log_{10}(A_c). \tag{4.7}$$

A good differential pre-amplifier should have a high open loop differential gain, and a low common mode gain. Therefore the second performance metric is the ratio of the differential gain to the common mode gain. It is called the common mode rejection ratio (CMRR), and it is given in dB as

$$CMRR(dB) = 20\log_{10}\left(\frac{A_d}{A_c}\right). \tag{4.8}$$

In summary, we use the same variability model Eq. (4.5) to describe the randomness arising from process variation. The quantities of interest are the open loop gain given in (4.6) and the common mode rejection ratio given in (4.8). These performances are evaluated through simulation using Hspice.

4.2 PLS-based algebraic method

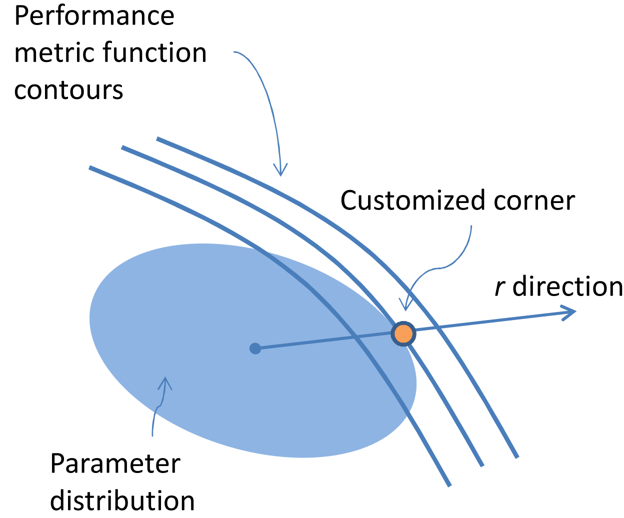

Fig. 4.4: Illustration of customized corner as the tangential point between the performance function contours and the variability ellipsoid.

The objective of customized corner estimation is to find a combination of the random process parameters that represent the worst performance situation at a certain yield level. For any given probability level α, there is a corresponding ellipsoid in the process parameter space (ξ-space) such that the probability of ξ being enveloped inside the ellipsoid is α. The volume of the ellipsoid is therefore dependent on α. The higher the probability level is, the larger the volume of the ellipsoid. Among all ξ points within a given ellipsoid, the performance metric will achieve its extremity at the ellipsoid surface, assuming that the delay is roughly a monotonic function of process parameters. More accurately, the extreme point is the tangential point between performance function contour and the ellipsoidal surface (Fig. 4.4). This point is the customized corner point that we are interested in.

We have shown in section 3.4.4 that partial least square regression (PLS) has the capability to identify the direction where the tangential point resides. So we propose to use PLS on a small set of randomly generated (or measured) samples in order to find the r direction as we did in Chapter 3, and then project the point in ξ-space onto the r direction and reduce the high dimensional problem to a one dimensional problem. If the original parameter ξ is a zero-centered multidimensional Gaussian random variable with a covariance matrix Σ, by projecting ξ to direction r, we have

$$\tilde{\xi} = r^T \xi \sim \mathcal{N}(0, r^T \Sigma r). \tag{4.9}$$

Thus, the α-quantile of the scalar $\tilde{\xi}$ can be obtained as

$$\tilde{\xi}^{(\alpha)} = \sqrt{r^T \Sigma r}\,\Phi^{-1}(\alpha). \tag{4.10}$$

The corresponding point in the original ξ-space is

$$\xi^{(\alpha)} = r\sqrt{r^T \Sigma r}\Phi^{-1}(\alpha) \tag{4.11}$$

where Φ is the cumulative distribution function for standard normal distribution. This point $\xi^{(\alpha)}$ is the customized corner point at the α-yield level, and the performance function value at $\xi^{(\alpha)}$

$$f^{(\alpha)} = f\left(\xi^{(\alpha)}\right). \tag{4.12}$$

should be the α-quantile of the distribution of that performance metric. Based on the quantile information, we can also construct the probability density function (PDF) of the performance:

$$PDF\left(f^{(\alpha)}\right) \propto \exp\left(-\frac{\left(\Phi^{-1}(\alpha)\right)^2}{2}\right). \tag{4.13}$$

Notice the PDF function given by Eq. (4.13) can be very different from a Gaussian distribution. It maps the value of the performance PDF function to the PDF of the underlying projected process parameter $\tilde{\xi}$ (Fig. 4.5). Even if the distribution of $\tilde{\xi}$ is Gaussian, that of f may not be.

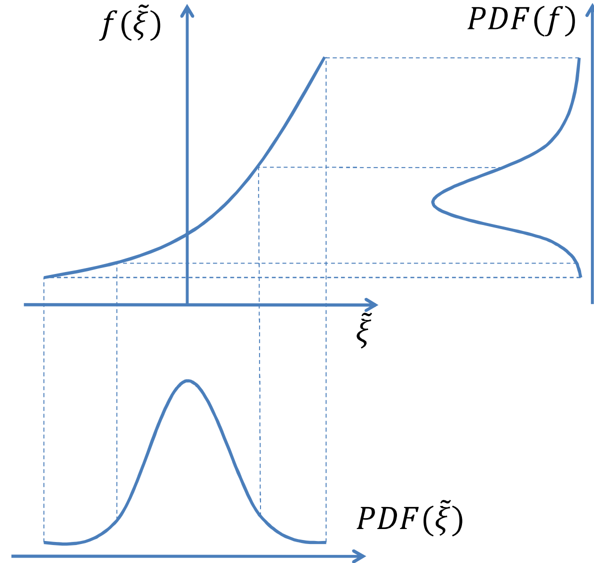

Fig. 4.5: Mapping the PDF of $\tilde{\xi}$ to the PDF of performance function f.

The accuracy of the method is verified by comparing the quantile prediction and PDF estimation given by the PLS method with Monte Carlo simulation

We implemented our method in MATLAB and applied it to ISCAS'85 benchmark circuits [8], all of which are digital logic circuits and are modeled using the framework outlined in section 4.1.1. To check the quality of the customized corner estimation, we also perform 10^4 Monte Carlo (MC) simulation for every benchmark circuits. The number of Monte Carlo samples is chosen such that the standard error of estimating the failure probability of 1% is

below 10% [3]. The MC simulation results are parsed using kernel density estimation [9], where Gaussian kernel is used and 100 points equally spaced across data range are used to evaluate the probability density function. The quantiles and the density function given by MC are compared to that estimated by PLS-based methods.

Fig. 4.6 shows the PDF estimation for the smallest circuit in ISCAS' 85: c17, which has 6 gates. The kernel density estimation result is shown in solid line, and the estimation given by our proposed PLS-based method is shown in dashed line. The histogram of Monte Carlo simulation result is also shown in the figure. 500 samples are used to run the PLS regression. The PDF estimation given by PLS regression closely matches that of density estimation via Monte Carlo. What is also worth noticing is that even though the underlying random variable is Gaussian, the critical delay is highly non-Gaussian because of nonlinearity and numerous *max* operations in block-based critical delay calculation [4]. This indicates that our method is capable of dealing with the nonlinearity and *max* operation in timing analysis, and the result is very close to MC simulation.

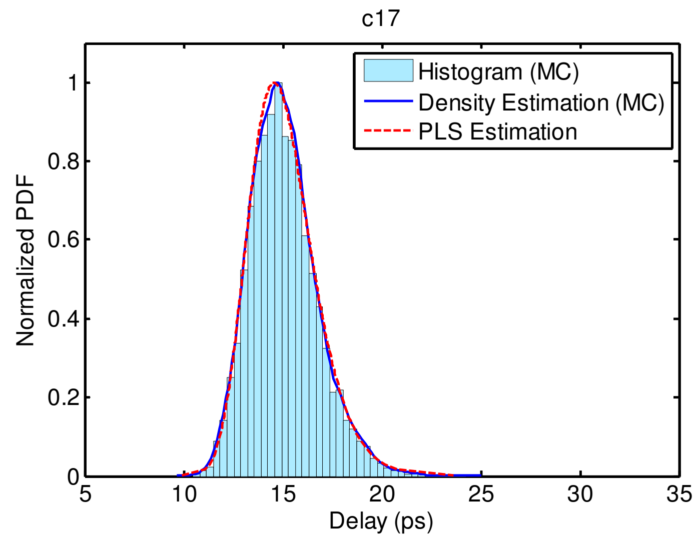

Fig. 4.6: PDF function comparison for c17. The solid line is given by kernel density estimation, and the dashed line is given by our proposed method based on PLS regression. 500 samples are used in PLS regression.

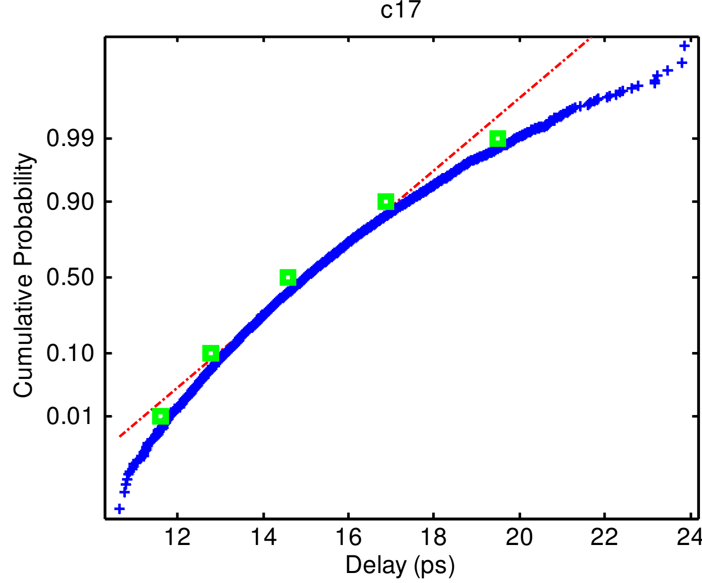

Fig. 4.7: Probability plot and quantile prediction for c17. The cross marks are Monte Carlo simulation results. The green boxes are predictions given by the PLS method. The straight dash line represents normal distribution.

The non-Gaussian characteristics of the critical delay are further illustrated in Fig. 4.7. In Fig. 4.7, the Monte Carlo results are plotted against the quantiles of standard normal distribution. Should the distribution be Gaussian, the data points would lay around a straight line. The deviation from the straight dashed line indicates light lower tail and heavy upper tail for critical delay distribution. The prediction given by PLS regression at 5 different probabilities (0.01, 0.1, 0.5, 0.9, 0.99) are marked by squares in Fig. 4.7. We can observe that they track the real distribution very well. If one uses a simple Gaussian distribution to make the estimation, one would obtain the points on the dashed line which can lead to significant errors, especially at the tails.

A similar PDF estimation result is shown in Fig. 4.8 for the largest circuit of ISCAS'85: c7552, which has 3512 gates. Fig. 4.8 shows that even for circuits with thousands of gates, the PDF estimation via the proposed method is still accurate as verified by MC simulation, and our method readily extends to larger problems of high dimensionality. Similar to Fig. 4.7, both the Monte Carlo simulation results and PLS regression prediction are plotted against the quantiles of a normal random variable in Fig. 4.9, which confirms the non-Gaussian characteristics of the result and the ability of the proposed method to track the real distribution.

In order to quantify the accuracy of the PDF estimation, we define the quantity PDF error as the integrated discrepancy of PDF estimation:

$$E_{PDF} = \int |PDF_{PLS}(x) - PDF_{MC}(x)|\, dx, \tag{4.14}$$

where $PDF_{PLS}(x)$ is the PDF estimation given by the PLS regression-based method, and $PDF_{MC}(x)$ is the PDF estimation given by kernel density estimation on Monte Carlo simulation results. Based on this definition, we summarize the PDF error for ISCAS'85

circuits in Table 2. For these results, 500 samples are used in PLS regression, and 10,000 samples are used in MC simulation. Table 4.2 shows that the PDF error for all circuits that we tested, most of them are below 0.1.

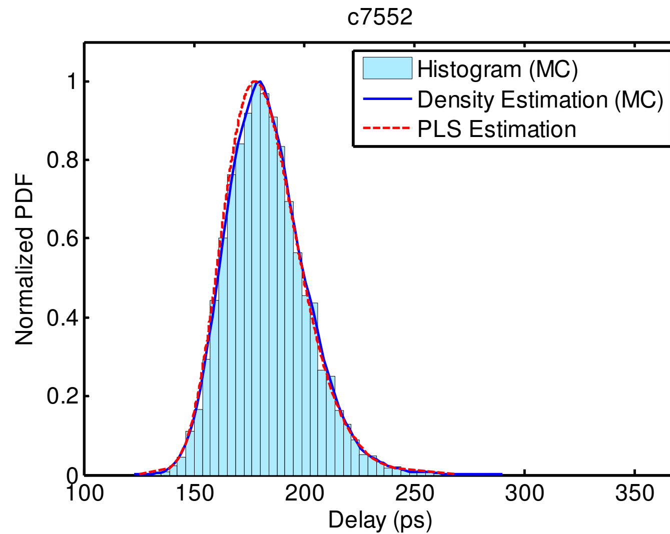

Fig. 4.8: PDF function comparison for c7552. The solid line is given by kernel density estimation, and the dashed line is given by our proposed method based on PLS regression. 500 samples are used for PLS regression.

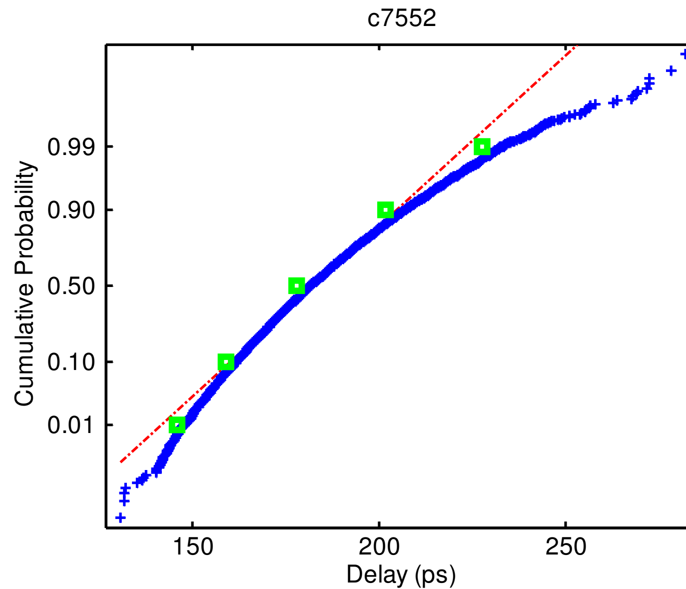

Fig. 4.9: Probability plot and quantile prediction for c7552. The cross marks are Monte Carlo simulation results. The green boxes are predictions given by PLS methods. The straight dash line represents normal distribution.

