Foundations and Trends® in
Electronic Design Automation
Vol. 1, No. 4 (2006) 331–480
© 2007 X. Li, J. Le and L. T. Pileggi
DOI: 10.1561/1000000008

Statistical Performance Modeling and Optimization

Xin Li[1], **Jiayong Le**[2] **and Lawrence T. Pileggi**[3]

[1] *Department of ECE, Carnegie Mellon University, Pittsburgh, PA 15213, USA, xinli@ece.cmu.edu*
[2] *Extreme DA, 165 University Avenue, Palo Alto, CA 94301, USA, kelvin@extreme-da.com*
[3] *Department of ECE, Carnegie Mellon University, Pittsburgh, PA 15213, USA, pileggi@ece.cmu.edu*

Abstract

As IC technologies scale to finer feature sizes, it becomes increasingly difficult to control the relative process variations. The increasing fluctuations in manufacturing processes have introduced unavoidable and significant uncertainty in circuit performance; hence ensuring manufacturability has been identified as one of the top priorities of today's IC design problems. In this paper, we review various statistical methodologies that have been recently developed to model, analyze, and optimize performance variations at both transistor level and system level. The following topics will be discussed in detail: sources of process variations, variation characterization and modeling, Monte Carlo analysis, response surface modeling, statistical timing and leakage analysis, probability distribution extraction, parametric yield estimation and robust IC optimization. These techniques provide the necessary CAD infrastructure that facilitates the bold move from deterministic, corner-based IC design toward statistical and probabilistic design.

1 Introduction

As integrated circuit (IC) technologies continue shrinking to nanoscale, there is increasing uncertainty in manufacturing process which makes it continually more challenging to create a reliable, robust design that will work properly under all manufacturing fluctuations. Large-scale process variations have already become critical and can significantly impact circuit performance even for today's technologies [14, 72, 73, 100]. Figure 1.1 shows the relative process variations (3σ/mean) predicted by the International Technology Roadmap for Semiconductors (ITRS) [100]. These large-scale variations introduce numerous uncertainties in circuit behavior and make it more difficult than ever to achieve a robust IC design.

In addition, when we consider some of the promising new device structures (e.g., carbon nano-tube [8, 34], FinFET [20], etc.) that have been recently proposed to maintain the aggressive pace of IC technology scaling, it is apparent that applying them to high-volume production will be a challenging problem due to the manufacturing uncertainty, or even their likelihood of failure. While it has already become extremely difficult to reliably manufacture nano-scale devices and achieve high product *yield* (defined as the proportion of the manufactured chips

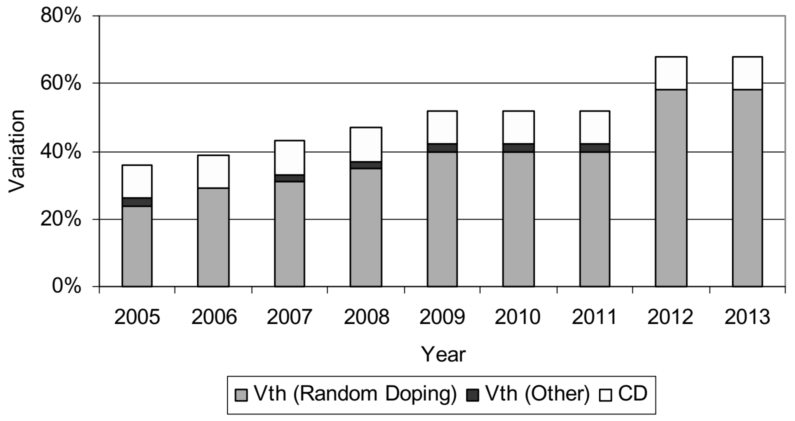


Fig. 1.1 Relative process variations (3σ/mean) predicted by [100].

that function correctly) with today's technologies, it would be almost impossible to do so affordably with tomorrow's technologies using existing deterministic design methodologies. For this reason, a paradigm shift in IC design is required to simultaneously improve circuit performance and product yield when using manufacturing technologies that present significant uncertainty.

The yield loss in a manufacturing process can be classified into two broad categories: *catastrophic* (due to physical and structural defects, e.g., open, short, etc.) and *parametric* (due to parametric variations in process parameters, e.g., V_{TH}, T_{OX}, etc.). As process variations become relatively large due to technology scaling, parametric yield loss is becoming increasingly significant at 90 nm technologies and beyond. Therefore, we focus on the parametric yield problem in this paper. We will review a number of recently-developed techniques that handle large-scale process variations at both transistor and system levels to facilitate affordable statistical integrated circuit design. Especially, we will focus on the following two questions:

- *When should process variations be considered?* Ideally, we want to take into account process variations in the earliest design stage. However, this strategy may not be necessary and/or efficient in practice. During early-stage system-level design, many simplified models must be used to make the large-scale design problem tractable. The errors of these system-level models may be comparable to, or even larger than, the uncertainties caused by process variations. In such cases, it is not meaningful to model and analyze process variations at system level. As the design moves from system level down to circuit level, more accurate circuit-level models become available. We should start to consider process variations at the stage where circuit models are sufficiently accurate and process variations become the dominant uncertainties that impact performance.