For some applications, we are interested in the quantiles at the tail of the distribution. To test the prediction capability at the tail, we define the 0.99-quantile error as

$$E_{0.99} = \frac{\left|Q_{PLS}^{(0.99)} - Q_{MC}^{(0.99)}\right|}{Q_{MC}^{(0.99)}}, \tag{4.15}$$

where $Q_{MC}^{(0.99)}$ and $Q_{PLS}^{(0.99)}$ are the 0.99-quantiles given by Monte Carlo simulation and PLS regression methods, respectively. The 0.99-quantile error for ISCAS'85 circuits are also summarized in Table 4.2. The prediction errors are below 3%. Compared with plain Monte Carlo, the PLS-based method shows faster convergence. This is shown by performing 50 independent experiments for both Monte Carlo and PLS-based method, and comparing the standard error of 0.99 quantile prediction. The speed-up is summarized at the last column of Table 4.2.

	Total Gates	PDF Error	0.99 Quantile Error	Speed up compared to MC
c17	6	0.0526	0.0205	6.32x
c432	160	0.0858	0.0168	4.51x
c880	383	0.0441	0.0115	5.68x
c1908	880	0.0554	0.0266	6.17x
c2670	1193	0.1121	0.0145	7.76x
c3540	1669	0.0599	0.0169	6.28x
c5315	2307	0.0294	0.0162	9.81x
c6288	2416	0.0732	0.0155	18.58x
c7552	3512	0.0459	0.0201	8.21x

Table 4.2: PDF error for ISCAS'85 benchmark circuits. 500 samples are used in PLS regression. 10,000 samples are used in MC simulation. The number of MC simulations is chosen such that the standard error of estimating the failure probability of 1% is 10%. The speed-up factor is the ratio of standard error for 0.99 quantile prediction achieved by MC and PLS method using 500 samples.

In the simulation results presented above, we have been using 500 samples for PLS regression. It is interesting to see how sensitive the accuracy of PDF estimation is to the number of samples. To this end, we run our method with various sample size from 10 to 1000, and the results on selected circuits are shown in Fig. 4.10. The result for c17 shown in solid line indicates that a very small sample size does not differ much from large sample size. This is reasonable since the dimension of c17 is only 6. For larger circuits such as c1908 (dotted line) and c7552 (dash line), a sample size that is too small can have a negative impact on the accuracy. However, the PDF errors for both circuits decrease to below 0.1 when the sample size is above 100. This indicates 100 samples can be a good choice for most applications. The same trend is also observed for 0.99-quatile error as shown in Fig. 4.11, which indicates 100 samples are enough for the estimation of 0.99-quantiles.

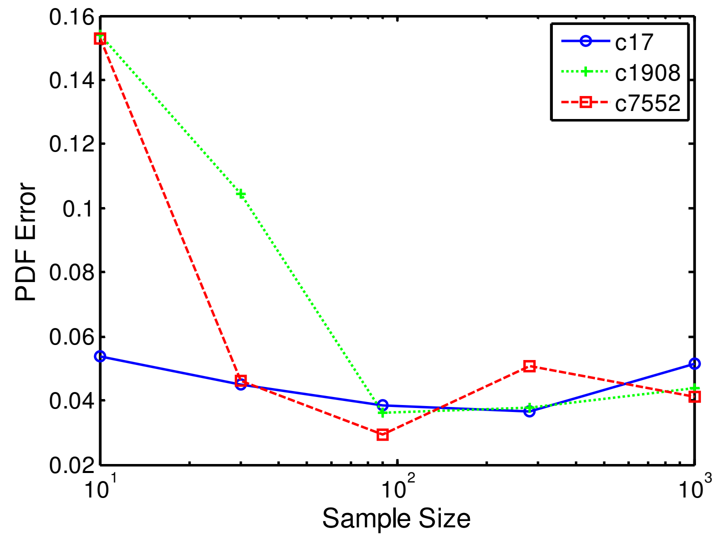

Fig. 4.10: PDF error vs. sample size in PLS regression.

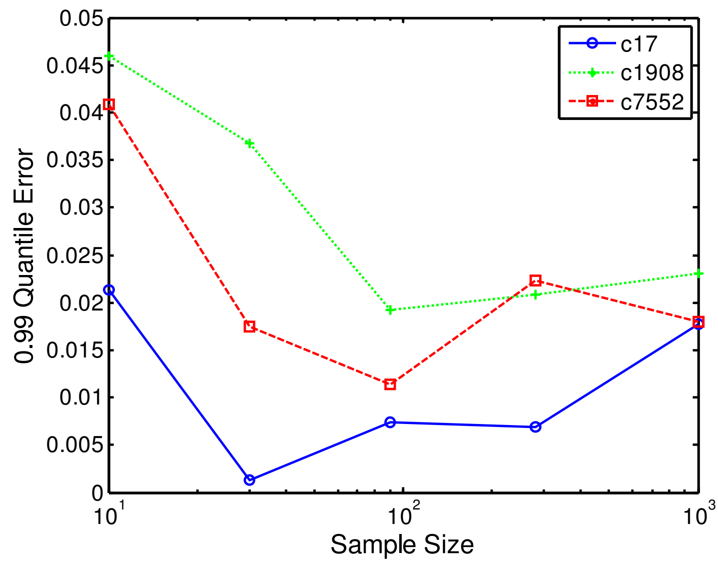

Fig. 4.11: 0.99-quantile error vs. sample size in PLS regression.

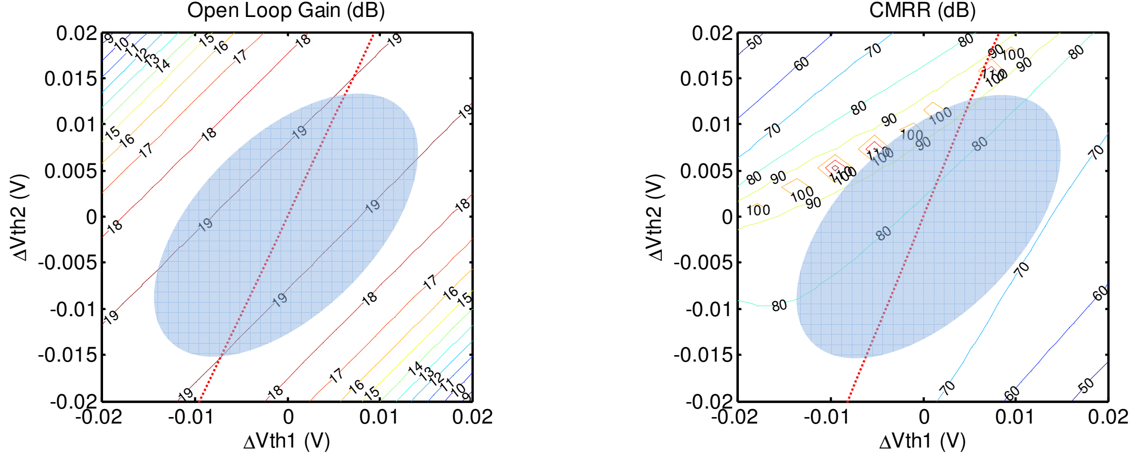

Fig. 4.12: The performance function contours in the $\Delta Vth1 - \Delta Vth2$ plane. The dotted line is the r direction given by PLS regression.

We also applied the method to the pre-amplifier example, but did not obtain meaningful results (Fig. 4.12). In Fig. 4.12, the ellipse describing the variability is superimposed upon the contour lines of the performance functions for both open loop gain and CMRR. What is also shown in dotted line is the r direction as picked by PLS regression. It is observed that the r direction given by the PLS regression is not the direction we really want. The reason is that both the open loop gain and the CMRR are highly nonlinear and non-monotonic functions of threshold voltage. In this situation, the PLS-based method is not applicable and we have to resort to optimization based methods as shown in the next section.

4.3 Optimization-based methods

4.3.1 Nonlinear classifier-assisted method

Provided the objective is to find the tangential point between the contours of the performance function and the ellipsoidal surface characterizing the variability, we can treat it as a pure geometric problem in the ξ-space. The biggest challenge is how to describe the contour shape of the performance function. The nonlinear classifier, such as support vector machine (SVM) classifier [10], is a powerful tool to address this problem in high dimensional space. It uses the kernel transformation [11] to implicitly work on an augmented space, thus capable of handling nonlinearity. In this section, we propose to use SVM classifier to describe the performance function contours and solve a nonlinear equation to find the tangential point.

The method starts with simulations on a number of training samples. The sampled points are artificially divided into two classes based on the simulated performance value. This is achieved by introducing a threshold F, such that half of the samples have performance values smaller than F, and those samples are labeled by the number 0. The other half is labeled by the number 1 (Fig. 4.13). An SVM classifier separating the two classes characterizes the hyper-surface where the performance function has the value F. Therefore, the separating surface of the classifier approximates the contours of the performance function. We then use this contour information to find the tangential point with the ellipsoid surface as described in detail in the following.

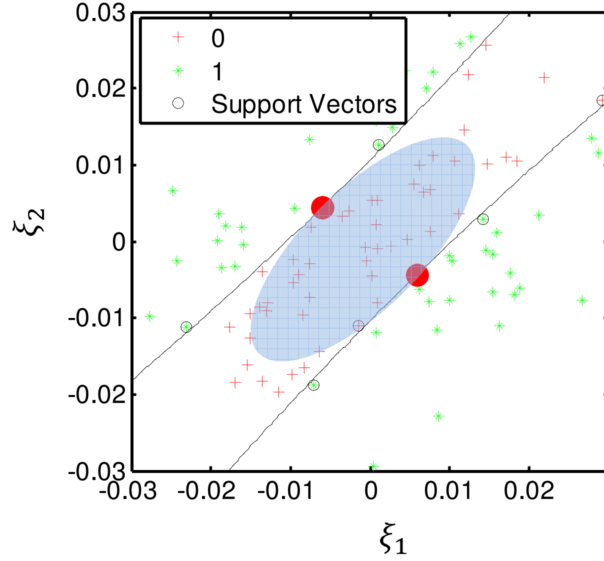

Fig. 4.13: Illustration of the nonlinear classifier-based method.

Suppose the decision equation in SVM is given by

$$D(\xi) = \sum_{i=1}^{N} \alpha_i y^{(i)} K(\xi^{(i)}, \xi) + b, \tag{4.16}$$

where N is the number of training samples, $y^{(i)}$ is the indicator function taking value 1 or -1 for sample (i), α_i and b are fitting parameters given by the training process. We use the quadratic kernel function in our analysis:

$$K\left(\xi^{(i)}, \xi\right) = \xi^T \xi^{(i)} \left(1 + \xi^T \xi^{(i)}\right). \tag{4.17}$$

The normal direction at the separating surface is given by

$$d_1 = \nabla D(\xi) = \sum_{i=1}^{N} \alpha_i y^{(i)} \nabla K(\xi^{(i)}, \xi). \tag{4.18}$$

On the other hand, the normal direction of the ellipsoid is

$$d_2 = \nabla(\xi^T \Sigma^{-1} \xi) = 2\Sigma^{-1}\xi. \tag{4.19}$$

What we intend to find is the point where $d_1 || d_2$ and $D(\xi) = 0$. This amounts to solving the following nonlinear simultaneous equations:

$$\begin{cases} \left(d_1^T d_2\right)^2 = \|d_1\|^2 \|d_2\|^2 \\ D(\xi) = 0. \end{cases} \tag{4.20}$$

Eq. (4.20) is solved iteratively using Levenberg–Marquardt method [12].

To solve Eq. (4.20), 10 different points are randomly generated as initial values, and the equation (4.20) is solved for each of them. Many of the solutions are identical, and we merge them if they are close to each other. The customized corners given by Eq. (4.20) are tested at different yield tolerance levels. We use the number of sigma's γ to represent the yield tolerance level. The quantity γ is set so that the ellipsoid characterizing the variability is expressed as $\{\xi: \xi^T\Sigma^{-1}\xi \leq \gamma^2\}$. We let the size of the variation ellipsoid to change from γ =2 to 3. At each γ level, we randomly sample 1000 points on the surface of the ellipsoid $\{\xi: \xi^T\Sigma^{-1}\xi \leq \gamma^2\}$. If the corner points picked by Eq. (4.20) are the ideal corner points, then the performance values at the 1000 sample points should not be worse than that given by the corner points. We count how many points out of the entire sample are not confined by the customized corner prediction, and call that the error rate. The results are shown in Fig. 4.14. It is observed that for open loop gain, most of the sampled points are confined by the customized corner prediction, and only 1 out of 1000 at the yield level $\gamma = 2$ generates an error. For CMRR, the estimated customized corners envelope the worst-performance points to some extent. But there is a small portion of the sampled points not confined by the estimated customized corners, and the error rate can be as high as 2.5%.

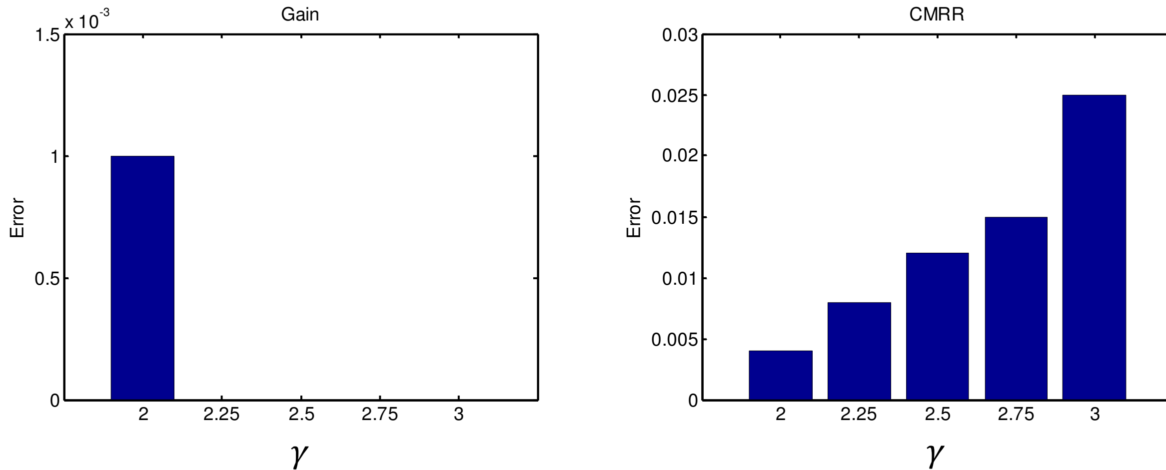

Fig. 4.14: The error rate tested by ellipsoid surface samples for the classifier-based method.

4.3.2 Direct optimization method

The method presented in the last section seeks to find the tangential point based on the description of the performance function contours using SVM. We have shown that the method, though approximates the customized worst-case corner points to some extent, cannot find the desired worst-case points with high accuracy. In this section, we intend to work directly on the optimization problem:

$$\begin{array}{ll} \text{minimize} & f(\xi) \\ \text{subject to} & \xi^T\Sigma^{-1}\xi \leq \gamma^2. \end{array} \tag{4.21}$$

The optimization problem of Eq. (4.21) is solved using the interior point method with the initial points picked randomly. The solutions to the problem are regarded as the customized corner points for performance metric $f(\xi)$.

The result is checked via the same procedure as detailed in 4.3.1. For every γ level, 10 randomly generated initial points are used in the optimization problem (4.21). The error rates are shown in Fig. 4.15. The error rates for both the open loop gain and CMRR are reduced below 10^{-3} across the yield tolerance range we simulated. This means that the customized corner as picked by Eq. (4.21) represents the worst-case performance at a given yield level. But the quality of the corner estimation comes at the price of the computation. Our simulation shows that the optimization problem Eq. (4.21) takes 100~300 function evaluations to reach the solution, and this problem is solved for every choice of the initial points. By contrast, in classifier-based method, we used only 100 evaluations to simulate the training sample, and the subsequent nonlinear equation solving does not require any expensive simulation.

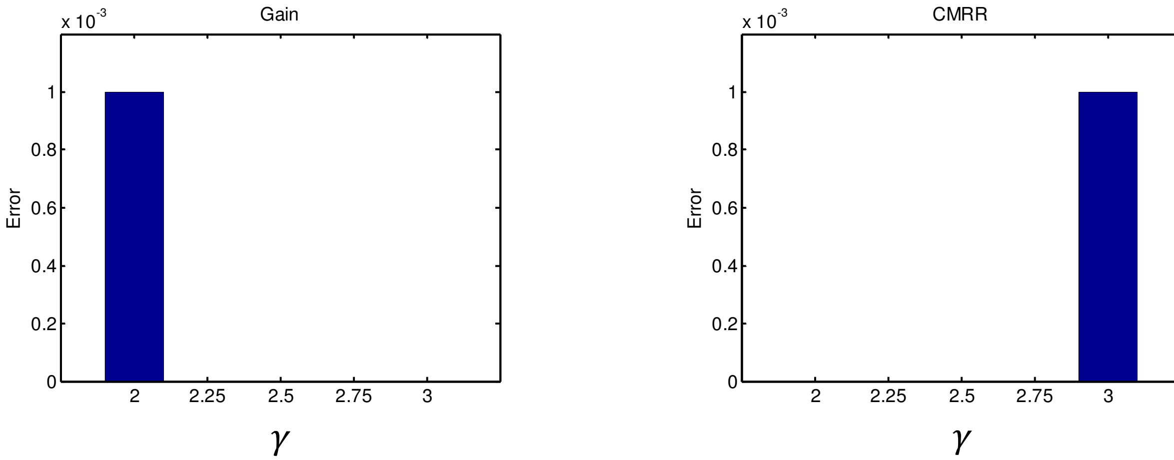

Fig. 4.15: The error rate tested by ellipsoid surface samples for the direct optimization method.

4.4 Summary

The PLS-based method estimates the customized corner through algebraic calculation. Its major advantage is computational speed and the ability to handle large scale problem. It is shown to be powerful in estimating the customized corner for critical delay of digital logic circuits, but it cannot give good estimation for analog applications.

For analog circuits, we propose to use nonlinear classifier-based method. An SVM classifier is built on artificially divided training sample. A nonlinear equation is solved based on the decision function to yield the customized corners. These corners are shown to be good representatives for the worst-performance points, but can fail sometimes. A more robust method, based on direct optimization, is able to improve the result, but at the expense of a more computationally-intensive optimization procedure.

The comparison is also summarized in Table 4.3.

Method	Speed	Accuracy
PLS-based	Fast	Good only for digital circuits
Nonlinear Classifier method	Medium	Good
Direct optimization	Slow	Best

Table 4.3: Comparison of the methods in Chapter 4.

4.5 References

[1] P. Friedberg, W. Cheung, G. Cheng, Q. Y. Tang and C. J. Spanos, "Modeling spatial gate length variation in the 0.2um to 1.15mm separation range," SPIE Vol. 6521, 652119 (2007)

[2] J. M. Rabaey, A. Chandrakasan and B. Nikolic, "Digital integrated circuits: a design perspective," second edition, Prentice Hall (2003)

[3] Y. Ben, L. El Ghaoui, K. Poolla and C. J. Spanos, "Yield-constrained digital circuit sizing via sequential geometric programming," IEEE International Symposium on Quality Electronic Design, 114 (2010)

[4] Cristiano Forzan, Davide Pandini, "Statistical static timing analysis: A survey," Integration, the VLSI Journal, Volume 42, Issue 3, 409-435 (2009)

[5] T. Sakurai and A. Newton, "Alpha-power law MOSFET model and its application to CMOS inverter delay and other formulas," IEEE Journal of Solid-State Circuits 25(2), pp. 584–593 (1990)

[6] P. R. Gray, P. J. Hurst, S. H. Lewis and R. G. Meyer, "Analysis and design of analog integrated circuits," 4th edition, Wiley (2001)

[7] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm design exploration," IEEE International Symposium on Quality Electronic Design, 717-722 (2006)

[8] http://www.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html

[9] A. W. Bowman, and A. Azzalini. Applied Smoothing Techniques for Data Analysis. New York: Oxford University Press, (1997)

[10] T. Hastie, R. Tibshirani and J. Friedman, "The elements of statistical learning: data mining, inference, and prediction," second edition, Springer (2009)

[11] B. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in Proc. 5th Annual ACM Workshop on Computational Learning Theory, New York, NY: ACM Press, pp. 144-152 (1992)

[12] K. Levenberg, "A Method for the Solution of Certain Non-Linear Problems in Least Squares". *The Quarterly of Applied Mathematics* **2**: 164–168 (1944)

Chapter 5 Mathematical Formulation of Optimal Circuit Design

This chapter gives an overview of formulating the circuit design practice as an optimization problem. It establishes the mathematical framework of which the solution will be discussed in the next chapter. We start our discussion with deterministic circuit design without the impact of process variation. It is followed by the introduction of the yield-constrained design problem and its worst-case surrogate. Current methods to solve the worst-case problem and the yield-constrained problems are introduced and their limitations are discussed. At the end of this chapter, we describe the digital gate sizing problem in detail as we will use it as an example to illustrate our methods in the next chapter.

5.1 Deterministic Circuit Design Problem

When designing a circuit, designers are given a set of specifications and possibly several performance metrics to optimize. The circuit coming out of the design phase must meet these specifications, and is hoped to achieve the highest possible performance. Designers have the freedom of choosing different circuit topologies and tuning design parameters to achieve the goal.

This design practice can be formulated as a mathematical optimization problem. Suppose the design parameters are denoted as a vector x, then circuit performance metrics are functions of x. For every specification, it is possible to express the requirement posed by it as an inequality $g_i(x) \leq 0$, where $i = 1, \dots, p$, with p being the number of specifications. We can denote the circuit performances to be optimized as a vector $f(x)$, of which every entry corresponds to a performance metric. Based on these notations, the circuit design problem is mathematically formulated as:

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & g_i(x) \leq 0, \quad i = 1, \dots, p \end{array} \quad (5.1)$$

This is known as a multi-objective optimization problem since the objective function is a vector. It is often solved by optimizing the weighted sum of all entries in the objective function vector, i.e. a scalar $f_o(x) = w^T f(x)$ [1]. This scalarization procedure gives rise to the solution that lies on the Pareto-optimal surface defined in the performance metric space [25]. Since we will always scalarize the objective function, we assume in our formulation that the function $f(x)$ in Eq. (5.1) is a scalar.

It is possible that in many design settings there is no performance to be optimized, and the only design objective is to meet specs. It is easy to encompass this type of problem in the formulation. This can simply be done by setting the objective function $f(x)$ equal to a constant $f(x) = 1$. With the objective function being a constant, the optimization problem stated in Eq. (5.1) is known as a feasibility problem [1].

It is also worth noting that the design variable can be used not only to describe the parametric design choices, but also circuit topologies. This generalization allows the formulation in Eq. (5.1) to cover the entire circuit design problem. The resulting problem is a mixed combinatorial optimization [2]. Although the nonlinear mixed combinatorial optimization problem can be tackled via heuristics, it is in general very hard to solve. In our discussion, we consider only the situation where the topology is fixed, and the design variable x is parametric. It corresponds to the parameter tuning phase in circuit design (chapter 1).

The mathematical optimization formulation can be applied to various circuit types at different levels in design hierarchy. For instance, Ref. [3] models the gate-level digital logic design as a geometric programming problem; Ref. [4] minimizes the power consumption of SRAM cell at both circuit and layout level; Ref. [5] obtains the optimal design of CMOS op-amp via geometric programming.

To solve the mathematical optimization problem, one can use general-purpose nonlinear optimization methods as described in the survey paper [6]. Although these methods have been successfully employed in various design situations [7], [8], they suffer from being frequently trapped in locally optimal designs. To circumvent this difficulty, global optimization techniques such as branch and bound [9] and simulated annealing [10] can be adopted. Branch and bound method can reach the global optimal solution without using heuristics, but the convergence in general is extremely slow. Simulated annealing method uses stochastic method to reduce the chances of finding a non-globally optimal solution, but it cannot be avoided completely and the convergence tends to be slow. In the last two decades, convex optimization gained plenty of attention. Once the problem is formulated as a convex optimization problem [1], powerful methods such as interior-point [12] can be used and the convergence to globally optimal solution is guaranteed. The major disadvantage of a convex optimization approach is that the problem must conform to the form of convex optimization. In the digital-circuit-sizing example discussed in the following sections, the problem takes the form of geometric programming [1], a form of convex optimization, and therefore can be solved efficiently using interior method. In other cases where the problem is not inherently convex, or closed form expression of the function is not available, one can approximate the problem with a convex formulation [5].

5.2 Robust circuit design

The design problem described in the previous section assumes a deterministic relation between circuit performance and design. In reality, the performance can fluctuate around its nominal value due to the variation arising from the manufacturing process or operating environment. As discussed in chapter 1, the impact from manufacturing process variation is becoming more pronounced as the technology nodes shrink to 32 nm and below. Any realistic design must take into account the process variation [13].

5.2.1 Yield-constrained design

We assume the variability can be described by random variables. For consistency with the preceding chapters, we denote the parameters related to process variation as a vector ξ.

Every entry in ξ is a random variable, and ξ is a random vector following certain distribution. As a result, the performance metrics are also functions of ξ.

In practice, we want the probability of meeting specifications to be greater than a certain value $1-\delta$, where δ is a small number, e.g. 0.01. This probability is also known as parametric yield. Equivalently we can also ask the probability of failing to meet specs be smaller than δ. We shall replace the deterministic constraint in Eq. (5.1) by a probability constraint $\Pr(\xi: \exists i \; g_i(x,\xi) > 0) \le \delta$. The objective that we try to minimize is now the expectation of the objective function. This new problem is a yield-constrained optimization problem, and is formally stated as:

$$\begin{array}{ll} \text{minimize} & E(f(x,\xi)) \\ \text{subject to} & \Pr\left(\xi: \exists i \; g_i(x,\xi) > 0\right) \le \delta \end{array} \quad (5.2)$$

We denote the formulation as described by Eq. (5.2) as the yield-constrained design problem. It is also recognized as the "chance-constrained optimization" problem in the community of convex optimization [14-19].

5.2.2 Worst-case Design

In most situations, the problem as defined in Eq. (5.2) is extremely hard to solve. So people resort to conservative approximations to the yield-constrained problem. The most popular approach is the worst-case design.

Suppose we choose a set Ω, such that the probability for ξ to take a value outside of the set is small enough, then we can design the circuit with the worst-case scenario that can possibly be given by that set. This idea can be formulated by the following worst-case design problem:

$$\begin{array}{ll} \text{minimize} & \max_{\xi \in \Omega} f(x,\xi) \\ \text{subject to} & \max_{\xi \in \Omega} g_i(x) \le 0, \quad i = 1, \dots, p \end{array} \quad (5.3)$$

In many situations, the optimization problem in Eq. (5.3) is easier to handle than that in Eq. (5.2). For instance, the problem given by Eq. (5.3) often maintains the convexity of its deterministic counterpart whereas the chance-constrained problem Eq. (5.2) does not [20]. But its disadvantage is obvious: the set Ω is often larger than necessary since people have poor knowledge about the connection between the size of the set and the final parametric yield, and hence use conservative estimation.

In summary, it is ideal to work on the yield-constrained optimization problem (5.2). But the practical difficulty of lacking efficient algorithm prevents us from directly tackling the yield-constrained problem. Instead, people opt to deal with the conservative worst-case problem (5.3) as happened in both convex optimization community [20] and design automation community [5], [21]. In recent years, many researchers started to look at the yield constrained problem [22], [23]. However, these existing methods either resort to overly simplified circuit performance formulae in order to conform to a particular optimization formulation, or rely on the assumption that the variability carries certain closed-form distribution. The latter is a particularly serious problem since the closed-form

distribution can disagree tremendously with reality at the remote tail of the distribution, which in fact is the most important region for yield estimation. A realistic variability model extracted from data [24] is often too complicated to be included in the existing optimization framework. Therefore, a generic optimization framework that is capable of utilizing a more realistic and accurate variability model is desired. We will introduce our contribution to the problem in chapter 6. The example we choose to work on is the gate sizing problem [3], which is going to be described in the next section.

5.3 Example: Digital Circuit Sizing

In this section, we introduce a design example that we will use in the next chapter: digital circuit sizing. As discussed in section 4.1.1, there are two important numbers associated with the digital logic: one is the longest time it takes for the signal to traverse the block; it is called critical delay $D_{critical}$; the other one is the maximum allowable delay determined by the clock frequency and the setup time of the flip flop; and we denote it as D_{max}. In order for the circuit to work, the critical delay must be less than D_{max}, and we refer to the situation where critical delay is greater than D_{max} as failure event.

As in chapter 4, we adopt the RC mode to describe the critical delay. Unlike the RC model in chapter 4, the size-dependence of the RC parameters will be considered in this example. If we use x_i to describe the size of gate i, then the equivalent resistance R_i^{eq} is represented as a constant divided by gate size

$$R_i^{eq}(x) = \frac{R_i}{x_i}. \tag{5.4}$$

The constant R_i depends on gate type, threshold voltage etc., but is independent of gate size. The equivalent capacitance is the sum of all caps connected to this gate, and it is proportional to the size:

$$C_i^{eq}(x) = C_i^{int} x_i + \sum_{j \in FO(i)} C_j^{in} x_j, \tag{5.5}$$

where C_i^{int} is unit-size intrinsic capacitance of gate i, C_j^{in} is the unit-size input capacitance of gate i, and $FO(i)$ denotes the indices of the gates that connect to the output of gate i, or the fan-out gate i. The delay of a single gate can be expressed by $D_i(x) = 0.69 R_i^{eq}(x) C_i^{eq}(x)$.

Starting from the input, there could be several paths leading to the output, and the delay associated with that path is the sum of the delay of the gates belonging to that path:

$$D_p = \sum_{i \in path(p)} 0.69 R_i^{eq}(x) C_i^{eq}(x) = \sum_{i \in path(p)} R_i \widetilde{C}_\iota(x), \tag{5.6}$$

where we group all the terms that are not related to R_i together and call them $\widetilde{C}_\iota(x)$. $\widetilde{C}_\iota(x)$ is a posynomial [1] of x. By definition, the critical delay the maximum of all path delays:

$$D_{critical} = \max_{p} \left(\sum_{i \in path(p)} R_i \widetilde{C}_i(x) \right). \quad (5.7)$$

5.3.1 Deterministic Digital Circuit Sizing

The objective of digital circuit sizing is to tune the size of the circuit x, to optimize some metric under performance constraints. In this work, we assume that we would like to minimize the area of the circuit, or equivalently, the dynamic power consumption of the circuit. It is a linear function of the size:

$$A(x) = a^T x. \quad (5.8)$$

where a is a constant vector of which every entry a_i is the unit size area of gate i. In addition to the performance constraint that the critical delay being less than D_{max}, the gate sizes cannot be less than one due to minimum feature size limitation. Putting all these together, we have the following deterministic digital circuit sizing problem

$$\begin{array}{ll} \text{minimize} & A(x) = a^T x \\ \text{subject to} & D_{critical}(x) < D_{max} \\ & x \geq 1. \end{array} \quad (5.9)$$

The critical delay $D_{critical}(x)$ is given in Eq. (5.7), and the coefficients for different types of gates used in our simulation are summarized in Table 4.1 of chapter 4. This problem is recognized as a geometric programming (GP) problem [1], and can be solved efficiently using off-the-shelf algorithms.

5.3.2 Yield-constrained Digital Circuit Sizing

We adopt the same variability model as described in section 4.1.1, where ξ is the random variable characterizing the threshold voltage fluctuation. Since equivalent resistance is a function of random variable ξ based on the variability model described in section 4.1.1, the critical delay is also a function of it. We can re-define the digital circuit sizing problem as

$$\begin{array}{ll} \text{minimize} & A(x) = a^T x \\ \text{subject to} & P_f = \Pr(D_{critical}(x, \xi) > D_{max}) \leq \delta \\ & x \geq 1, \end{array} \quad (5.10)$$

incorporating a yield constraint.

We will discuss various methods to solve (5.10) in Chapter 6.

5.4 References

[1] S. Boyd and L. Vandenberghe, "Convex optimization," Cambridge University Press (2004)

[2] M. W. Cooper, "A survey of methods for pure nonlinear integer programming," Management Science, Vol. 27, pp 353-361 (1981)

[3] S. P. Boyd, S.-J. Kim, D. D. Patil, and M. A. Horowitz, "Digital circuit optimization via geometric programming," Oper. Res., vol. 53, no. 6, pp. 899–932 (2005)

[4] E. Morifuji, D. Patil, M. Horowitz and Y. Nishi, "Power optimization for SRAM and its scaling," IEEE Trans. Electron Devices vol. 54, No. 4 (2007)

[5] M. D. M. Hershenson, S.P. Boyd and T. H. Lee, "Optimal design of a CMOS Op-Amp via geometric programming," IEEE Transactions on Computer-aided design of integrated circuits and systems, Vol. 20, No.1, (2001)

[6] R. K. Brayton, G. D. Hachtel and A. Sangiovanni-Vincentelli, "A survey of optimization techniques for integrated circuit design," Proc. IEEE, vol. 69, pp. 1334-1362 (1981)

[7] P. C. Maulik and L. R. Carley, "High-performance analog module generation using nonlinear optimization," in Proc. 4th Annu. IEEE Int. ASIC Conf. Exhibit, pp. T13-5.1 (1991)

[8] P. C. Maulik, M. J. Flynn, D. J. Allstot and L. R. Carley, "Rapid redesign of analog standard cells using constrained optimization techniques," Proc. IEEE Custom Integrated Circuit Conf., pp. 8.1.1-8.1.3 (1992)

[9] C. Xinghao and M. L. Bushnell, "Efficient branch and bound search with application to computer-aided design," Norwell, MA: Kluwer (1996)

[10] D. F. Wong, H. W. Leong and c. L. Liu, "Simulated annealing for VLSI design," Norwell, MA: Kluwer (1988)

[11] D. G. Luenberger, "Linear and nonlinear programming," second edition, Addison-Wesley (1984)

[12] Y. Nesterov and A. Nemirovsky, "Interior-point polynomial methods in convex programming," Philadelphia, PA: SIAM (1994)

[13] L. Liebman, "DfM, the teenage years," Proc of SPIE Vol. 6925, 692502 (2008)

[14] A. Nemirovski and A. Shapiro, "Convex approximations of chance constrained programs," SIAM J. on Optimization, vol. 17, no. 4, pp. 969-996 (2006)

[15] A. Ben-Tal and A. Nemirovski, "Stable truss topology design via semidefinite programming,' SIAM J. on Optimization 7:4, 991-1016 (1997)

[16] A. Ben-Tal and A. Nemirovski, "Robust convex optimization," Math. Of Oper. Res. 23:4 (1998)

[17] L. El Ghaoui and H. Lebret, "Robust solution to least-square problems with uncertain data," SIAM J. of Matrix Anal. Appl. 18, 1035-1064 (1997)

[18] L. El Ghaoui, F. Oustry and H. Lebret, "Robust solutions to uncertain semidefinite programs," SIAM J. on Optimization 9, 33-52 (1998)

[19] A. Ben-Tal and A. Nemirovski, "Robust solutions of linear programming problems contaminated with uncertain data," Math. Program., Ser. A 88: 411-424 (2000)

[20] A. Ben-Tal, L. El Ghaoui and A. Nemirovski, “Robust Optimization,” Princeton University Press (2009)

[21] X. Liu, W. Luk, Y. Song, P. Tang and X. Zeng, “Robust analog circuit sizing using ellipsoid method and affine arithmetic,” ASP-DAC (2007)

[22] J. Singh, V. Nookala, Z. Luo and S. Sapatnekar, “Robust gate sizing by geometric programming,” DAC (2005)

[23] M. Mani and M. Orshansky, “A new statistical optimization algorithm for gate sizing,” IEEE International Conference on Computer Design (2004)

[24] K. Qian, B. Nikolic, and C. J. Spanos, “Hierarchical modeling of spatial variability with a 45nm example,” Proc of SPIE, 727505 (2009)

[25] F. Kashfi, S. Hatami, and M. Pedram, “Multi-objective optimization techniques for VLSI circuits,” pp 156, ISQED (2011)

Chapter 6 Solving the Yield-constrained Design Problem

The presence of the yield constraint makes it hard to solve the yield-constrained optimization problem. As shown in Chapter 2, it is difficult to evaluate the yield; it is even more difficult to include it in an optimization problem. One cannot treat the yield as just another nonlinear function and throw the entire problem in Eq. (5.12) into a generic nonlinear optimization solver, because every function can be evaluated hundreds of times before the solution is reached and that poses a daunting computational burden if one of the functions being evaluated is the yield. A more feasible approach is to replace the original problem with approximations.

In this chapter, we will introduce three methods. The first method, gradient projection, can be applied with high flexibility but has convergence problems when the dimensionality is high. The second approach, convex approximation, approximates the yield constraint with convex constraints. This approach can efficiently handle the situation where the variability does not possess much correlation, but becomes problematic when strong correlation is present. The third method, sequential geometric programming, can overcome most of the difficulty by reducing the original problem to a sequence of convex optimization problems, and the sequence is controlled by fast yield simulation. This allows it to be applicable in various situations.

We will use the example of digital circuit sizing described in Chapter 5 to illustrate different methods.

6.1 Gradient projection method

In this section, we propose a method that is intended to be as generic as possible to solve the problem (5.10), which is re-stated here for the convenience of discussion:

$$\begin{array}{ll} \text{minimize} & A(x) = a^T x \\ \text{subject to} & P_f = \Pr(D_{critical}(x,\xi) > D_{max}) \le \delta \\ & x \ge 1. \end{array} \tag{6.1}$$

Since it is impossible to directly apply generic nonlinear optimization method to this problem due to computational challenge of evaluating failure probability, our approach is to reduce the number of P_f calculation while treating the rest of the problem as a common nonlinear optimization problem. Before we delve into the method itself, we will first give a brief introduction to the generic nonlinear optimization methods.

In general, nonlinear programming methods can be categorized according to the orders of derivatives involved [1]. Second-order methods need evaluation of the second-order derivatives, or Hessian, of the function being optimized, first-order methods need gradient, and zeroth-order methods only need the evaluation of the function value itself. These methods [1], [2] are summarized in Table 6.1. The 0th order methods are very flexible in the forms of the problems they can deal with, but these methods tend to converge very slowly. The 1st order methods take advantage of the gradient information to speed up the

convergence, but they require the functions to be smooth such that the gradient calculation can be carried out. The 2nd order methods are the fastest if the functions are smooth and convex. In fact, if the objective function is a quadratic function, a 2nd order method such as Newton's method can reach the solution in a single iteration. However, the drawback of 2nd order methods is also as pronounced. If the Hessian matrix at the point being evaluated is not positive-definite, the algorithm could give rise to a catastrophic step and a meaningful solution could never be reached.

Method	0th order	1st order	2nd order
Examples	Nelder-Mead Genetic Algorithm	Steepest descent Projected gradient Cutting-plane	Newton Interior-point BFGS
Speed	Slow	Medium	Fast
Robustness to Noise	Good	Good	Bad

Table 6.1: Comparison of different nonlinear optimization methods

An important feature about using simulation in failure rate estimation is that the evaluation result contains noise, and this will incur significant numerical errors in gradient and Hessian calculation. Based on the discussion in the last paragraph, all 2nd order methods risk the failure of convergence if we were to use them directly on the yield-constrained problem. First-order methods provide good resilience to noise, while at the same time they maintain descent convergence speed compared to 0th order methods.

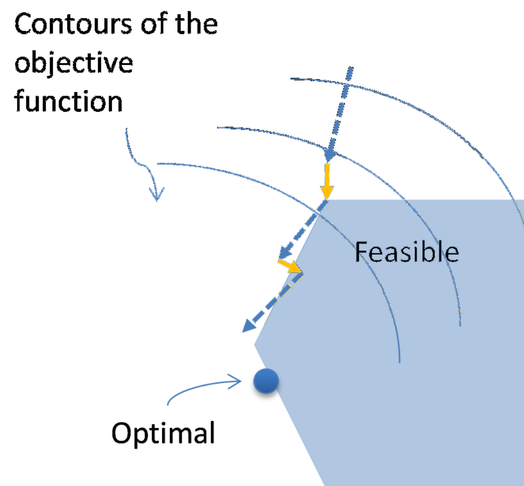

Fig. 6.1: Projected gradient method.

Among various 1st order methods, we choose to use the projected gradient method [1], [3] because it allows completely separate treatment of the objective function and constraints. In this way, we can make the number of yield evaluations as small as possible. The method is a combination of gradient descent and feasibility projection and it will be described next in some detail.

But before we embark upon the explanation of the algorithm itself, we need to take a closer look at the problem. The failure probability will be evaluated through stochastic simulation, which is essentially the mean of a large number of random results. As we have shown in Chapter 2, the estimate $\hat{P}_f$ is approximately a random variable following Gaussian distribution according to the central limit theorem. Given the fact that the evaluation of P_f is random in nature, the yield constraint needs to be modified in order to accommodate this randomness. If we would like to reject the fraction non-conforming γ with probability $1-\beta$, we borrow the idea of upper control limit (UCL) in statistical process control theory [4] and arrive at the modified constraint

$$P_f = \Pr(D_{critical}(x,\xi) > D_{max}) \leq \tilde{\delta}, \tag{6.2}$$

where $\tilde{\delta}$ is given by

$$\tilde{\delta} = \delta - (Z_\gamma + Z_\beta)\frac{k\delta}{1+kZ_\gamma}. \tag{6.3}$$

The quantities Z_β and Z_γ are the $(1-\beta)$ and $(1-\gamma)$ quantile of the standard normal distribution.

6.1.1 The algorithm

With the modified yield constraint, we are ready to describe the projected gradient method. First, we calculate the gradient g_A of the objective function by finite difference at an arbitrary initial point. The point is then updated along the direction of g_A, by step length α_A (as shown by the blue dash arrows in Fig. 6.1). The updated point is often infeasible. We call a subroutine to project the point back to feasible region (yellow solid arrows in Fig. 6.1). The new objective function is calculated at this new feasible point. If the function change is smaller than *Tolfun*, the loop is stopped; if not, the entire process is repeated. The algorithm reads:

Repeat until change in $A(x)$ is smaller than *TolFun*
1 Get the local gradient g_A of $A(x)$
2 Update x to x - $\alpha_A\, g_A$
3 Project x to feasible region
4 Calculate $A(x)$

There remains one question: how to project an infeasible point to feasible region. The projection operation for a size constraint $x \geq 1$ is straightforward: if $x_i < 1$, set it to be 1. For the yield constraint, we will decrease the critical delay if the failure probability is out of spec. This can be done by moving on the descend direction of the critical delay and is implemented by the following algorithm:

Repeat if x is infeasible
1 Get a local gradient g_D of $D_{critical}(x)$
2 Update x to x - $\alpha_D\, g_D$

6.1.2 Results and Discussion

The method outlined in section 6.1.1 is implemented in MATLAB 7.4.0 on a desktop with 2.4 GHz Intel Core 2 Quad processor. The method is first applied to the ISCAS '85 benchmark circuit c17 [5] which has 6 gates. The parameters used in the simulation are summarized in Table 6.2. The objective function $A(x)$ is recorded at every step, and its values are plotted in Fig. 6.2. As shown in Fig. 6.2, the convergence is achieved after a number of iterations. Even though the convergence is not comparable to the quadratic convergence rate of the Newton's method [1], it still gives the result within a reasonable number of iterations. The optimized circuit is tested by Monte Carlo simulation, of which the histogram of the critical delay is shown in Fig. 6.3. The histogram shows the probability of exceeding D_{max} is approximately 1%, indicating the yield constraint is active, just as desired.

TolFun	α_D	α_A	D_{max}	δ	γ	β
0.5	0.1	0.05	12.5 ps	0.01	0.05	0.20

Table 6.2: Simulation parameters in optimizing ISCAS '85 c17.

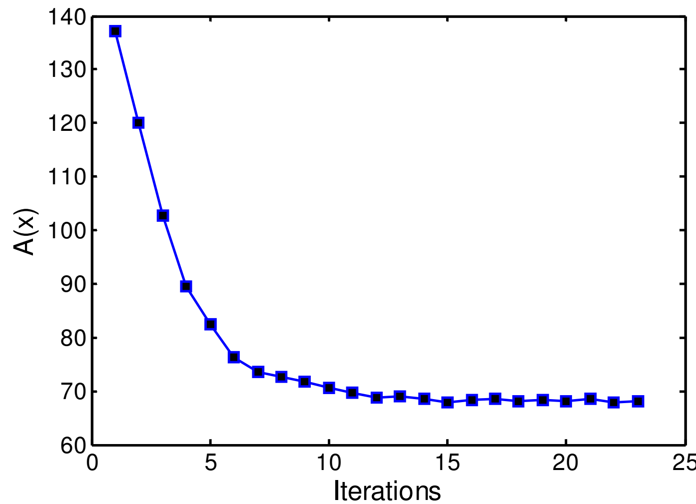

Fig. 6.2: Objective function vs. iteration for ISCAS '85 c17.

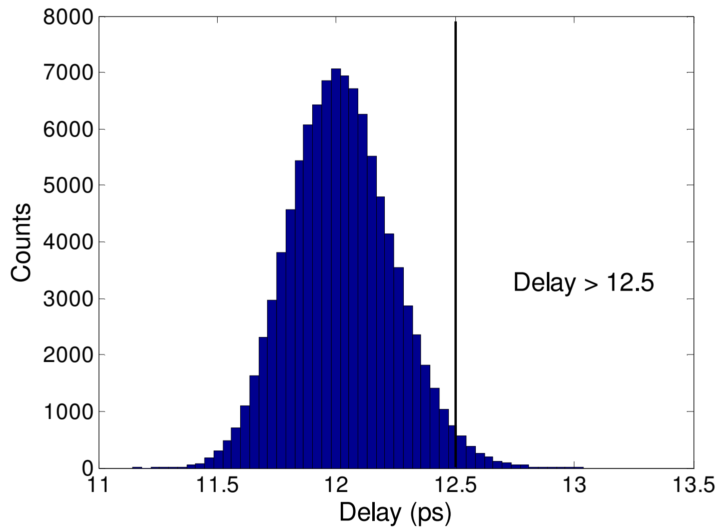

Fig. 6.3: Delay histogram of the optimized c17.

In order to study how the method scales with the problem size, we create a scalable circuit by cascading a basic unit (Fig. 6.4). As a result, the number of paths grows exponentially as a function of the number stages. The maximum tolerable delay is set to 12.5 ps×Nstage (number of stages), and the simulation parameters are summarized in Table 6.3. We plot the run-time as a function of the number of gates in Fig. 6.5 and as a function of the number of paths in Fig. 6.6. It can be seen that the run-time grows linearly with the path number, but exponentially with the number of gates. This indicates this simulation-based projected gradient method, though flexible, is slow in general.

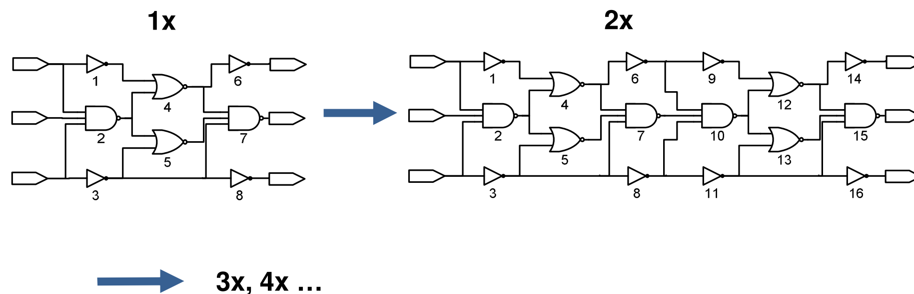

Fig. 6.4: Circuit cascade. The basic circuit is modified from the one used in [6].

TolFun	α_D	α_A	D_{max}	δ	γ	β
0.5	0.1	0.05	12.5 ps×Nstage	0.01	0.05	0.20

Table 6.3: Simulation parameters used in optimizing cascaded circuit.

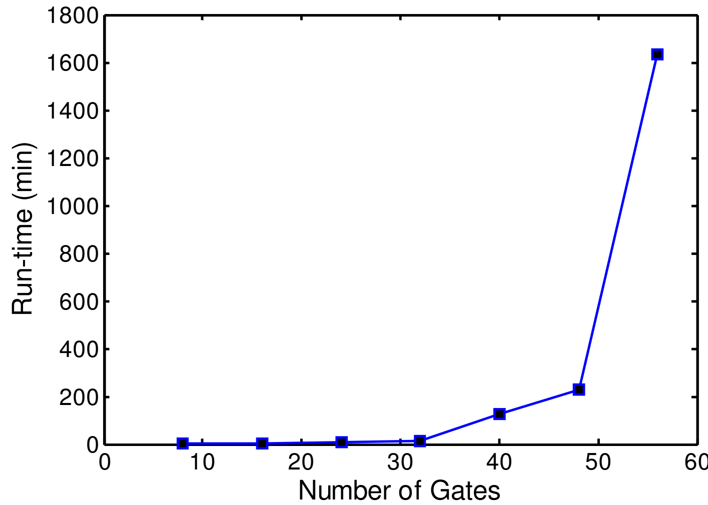

Fig. 6.5: Run-time as a function of the number of gates.

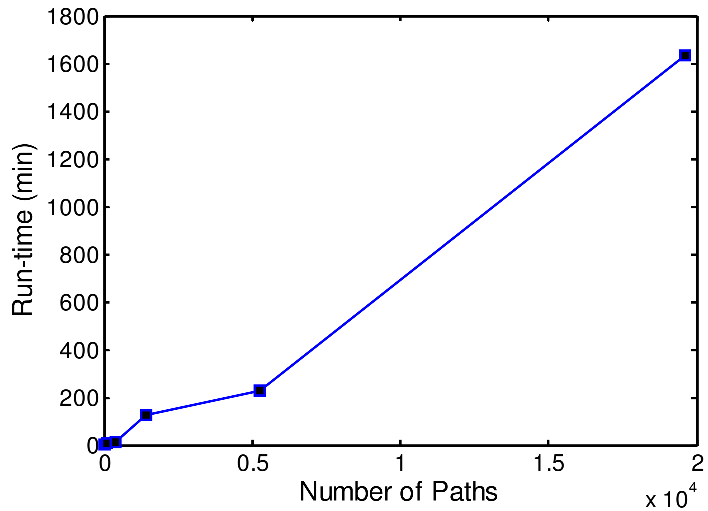

Fig. 6.6: Run-time as a function of the number of paths.

6.2 Convex Approximation

The method in this section is derived from the most recent developments in robust convex optimization [7]. The yield-constrained problem is also known as the chance-constrained optimization in that field. The study of robust convex optimization itself dates back to 1973 [8]. The entire field was revived around 1997 [9-12], and was followed by significant progress in both theory and applications. Systematic research on chance-constrained robust convex problem was not seen until very recently [13], [14]. Most of these approximation techniques have been successfully used in the robust versions of Linear Programming (LP) and Semi-Definite Programming (SDP). Here we will modify these techniques to make them applicable in geometric programming as found in the

circuit sizing problem. To facilitate the usage of convex approximation techniques, we first need to modify the original problem.

Suppose we can express the variability of the equivalent resistance explicitly as

$$R_i = R_i^0 + R_g\zeta_g + R_i^r\zeta_i, \tag{6.4}$$

where the global perturbation random variable $\zeta_g \in [-1,1]$; the deterministic constant $R_g \geq 0$ denotes the range of impact of global perturbation; the local perturbation random variables $\zeta_i \in [-1,1]$ are independent of each other and are independent of the global perturbation variable ζ_g; the deterministic constant $R_i^r \geq 0$ characterizes the range of impact of the local perturbation. Information about intra-die spatial variability can be added if gate positions and the form of spatial variability are known. Notice that the only assumption on ζ_g and ζ_i is that they take values on [-1,1]; otherwise we do not assume them to follow any particular distribution.

Next, we will relate the coefficients in Eq. (6.4) to the variability model described in section 4.1.1. In section 4.1.1, we stated that the resistance is linked with the threshold voltage by the α-power law $R_i \propto \left(V_{dd} - V_{th,i}\right)^{-\alpha}$, and the threshold voltage is described by a hierarchical model

$$V_{th,i} = V_{th,i}^0 + \xi_i = V_{th,i}^0 + a_g\epsilon_g + a_i\epsilon_i \tag{6.5}$$

To find the coefficients R_g and R_i^r in Eq. (6.4), we simply let the global and local random variables in Eq. (6.5) to take their extreme values and estimate the change in threshold voltage, and consequently, in equivalent resistance through the α-power law. In case where the range of ϵ_g or ϵ_i is infinite, a practically "distant" value can be used instead (e.g. 3σ value in case of Gaussian). The goal is to obtain a rough estimate of the impact of the variability in threshold voltage on the equivalent resistance, through the coefficients R_g and R_i^r. This can be achieved as long as a simulation can be carried out to calculate $V_{th,i}$. In our example, we assume ϵ_g and ϵ_i's are independent and they follow the standard normal distribution $\mathcal{N}(0,1)$. In order to estimate the parameters R_g and R_i^r, we let ϵ_g and ϵ_i's to take their 3σ value, i.e. ϵ_g, ϵ_i =3, and arrive at the following formula:

$$R_g = R_i^0 \frac{\left(V_{dd} - V_{th,i}^0\right)^\alpha}{\left(V_{dd} - V_{th,i}^0 - 3a_g\right)^\alpha} - R_i^0, \tag{6.6}$$

$$R_i^r = R_i^0 \frac{\left(V_{dd} - V_{th,i}^0\right)^\alpha}{\left(V_{dd} - V_{th,i}^0 - 3a_i\right)^\alpha} - R_i^0. \tag{6.7}$$

6.2.1 Box Approximation

An obvious way of replacing the yield constraint in Eq. (6.1) is to dictate the critical delay to be less than $D_{\max}$ for all possible perturbation values $\zeta_g, \zeta_i \in [-1,1]$. If we view ζ_g

and ζ_i's as elements of a vector $\zeta = (\zeta_g, \zeta_i \dots \zeta_m)^T$, then ζ takes values that are determined by the perturbation set $\mathcal{Z}_{\text{Box}} = \{\zeta : \|\zeta\|_\infty \le 1\}$. A robust approximation to the yield constraint in Eq. (6.1) becomes

$$\max\left(\sum_{i\in \text{path}(p)} \left(R_i^0 + R_g\zeta_g + R_i^r\zeta_i\right)\widetilde{C}_\iota(x)\right) \le D_{max}\,, \qquad \forall \zeta \in \mathcal{Z}_{\text{Box}}, p \in \text{All Paths} \tag{6.8}$$

Because $R_g, R_i^r \ge 0$, Eq. (6.8) is equivalent to

$$\sum_{i\in \text{path}(p)} \left(R_i^0 + R_g + R_i^r\right)\widetilde{C}_\iota(x) \le D_{max}, \qquad \forall p \in \text{All Paths} \tag{6.9}$$

The resulting optimization problem is a geometric programming problem [15]. Since the perturbation set $\mathcal{Z}_{\text{Box}}$ has the shape of a box, the approximation Eq. (6.9) is also called a *box* approximation [10]. The box approximation guarantees that the delay requirement is always satisfied under the variability model described in Eq. (6.4); therefore it is a robust but overly conservative approximation of the yield constraint in Eq. (6.1). Since the worst possible situation is included, the design problem Eq. (6.1) with the yield constraint replaced by Eq. (6.9) is indeed a conservative worst-case design.

6.2.2 Ball Approximation

The box approximation satisfies the constraint overlooking the fact that $D_{critical} \le D_{max}$ can be violated with a finite probability. In order to remove this redundancy, a perturbation set that is smaller than $\mathcal{Z}_{\text{Ball}}$ should be used. It is established in [10] that using the *ball* perturbation set $\mathcal{Z}_{\text{Ball}} = \{\zeta : \|\zeta\|_2 \le \omega\}$ with $\omega = \sqrt{2\ln(1/\delta)}$ is a less conservative yet still safe approximation of the original chance constraint. Notice that the radius of the ball ω is determined by the maximum tolerable yield loss δ. The smaller the yield loss δ, the larger the radius will be. Under the perturbation described by $\mathcal{Z}_{\text{Ball}}$, the yield constraint in Eq. (6.1) is approximated as

$$\max\left(\sum_{i\in \text{path}(p)} \left(R_i^0 + R_g\zeta_g + R_i^r\zeta_i\right)\widetilde{C}_\iota(x)\right) \le D_{max}\,, \qquad \forall \zeta \in \mathcal{Z}_{\text{Ball}}, p \in \text{All Paths} \tag{6.10}$$

and it is equivalent to

$$\sum_{i\in path(p)} R_i^0\widetilde{C}_\iota(x) + \omega\sqrt{\sum_{i\in path(p)}\left(R_i^r\widetilde{C}_\iota(x)\right)^2 + \left(\sum_{i\in path(p)} R_g\widetilde{C}_\iota(x)\right)^2} \le D_{max}, \qquad \forall\, p \in \text{All Paths.} \tag{6.11}$$

The ball approximation takes its name from the fact that the perturbation set $\mathcal{Z}_{\text{Ball}}$ has the shape of a ball. The left hand side of the approximation is generalized posynomial [15]. Thus the resulting optimization problem is still a geometric programming problem.

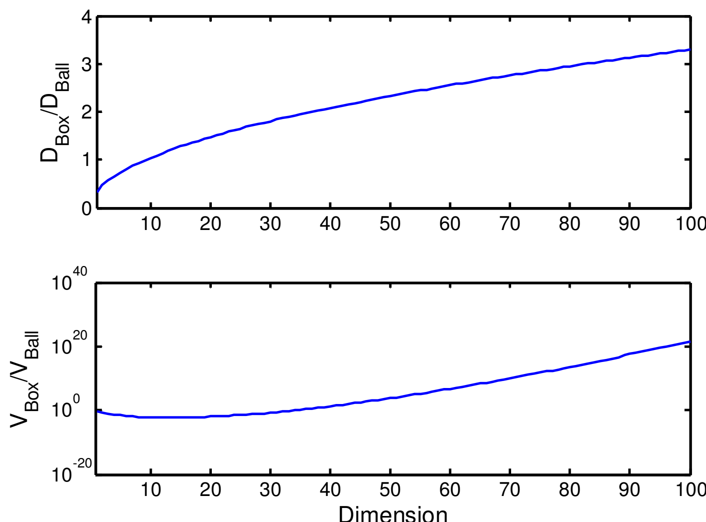

Fig. 6.7: The comparison of the size of a box and a ball in high dimension.

One can understand the benefit of using ball approximation against box approximation via geometric arguments. If the perturbation set $\mathcal{Z}$ is large, then the approximated constraint is conservative, and vice versa. The volume of $\mathcal{Z}_{\text{Box}}$ in n-dimensional space is $V_{box} = 2^n$ since the side length of the box is 2. The volume of an n-dimensional ball with radius ω is [16]

$$V_{ball} = \frac{\pi^{\frac{n}{2}}}{\Gamma\left(\frac{n}{2}+1\right)}\omega^n, \tag{6.12}$$

where Γ is the gamma-function. As the dimensionality n increases, the volume of the box will be incomparably larger than that of the ball. A numerical example with $\omega = \sqrt{2\ln(1/\delta)}$ and $\delta = 0.01$ is shown in Fig. 6.7. The upper panel of Fig. 6.7 shows the ratio of the diameter of the box (max diagonal distance within the box) to the diameter of the ball as a function of dimension. The lower panel shows the ratio of the volume of the box to that of the ball. It is clear that the ball can be significantly smaller than a box in high dimensional space. It is this unintuitive property of the high dimensional ball that guarantees the effectiveness of the ball approximation method.

6.2.3 Bernstein Approximation

A more sophisticated way to approximate the yield constraint is through the Bernstein approximation [10], in which the distribution $P_i(\zeta_i)$ associated with each perturbation variable ζ_i is assumed to satisfy

$$\int \exp(ts)dP_i(s) \le \exp\left\{\max\left[\mu_i^+ t, \mu_i^- t\right] + \frac{1}{2}\sigma_i^2 t^2\right\}, \tag{6.13}$$

where μ_i^+, μ_i^- and σ_i are deterministic constants. A large number of well-known distributions fall under this category. For a unimodal distribution defined on [-1,1], it is shown [10] that $\mu_i^+ = 0.5$, $\mu_i^- = 0.5$ and $\sigma_i = 1/\sqrt{12}$. Under the unimodal assumption, the approximation of the yield constraint is

$$\begin{aligned} &\frac{\omega}{\sqrt{12}} \sqrt{\sum_{i \in path(p)} \left(R_i^r \widetilde{C}_\iota(x)\right)^2 + \left(\sum_{i \in path(p)} R_g \widetilde{C}_\iota(x)\right)^2} \\ &+ \sum_{i \in path(p)} \left(R_i^0 + \frac{1}{2}R_i^r + \frac{1}{2}R_g\right)\widetilde{C}_\iota(x) \le D_{max}, \qquad \forall\, p \in \text{All Paths}. \end{aligned} \tag{6.14}$$

The left hand side of this approximation is also a generalized posynomial [15], thus the approximated problem is a geometric programming problem, and can be solved using efficient algorithms.

6.2.4 Dynamic Programming Formulation

The summations that appear in the above constraints (6.9), (6.11) and (6.14) are for all paths. This could pose a computational problem because the number of paths can grow exponentially with the number of gates; hence the number of constraints grows correspondingly. To address this problem, one can adopt the dynamic programming formulation as described here. Suppose we define the delay associated with gate i as D_i, then the arrival time T_i^D of the signal right after the gate must satisfy (Fig. 6.8)

$$T_i^D \ge T_j^D + D_i, \qquad \forall j \in \text{FI}(i), \tag{6.15}$$

where the notion FI(i) represents the indices of the gates that connect to the input of gate i, or the fan-in of gate i. In box approximation (6.9), the term in the summation is the gate delay $D_i = \left(R_i^0 + R_g\zeta_g + R_i^r\zeta_i\right)\widetilde{C}_\iota(x)$, and equation (6.9) can be written as

$$T_i^D \ge T_j^D + \left(R_i^0 + R_g + R_i^r\right)\widetilde{C}_\iota(x), \qquad \forall j \in \text{FI}(i). \tag{6.16}$$

If we want the summation of the gate delays along all paths to be less than some value D_{max} as in (6.9), then we only need to look at the arrival time at the output nodes and set those to be less than D_{max}. We can further simplify this by appending a single artificial output node with zero delay right after the real output nodes, with all real output nodes as its fan-in. If we denote this new artificial node as o, we can replace the inequalities involving all paths with a single inequality of T_o^D

$$T_o^D \le D_{max}. \tag{6.17}$$

The complete problem with box approximation reads:

$$\begin{aligned} \text{minimize} \quad & A(x) = a^T x \\ \text{subject to} \quad & T_i^D \geq T_j^D + \left(R_i^0 + R_g + R_i^r\right)\widetilde{C}_\iota(x), \qquad \forall j \in \mathrm{FI}(i) \\ & T_o^D \leq D_{max} \\ & x \geq 1. \end{aligned} \tag{6.18}$$

To deal with the first summation under the square root in ball approximation (6.11), we can similarly introduce an artificial circuit that carries the same topology as the real one, but the "delay" associated with each gate is defined as $(D_i^r)^2 = \left(R_i^r \widetilde{C}_\iota(x)\right)^2$. In the same manner, if we introduce the arrival time T_i^r for this artificial circuit, we have

$$T_i^r \geq T_j^r + \left(R_i^r \widetilde{C}_\iota(x)\right)^2, \quad j \in \mathrm{FI}(i). \tag{6.19}$$

Similarly, for the second summation under the square root in Eq. (6.11), we define arrival time T_i^g on the same topology:

$$T_i^g \geq T_j^g + R_g \widetilde{C}_\iota(x), \quad j \in \mathrm{FI}(i). \tag{6.20}$$

The complete problem with ball approximation reads:

$$\begin{aligned} \text{minimize} \quad & A(x) = a^T x \\ \text{subject to} \quad & T_i^D \geq T_j^D + R_i^0 \widetilde{C}_\iota(x), \qquad j \in \mathrm{FI}(i) \\ & T_i^r \geq T_j^r + \left(R_i^r \widetilde{C}_\iota(x)\right)^2, \quad j \in \mathrm{FI}(i) \\ & T_i^g \geq T_j^g + R_g \widetilde{C}_\iota(x), \quad j \in \mathrm{FI}(i) \\ & T_o^D + \omega\sqrt{T_o^r + \left(T_o^g\right)^2} \leq D_{max} \\ & x \geq 1. \end{aligned} \tag{6.21}$$

The summation in Bernstein approximation (6.14) can be handled using similar methods, and the complete problem reads:

$$\begin{aligned} \text{minimize} \quad & A(x) = a^T x \\ \text{subject to} \quad & T_i^D \geq T_j^D + \left(R_i^0 + \frac{1}{2}R_i^r + \frac{1}{2}R_g\right)\widetilde{C}_\iota(x), \qquad j \in \mathrm{FI}(i) \\ & T_i^r \geq T_j^r + \left(R_i^r \widetilde{C}_\iota(x)\right)^2, \quad j \in \mathrm{FI}(i) \\ & T_i^g \geq T_j^g + R_g \widetilde{C}_\iota(x), \quad j \in \mathrm{FI}(i) \\ & T_o^D + \frac{\omega}{\sqrt{12}}\sqrt{T_o^r + \left(T_o^g\right)^2} \leq D_{max} \\ & x \geq 1. \end{aligned} \tag{6.22}$$

The optimization problems stated in (6.18), (6.21) and (6.22) are all recognized as geometric programming (GP) problems, a well-known formulation in convex optimization, solvable by very efficient off the shelf algorithms.

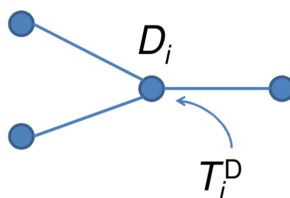

Figure 6.8: Abstract network

6.2.5 Results and Discussion

We apply the three approximated GP problem formulations to an 8-bit adder [17] with 88 gates. The maximum allowable delay is set to $D_{\max}$=70ps, and δ is set to 0.01. The GP problems (6.18), (6.21) and (6.22) are modeled and solved using the Matlab toolbox of MOSEK [18], on a desktop with 2.4 GHz Intel Core 2 Quad processor and Matlab 7.4.0. After the optimization, the optimally sized circuits are tested with a 50,000-run Monte Carlo simulation to find the delay distribution, where the random numbers are generated according to (6.5).

We test the method in two scenarios. In the first test scenario, the global variability coefficient a_g is assumed to be 0, and local variability coefficient $a_i = 20$ mV. This is the situation where the global variation does not play a significant role and the local variability dominates. The simulated histogram is shown in the upper panel of Fig. 6.9. The optimized result given by the different methods are shown in the lower panel of Fig. 6.9. It is clearly seen that, both ball approximation and Bernstein approximation give better result compared to the worst-case (box approximation) optimization. The conservativeness of the worst-case approach is observed from the histogram, where the Bernstein approximation is the least conservative approach of these three. The run-time (Table 6.4) numbers suggest that the approximation methods introduced here are very efficient.

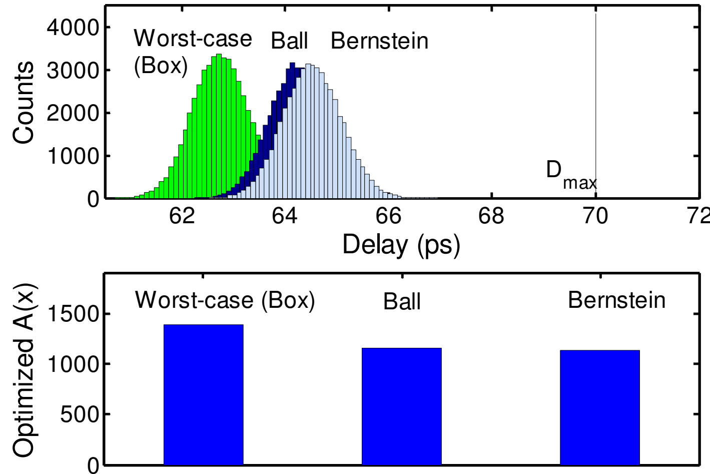

Fig. 6.9: Upper panel: the delay histogram from Monte Carlo simulation on optimized results. Lower panel: the optimized area.

To further study the efficiency of these methods, we employ them on ISCAS'85 benchmark circuits. The run-time of different methods is shown in Table 6.5. The proposed methods can solve circuits with hundreds of gates in seconds, and circuits with thousands of gates in minutes. The exact run-time of course depends on the architecture in addition to the size of the circuits.

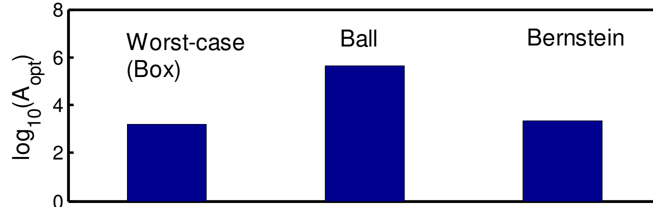

Fig. 6.10: Optimized area in scenario 2.

In the second test scenario, we adopt the hierarchical model where the global variability dominates, or equivalently, the variability across the entire circuit exhibits high correlation. In this test, we set $a_g = 20$ mV, and $a_i = 0.1a_g$. The optimized areas given by different methods are summarized in Fig. 6.10. What can be observed from this figure is opposite to the result in scenario 1: ball approximation gave much worse result than the box-approximation. The optimized area given by Bernstein approximation is also larger than that of box-approximation. It contradicts our assertion that ball and Bernstein approximations are less conservative than the box approximation, and this needs further discussion.

To understand the phenomenon, we need to take a close look at the path delay formula organized with respect to ζ:

$$\sum_{i\in\text{path}(p)} \left(R_i^0 + R_g\zeta_g + R_i^r\zeta_i\right)\widetilde{C}_i(x) = \sum_{i\in\text{path}(p)} R_i^0\widetilde{C}_i(x) + \left(\sum_{i\in\text{path}(p)} R_g\widetilde{C}_i(x)\right)\zeta_g + \sum_{i\in\text{path}(p)} R_i^r\widetilde{C}_i(x)\zeta_i\,. \tag{6.23}$$

At the second line of Eq. (6.23), the coefficients in front of ζ_g are contributed by all gates in the path, whereas that of ζ_i come from a single gate i. Moreover, because a_g is one order of magnitude larger than a_i, R_g is much larger than R_i^r according to Eq. (6.6) and (6.7). Based on these facts, we can conclude that the impact due to ζ_g is much larger than ζ_i, making the effective dimension of the problem low. Recall that the ball approximation scheme is only effective when the dimensionality is high; so, ball-approximation is not an appropriate choice for the situation when there is a strong global variability component.

	Box	Ball	Bernstein
Run-time (sec)	0.1719	0.2031	0.3750

Table 6.4: Run-time of different methods on an 8-bit adder.

	Total Gates	Run-time (sec)		
		Box	Ball	Bernstein
c17	6	0.0313	0.0625	0.0469
c432	160	4.6250	2.5156	2.6563
c880	383	0.7813	1.6406	2.0625
c1355	546	1.8125	2.9219	2.3125
c1908	880	1.5000	7.2031	7.6094
c2670	1193	2.5781	7.7500	10.1719
c3540	1669	10.7813	23.0156	16.1875
c5315	2307	5.1875	30.7656	46.9688
c6288	2416	46.3906	90.8438	97.2813
c7552	3512	16.0313	32.5156	35.7969

Table 6.5: Run-time on ISCAS'85 Benchmark Circuits (seconds).

6.3 Sequential Geometric Programming

It was shown in the last section that the convex approximation methods are not good choice when global variability dominates or variation across the circuit exhibits high correlation. We propose a sequential method in this section to overcome this problem. It is based upon the box-approximation of the original yield-constrained problem.

6.3.1 Scaled Box Approximation

Since the approximation deduced from the box perturbation set is more conservative than necessary, we can use a set that is smaller than $\mathcal{Z}_{\text{Box}}$, provided, of course, that the original yield constraint is never violated. One possible way is to scale the size of the box. Suppose that the size of the box is scaled by a scalar factor of f_{Box}, the scaled perturbation set is $\mathcal{Z}'_{\text{Box}} = \{\zeta : \|\zeta\|_\infty \leq f_{\text{Box}}\}$. If we require the critical delay to be less than $D_{\max}$ for all the perturbations defined by $\mathcal{Z}'_{\text{Box}}$, then the approximation (6.8) becomes

$$\begin{gathered}\sum_{i \in \text{path}(p)} \left(R_i^0 + f_{\text{Box}}(R_g + R_i^r)\right)\widetilde{C}_i(x) \leq D_{\max} \\ \forall p \in \text{All Paths}.\end{gathered} \tag{6.24}$$

We call (6.24) a *scaled box* approximation. Now the problem becomes determining the appropriate scale factor f_{Box} that will help us achieve the desired yield without being too conservative.

Similar to the dynamic programming formulation in the convex approximation approach, we introduce the idea of arrival time in order to avoid the path summation. In scaled box approximation (6.24), the term in the summation is the gate delay $D_i = \left(R_i^0 + f_{\text{Box}}(R_g + R_i^r)\right)\widetilde{C}_i(x)$, and the maximum arrival time T_i^D of the signal at the output of the gate satisfies the following inequality

$$\begin{gathered}T_i^D \geq T_j^D + \left(R_i^0 + f_{\text{Box}}(R_g + R_i^r)\right)\widetilde{C}_i(x), \\ \forall j \in \text{FI}(i).\end{gathered} \tag{6.25}$$

Similar to the last section, we can append a single artificial output node with zero delay right after the real output nodes, with all real output nodes as its fan-in. The arrival time at this output node o should be less than $D_{\max}$:

$$T_o^D \leq D_{\max}. \tag{6.26}$$

The complete problem with scaled box approximation is

$$\begin{array}{ll} \text{minimize} & A(x) = a^T x \\ \text{subject to} & T_i^D \geq T_j^D + \left(R_i^0 + f_{\text{Box}}(R_g + R_i^r)\right)\widetilde{C}_i(x), \\ & \qquad\qquad \forall j \in \text{FI}(i) \\ & T_o^D \leq D_{\max} \\ & x \geq 1. \end{array} \tag{6.27}$$

6.3.2 Sequential Geometric Programming

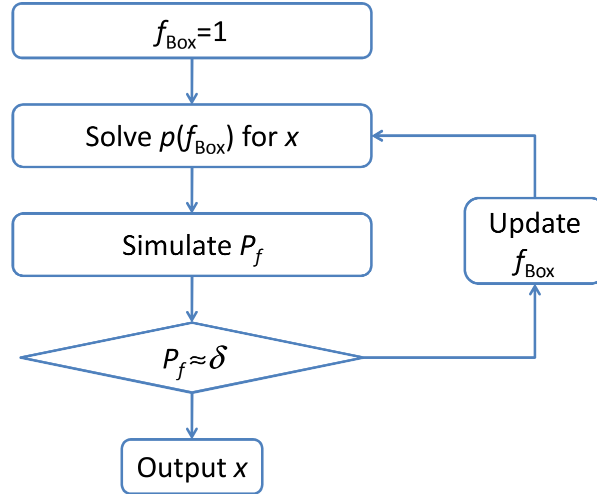

Fig. 6.11: Flow diagram of sequential geometric programming.

Let us denote the problem (6.27) as $p(f_{Box})$. If f_{Box}=1, then $p(1)$ is the conservative worst-case problem. If f_{Box}=0, the solution to the problem $p(0)$ is a risky design solution because no randomness is taken into account, and the resulting failure probability P_f is bigger than any δ that one might be interested in. Based on these observations, we conclude that the best f_{Box} that can make P_f close to δ is bounded in [0,1]. Therefore, we can iteratively tune the scale factor f_{Box}, solve the associated problem $p(f_{Box})$, and, through simulation, calculate the failure probability P_f corresponding to the solution at each iteration. If P_f is close enough to δ, then the algorithm exits; if not, a new value of f_{Box} is generated for the next iteration. The entire procedure is summarized in Fig. 6.11.

The algorithm can be viewed as a line search for the best f_{Box}. The line search can be any generic line search algorithm, and we use bisection in our examples. The notion "close enough" will become concrete in section 6.3.4. Because the optimization problem being solved at each iteration is a geometric programming problem (6.27), we call this method sequential geometric programming (SGP).

In the current implementation of SGP, we use a single scalar parameter f_{Box} to describe the set from which ζ's can take values. This limits the shape of the set. More parameters can be introduced to relax this limitation at the expense of more outer iterations.

It is understood that the geometric programming problem can be solved very efficiently in polynomial time as has been shown theoretically and observed experimentally [15]; the remaining question of SGP is how to efficiently calculate the failure probability. This is the topic of chapter 3. For the specific problem discussed in this chapter, importance sampling is found to be effective as will be shown in the next section.

6.3.3 Importance Sampling in SGP

As we discussed in Chapter 3, the variance reduction achieved in importance sampling strongly depends on the choice of the biased distribution $h(\xi)$, where $\xi = (\xi_1, \xi_2, \dots \xi_m)^T$and $\xi_i = a_g \epsilon_g + a_i \epsilon_i$. In this section we will work explicitly in the ϵ-space for

which the reason will be obvious from the following discussion. Ideally, one wants to shift the original distribution $g(\xi)$ to the place where failure is most likely to happen, or equivalently the point on the boundary that is the closest to the center of the original distribution. The search for such points in a $(m+1)$-dimensional space, where m (the number of gates) could be in the thousands, can be extremely time consuming. The situation can become even worse when there are many vertices in the boundary due the fact that many paths are nearly critical, and small perturbations can cause the critical delay to shift from one path to another. Fortunately, under the hierarchical model (6.5), the critical delay grows the most rapidly along the direction of the global variability axis ϵ_g (since global variation is typically much larger than local), therefore the boundary of interest is nearly parallel to the local variation axes ϵ_i. We can largely ignore the boundary points on all local variation axes. Consequently, the closest boundary can be approximately found by shifting $g(\xi)$ along ϵ_g axis by an amount $\epsilon_g^{\text{shift}}$ to the boundary as shown in Fig. 6.12.

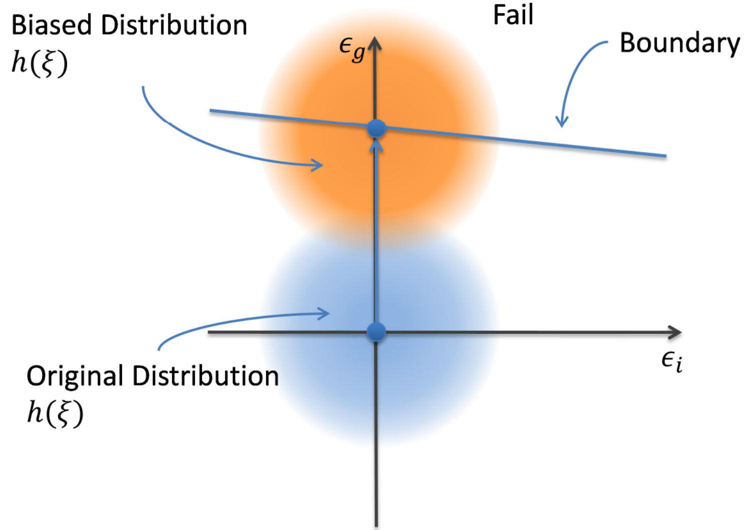

Fig. 6.12: Illustration of importance sampling. The random sample is drawn from the biased distribution $h(\xi)$, which has a significant number of points falling in the region of interest.

Following the model in (6.5), the original distribution is

$$g(\xi) = C \exp\left(-\frac{\sum_{i=1}^{m} \epsilon_i^2}{2}\right) \exp\left(-\frac{\epsilon_g^2}{2}\right), \tag{6.28}$$

and the biased distribution is

$$h(\xi) = C \exp\left(-\frac{\sum_{i=1}^{m} \epsilon_i^2}{2}\right) \cdot \exp\left(-\frac{\left(\epsilon_g - \epsilon_g^{\text{shift}}\right)^2}{2}\right), \tag{6.29}$$

where C is a normalization constant. The function $w(\xi)$ as defined in Eq. (2.7) is

$$w(\xi) = \frac{g(\epsilon)}{h(\epsilon)} = \frac{\exp\left(-\frac{\epsilon_g^2}{2}\right)}{\exp\left(-\frac{\left(\epsilon_g - \epsilon_g^{\text{shift}}\right)^2}{2}\right)}. \tag{6.30}$$

The shift amount $\epsilon_g^{\text{shift}}$ is found using the Newton-Raphson method, where the iteration follows the following formula:

$$\epsilon_{g(n+1)} = \epsilon_{g(n)} - \frac{D_{\text{critical}}\left(\epsilon_{g(n)}\right) - D_{\max}}{\frac{\partial D_{\text{critical}}\left(\epsilon_{g(n)}\right)}{\partial \epsilon_{g(n)}}}, \tag{6.31}$$

and the stopping criterion of the search for $\epsilon_g^{\text{shift}}$ is

$$\left|D_{\text{critical}}\left(\epsilon_{g(n)}\right) - D_{\max}\right| \leq 0.01 \text{ ps}. \tag{6.32}$$

Once $\epsilon_g^{\text{shift}}$ is found, hence the biased distribution $h(\xi)$ is determined, the failure probability is estimated by sampling at the biased distribution according to Eq. (2.7). It is shown in the next section that, with standard error k=5%, the total number of simulations required in importance sampling is around 10^3, almost independent of the probability that is being estimated.

6.3.4 Results and Discussion

We first show the efficiency of geometric programming and importance sampling, which in turn legitimize the idea of SGP which combines these two. SGP is applied to several sample problems showing the capability of solving yield-constrained digital circuit sizing problem with thousands of gates. Similarly, the simulations are done on a desktop with 2.4 GHz Intel Core 2 Quad processor and Matlab 7.4.0. The geometric programming problem is modeled and solved using the Matlab toolbox of MOSEK [18].

6.3.4.1 Geometric Programming Result

To test how the geometric programming formulation of the circuit sizing problem can be solved efficiently, we first apply the method to the cascaded circuit example used earlier in section 6.1.2 (Fig. 6.4). The problem being solved is (6.27) with f_{Box}=1. The number of stages is changed from 1 to 180, and the number of gates changes from 6 to 1080. As shown in Fig. 6.13, the run-time to solve the geometric programming problem (6.27) grows roughly linearly with the number of gates. But the exact run-time can depend heavily on the architecture of the circuit, and can also depend on how the problem is formulated and processed in the solver. It is clearly seen that the run-time is on the order of seconds even for a circuit with thousands of gates.

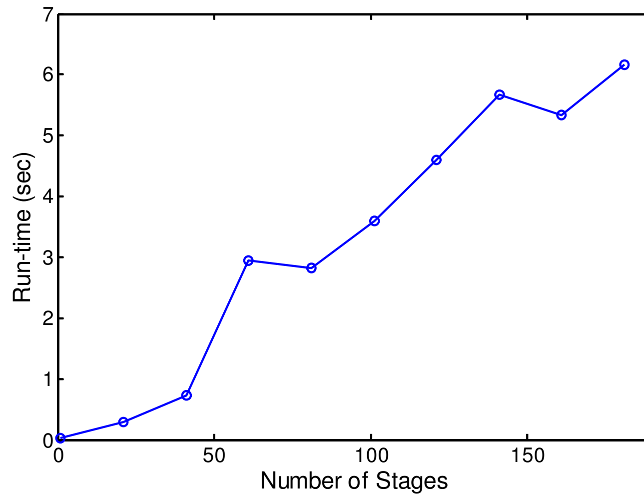

Fig. 6.13. GP Run-time vs. the number of stages in cascaded circuits. The run-time shows roughly a linear dependence of the size of the circuit. But the exact run-time depends on topology, formulation details.

6.3.4.2 Importance Sampling Result

Using Monte Carlo-based simulation to estimate probability, the accuracy is controlled by the standard error k as defined in Eq. (2.4). For the rest of this section, k is set to 5%. We apply the importance sampling method outlined in section 6.3.3 to a sample the circuit c17 in the ISCAS'85 benchmark circuit family [5]. The circuit under test is set so that the size of all its gates is unit (x=1). For the purpose of comparison, standard MC is also carried out. The parameter $D_{\max}$ is changed from 14.8 ps to 16.2 ps. As a result of the increase in $D_{\max}$, the failure probability is going to drop. Both importance sampling and standard MC give the same result as shown in the upper panel of Fig. 6.14. However, the number of simulation runs in Monte Carlo to achieve the accuracy set by k=5% grows exponentially, whereas the number of simulation runs in importance sampling to achieve the same accuracy shows much weaker, if any, dependence on $D_{\max}$, or equivalently on the probability being estimated. The number of simulation runs of importance sampling is roughly 1000. This can be understood by the following crude argument: the importance sampling biases the original distribution such that the effective P_f under the biased distribution is around 0.5, making the term $P_f/(1-P_f)$ close to one; to achieve the accuracy determined k=5%, the number of runs determined by (2.4) is on the order of 10^3. Even though the formula (2.4) is for pure MC, it still gives an approximate estimate of the required number of runs for importance sampling, since the method of importance sampling is indeed MC with a biased distribution.

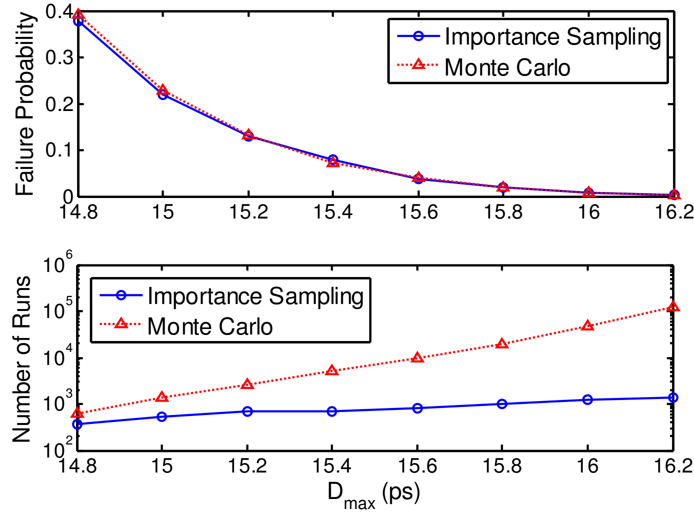

Fig. 6.14. Importance sampling and Monte Carlo simulation on unit-sized circuit c17. Upper panel: the estimated failure probability vs. $D_{\max}$. Lower panel: number of runs vs. $D_{\max}$. The standard error k is set to 5%.

6.3.4.3 SGP Result

Based on the results in 6.3.4.1 and 6.3.4.2, we observe that geometric programming can be solved very efficiently, and that the simulation runs required by importance sampling to estimate failure probability do not grow catastrophically large when the probability to be estimated is small. These observations show it is possible to combine these two elements in the iterative SGP method.

Due to the fact that the failure probability P_f is estimated via stochastic methods, thus the estimate is random in itself, it would be unreasonable to demand the final P_f to be infinitely close to δ. Therefore, the notion "close enough to δ" can be defined to be a region that is acceptable in practice. In our simulation, we use

$$P_f \in [\delta - 0.1\delta, \delta + 0.1\delta]. \tag{6.33}$$

We apply both SGP and the worst-case approach to an 8-bit adder [17]. The parameter δ is set to 0.01, corresponding to 99% yield, and $D_{\max}$ is set to 70 ps. After the optimization, a regular 50,000-run MC simulation is applied to the optimized circuits to visualize the delay distribution, where the random numbers are generated according to (6.5). The simulated histogram is shown in the upper panel of Fig. 6.15, and the optimized circuit area is shown in the lower panel of Fig. 6.15. It is clearly seen that worst-case approach ends up with a solution of which the failure probability is much less than necessary; this comes with an area penalty. SGP gives better result by pushing the delay distribution closer to the success/failure boundary, such that the failure probability is very close to δ; thus the yield constraint is active.

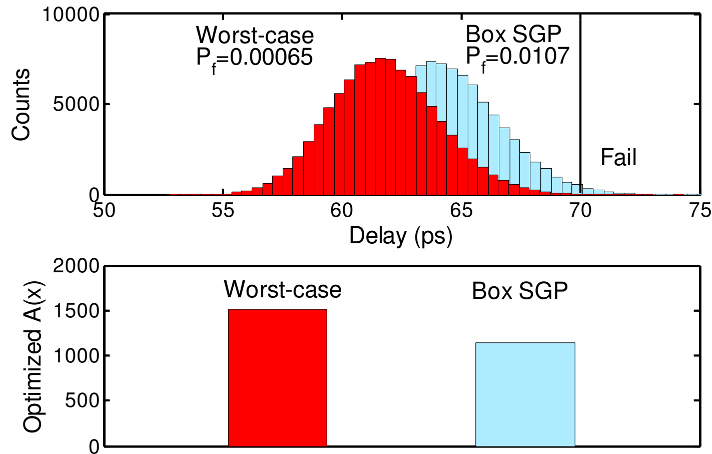

Fig. 6.15. SGP and worst-case approach on 8-bit adder. Upper panel: delay histogram from Monte Carlo simulation on the optimized circuits. Lower panel: optimized area given by different methods.

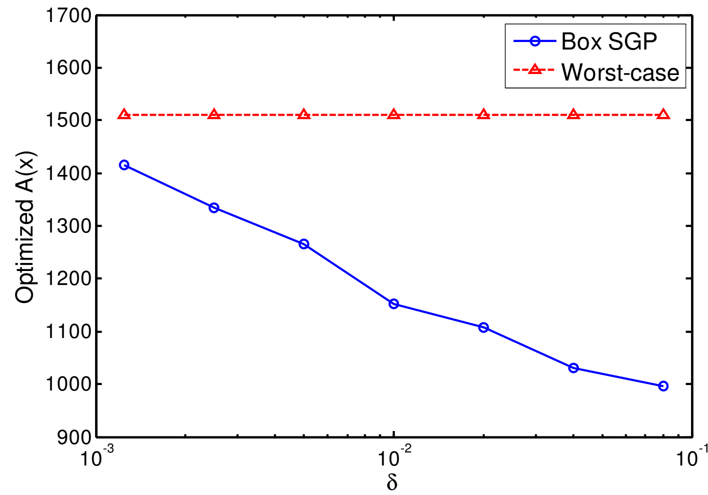

Fig. 6.16. Tradeoff curve of optimized area vs δ.

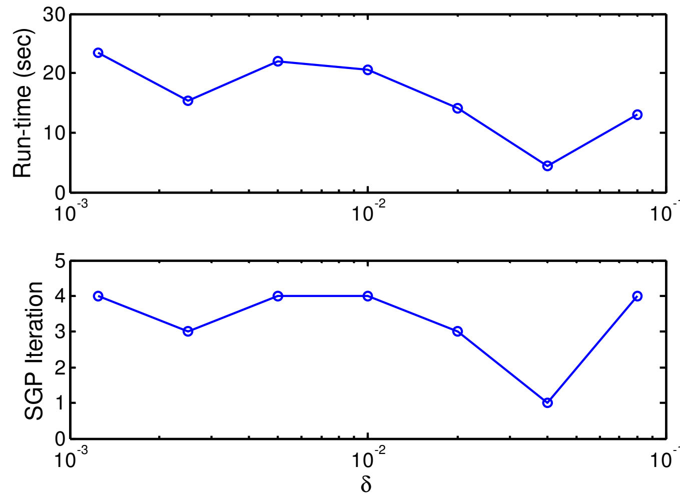

Fig. 6.17. Upper panel: runtime at different δ's. Lower panel: number of SGP iterations at different δ's.

To see how the method behaves at different δ's, we change δ from 10^{-3} to 10^{-1}, a typical tradeoff curve of the optimized area vs. δ is shown in Fig. 6.16. It is interesting to see how many SGP iterations are needed to reach the stopping criterion in (6.33). The total run-time and SGP iterations at different δ's are shown in Fig. 6.17. It is seen that the total run-time is only dozens of seconds for this circuit, and the number of SGP iterations never exceeds 4 throughout the entire δ range that we simulated.

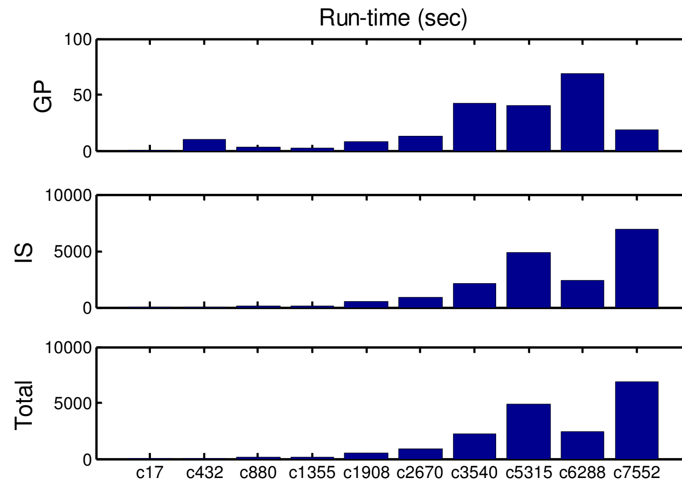

Fig. 6.18. Run-time breakdown for ISCAS'85 benchmark circuits.

We also applied the method to the ISCAS'85 benchmark circuit families, of which the largest circuit c7552 has 3,512 gates. The parameter δ is set to 0.01, and $D_{\max}$ is set to $N_{\text{logic}}\times 12.5$ ps, where N_{logic} is the logic depth of the circuit. The run-time is summarized in Fig. 6.18. The total time to solve the problem is no more than a few hours. Compared to geometric programming, the importance sampling simulation takes most of the run-time. This is expected since importance sampling involves large number of repetitions and is implemented completely in Matlab which may add significant overhead. A better implementation exploiting the nature of the repetitive runs or even using a parallel

architecture would have been far more efficient. However, even with the present code, the run-time of importance sampling can be well controlled as shown in Fig. 6.19. In Fig. 6.19, the number of simulation runs for importance sampling in each iteration is averaged and plotted for each benchmark circuit. The average number of simulation runs is around 10^3, which conforms to the argument in section 6.3.4.2 that the simulation runs should be at the order of 10^3 for k=5%. For the number of SGP iteration, as shown in Fig. 6.20, it is again below or equal to 4 for all the benchmark circuits we test.

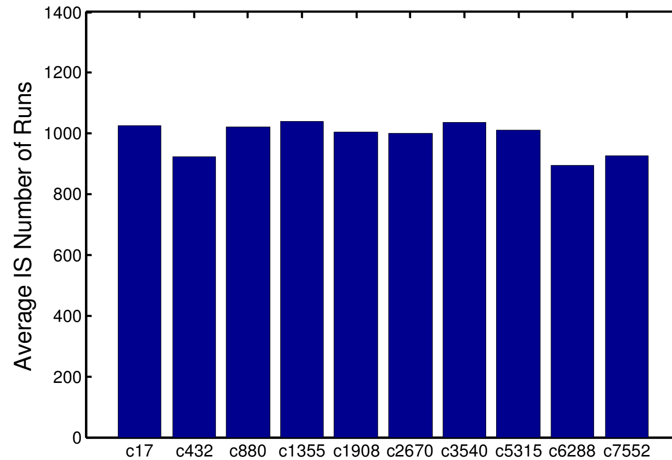

Fig. 6.19. Average number of runs for importance sampling at each iteration.

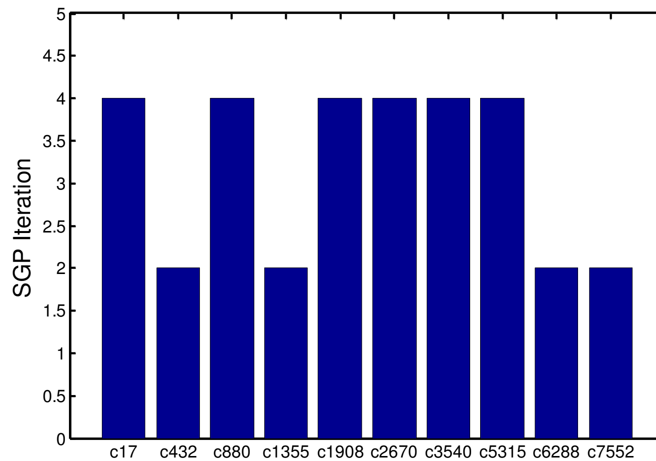

Fig. 6.20. The number of SGP iterations for ISCAS'85 benchmark circuits.

6.4 Summary

The first method, gradient projection, works directly on the non-convex form of the yield-constrained optimization problem. In order to reduce the number of expensive failure probability estimations, the method treats the objective function and the constraints separately. Although the method is flexible in dealing with different variability models, our

analysis shows that it suffers the high dimensionality, making it less useful when the number of gates is large.

The convex approximation method uses the three different approximation schemes: box approximation, ball approximation and Bernstein approximation. The box approximation is essentially the conservative worst-case method, whereas the ball approximation and Bernstein approximation can remove the conservativeness when the dimension of the problem is high. The approximated problems are convex and can be solved by off-the-shelf convex solvers. However, when the variability exhibits high correlation, neither ball nor Bernstein approximation gives satisfying result. This scenario is handled by the third method: the sequential geometric programming method.

The method of sequential geometric programming (SGP) consists of iterative usage of geometric programming and importance sampling. The fact that the yield constraint is handled by simulation makes the approach applicable to any variability model. The number of simulation runs in importance sampling is shown to be well controlled; few SGP iterations are generally needed to achieve practical convergence. The method is applied to 8-bit adder and also ISCAS'85 benchmark circuit families. All of them can be solved within reasonable amount of time using the proposed method.

6.5 References

[1] D. G. Luenberger, Y. Ye, "Linear and nonlinear programming," 3rd edition, Springer (2008)

[2] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, "Numerical recipes: the art of scientific computing," Cambridge University Press, (2007)

[3] Lecture notes, EE 364b, Stanford University http://www.stanford.edu/class/ee364b/lectures.html

[4] D.C. Montgomery, "Statistical quality control," 6th edition, Wiley (2008)

[5] http://www.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html

[6] S. P. Boyd, S.-J. Kim, D. D. Patil, and M. A. Horowitz, "Digital circuit optimization via geometric programming," Oper. Res., vol. 53, no. 6, pp. 899–932 (2005)

[7] A. Ben-Tal, L. El Ghaoui and A. Nemirovski, "Robust Optimization," Princeton University Press (2009)

[8] A. L. Soyster, "Convex programming with set-inclusive constraints and applications to inexact linear programming," Oper. Res. 1153-1157 (1973)

[9] A. Ben-Tal and A. Nemirovski, "Stable truss topology design via semidefinite programming,' SIAM J. on Optimization 7:4, 991-1016 (1997)

[10] A. Ben-Tal and A. Nemirovski, "Robust convex optimization," Math. Of Oper. Res. 23:4 (1998)

[11] L. El Ghaoui and H. Lebret, "Robust solution to least-square problems with uncertain data," SIAM J. of Matrix Anal. Appl. 18, 1035-1064 (1997)

[12] L. El Ghaoui, F. Oustry and H. Lebret, “Robust solutions to uncertain semidefinite programs,” SIAM J. on Optimization 9, 33-52 (1998)

[13] A. Ben-Tal and A. Nemirovski, “Robust solutions of linear programming problems contaminated with uncertain data,” Math. Program., Ser. A 88: 411-424 (2000)

[14] A. Nemirovski and A. Shapiro, “Convex approximations of chance constrained programs," SIAM J. on Optimization, vol. 17, no. 4, pp. 969-996 (2006)

[15] S. Boyd and L. Vandenberghe, "Convex optimization," Cambridge University Press (2004)

[16] H. Flanders, *Differential forms with applications to the physical sciences*, New York: Dover Publications, (1989)

[17] [Online] Available: http://www.pld.ttu.ee/testing/labs/adder.html

[18] [Online] Available: http://www.mosek.com

Chapter 7 Conclusions

7.1 Dissertation summary

Technology scaling necessitates close interaction between design and manufacturing. This thesis is focused on how to incorporate process variability in the design flow. More specifically, the thesis addresses the statistical verification and optimization of the integrated circuits.

Yield estimation is a crucial but expensive step in circuit verification. It is particularly challenging for SRAM cells since it has very low failure probability. We propose several methods intended to speed up the yield estimation. These methods include an extrapolation method using biased operating condition, extended extreme value theory, binary tree-based method and the partial least squares (PLS)-preconditioned importance sampling. Among these, we find the PLS-preconditioned importance sampling is the most suitable yield estimation method for SRAM failure analysis. It is shown to have high stability, and is at least 10 times faster than other importance sampling-based methodology. The method of extrapolation based on biased operating condition is not able to achieve the same level of efficiency, but it has potential applications when experimental measurements are involved. This is a unique advantage among all the proposed methods. The binary tree-based method is extremely fast for low dimensional problems, but suffers severely from the curse of dimensionality. Six variables are shows to be too many to justify the usage of the method, making the method not very useful in circuit yield analysis.

The corner models are used as a crude version of the yield estimation in circuit design. But current corner models cannot deal with local variability and can be misleading for certain performances. Customized corners are therefore needed. We find the shift-point used in PLS-preconditioned importance sampling is a good candidate of such customized corners. This hypothesis is verified using the ISCAS'85 benchmark circuits. But the PLS-based method is useful only when the circuit performances are roughly a monotonic function of the process parameters. When the performance function is more complicated, such as in analog circuit design, we propose to use nonlinear classifier-based method or the direction optimization method. The former is fast, but the corner points generated may fail. The direct optimization method can extract the desired customized corner points, albeit at a high computational cost. Depending on the requirement and computation budget, one can choose one method or the other.

Optimizing the circuit under the influence of variability is traditionally within the framework of worst-case corners. These methods lead to pessimistic design in general. To remove the pessimism, one needs to solve the yield-constrained optimization problem. But the presence of the failure probability calculation in the constraint makes it hard to solve. We propose three methods to solve the yield-constrained circuit optimization problem. The first method separates the treatment of objective function and constraints using gradient projection. This method has high flexibility in terms of the formulation, but is problematic

when the scale of the problem becomes large. The second method is based on the convex approximation idea from the most recent development in robust convex optimization. It can efficiently solve the yield-constrained problem if the variability across the circuit is independent. But when they are highly correlated, these convex approximation methods will give rise to very pessimistic solutions. The third method is a combination of the parameterized version of the convex approximation and fast yield estimation. It is able to accommodate a large variety of variability model because the yield is handled by simulation, and at the same time it is very efficient thanks to the convex nature of the problem being solved at every iteration step. The method is shown to be capable of dealing with circuits with thousands of gates.

7.2 Future research directions

Based on this work, several research directions can be pursued.

First, the extraction of customized corner models can be improved. The proposed methods in this thesis provide a solution to this problem, but the solution is not optimal and cannot be used as a generic methodology for all circuit applications. Although the direct optimization method can be used everywhere in theory, its computational cost for large circuits is prohibitive. Alternatively, one can exploit the nonlinear extension of the PLS regression techniques [1]. It can take advantage of the speed of regression to tackle the situation with highly nonlinear performance functions. One can identify a worst-case hyper-surface based on nonlinear PLS regression. But the conversion from the hyper-surface to discrete corner points needs further investigation.

Second, the sequential method is based on the box-approximation of the original yield-constrained problem. It is possible to incorporate other convex approximations and make it scalable. Moreover, one can use multiple parameters to describe the shape of the perturbation set. These possible extensions can allow for even better result to be reached, but they may also incur extra burden on the outer loop since more than one parameters need to be tuned and the simple line-search cannot be used.

The sequential optimization method should be able to be applied to other circuits, in particular the optimization of an SRAM cell [2]. It takes the industry large amount of manpower to design the SRAM cell right at the desired margin. It would be cost effective for the memory design company if this process can be automated through optimization formulation. The sequential optimization method introduced in this thesis should be able to be applied to this problem with some modification. That way, memory can also enjoy the automated design with the exact parametric yield.

7.3 References

[1] G. Baffi, E. B. Martin and A. J. Morris, “Nonlinear projection to latent structures revisited (the neural network PLS algorithm),” Computers and chemical engineering, 23, 1293-1307 (1999)

E. Morifuji, D. Patil, M. Horowitz and Y. Nishi, “Power optimization for SRAM and its scaling,” IEEE Trans. Electron Devices vol. 54, No. 4 (2007)