- *How should process variations be considered?* For example, the simplest way to model process variations is to define a number of process corners. That is, every process parameter is assigned with a lower bound and an upper bound, and the best-case and worst-case circuit performances are computed by enumerating all possible combinations of the extreme values of process parameters. The aforementioned corner model is simple; however, such a corner-based approach may result in large error, since it completely ignores the correlation among different process parameters. In addition, it is not guaranteed that the best/worst-case performance always occurs at one of these corners. An alternative approach to model process variations is to use statistical device models where process parameters are modeled as random variables. (More details on statistical device models can be found in Chapter 2.) The statistical device model is much more accurate, but also expensive, than the traditional corner model.

One of the major objectives of this paper is to review and compare different statistical IC analysis and optimization techniques, and analyze their trade-offs for practical industrial applications. The following topics will be covered in this paper:

- *Sources of process variations and their models.* We will briefly review both front-end of line (FEOL) variations and back-end of line (BEOL) variations in Chapter 2. Several techniques for variation characterization and modeling will be presented for both device-level and chip-level applications.

- *Transistor-level statistical methodologies.* In Chapter 3, we will discuss and compare a number of transistor-level statistical modeling, analysis and optimization techniques. In particular, the following topics will be covered: Monte Carlo analysis, response surface modeling, probability distribution extraction, parametric yield estimation, and robust transistor-level optimization. Several recently-developed methodologies, including projection-based performance modeling (PROBE) and asymptotic probability extraction (APEX), will be described in detail.
- *System-level statistical methodologies.* Most system-level statistical analysis and optimization techniques utilize a hierarchical flow to partition the entire system into multiple small blocks such that the large-size problem becomes tractable. In Chapter 4, we will discuss a number of system-level statistical methodologies that have been recently proposed. In particular, we will focus on the statistical timing and leakage problems for large-scale digital systems.

Finally, we will conclude and propose several possible areas for future research in Chapter 5.

2

Process Variations

Process variations are the deviations from the intended or designed values for the structural or electrical parameters of concern. In this paper, we focus on the *parametric variations* due to the continuously varying structural or electrical parameters. For modern semiconductor manufacturing processes, transistors and other active devices are first fabricated on top of semiconductor substrate. After that, metal layers are deposited to connect transistors and supply power. Based on the manufacturing steps, parametric variations can be classified into two broad categories: front-end of line (FEOL) variations for devices and back-end of line (BEOL) variations for interconnects. In this chapter, we briefly review the sources of these variations and their models.

2.1 Front-End of Line (FEOL) Variations

FEOL variations mainly refer to the variations at device level. The major sources of FOEL variations consist of transistor gate length and gate width variations, gate oxide thickness variations, doping-related variations, etc. After transistors are fabricated, these variations can be observed by measuring the corresponding device characteristics, including drain-source current (I_{DS}), threshold voltage (V_{TH}), gate leakage current (I_{Gate}), etc. For example, poly critical dimension (CD) variations can change I_{DS}, and gate oxide thickness variations and doping-related variations can change V_{TH}.

In nano-scale IC technologies, transistor gate length, or poly critical dimension, can significantly vary due to its small feature size. The sources of CD variations typically include exposure system variations, mask errors, and resist effects. Exposure system variations may have optical, mechanical or illuminative explanations [108]. Optical performance (e.g., flare) can introduce non-uniform exposure dose. Mechanical performance (e.g., vibrations during reticle scanning and/or wafer scanning) can cause focus variations. Illumination performance is mainly related to the polarization control of the laser.

As shown in Figure 2.1, mask errors can be puncture, burr, blotch, mask bias, etc. [118] Since mask errors affect all the dies within the same mask, the variations caused by mask errors are systematic. Furthermore, regions within a design with relatively low aerial-image contrast will be subject to larger amplification of mask errors.

Resist effects result in line edge roughness (LER) [7], as shown in Figure 2.2. LER has caused little worry in the past since the critical dimensions of MOSFETs were orders of magnitude larger than the roughness. However, as the aggressive technology scaling moves into nanometer regime, LER does not scale accordingly and becomes an increasingly large fraction of the gate length.

Along with gate length, gate width also shows substantial variations. Since many transistors in a practical design typically have much

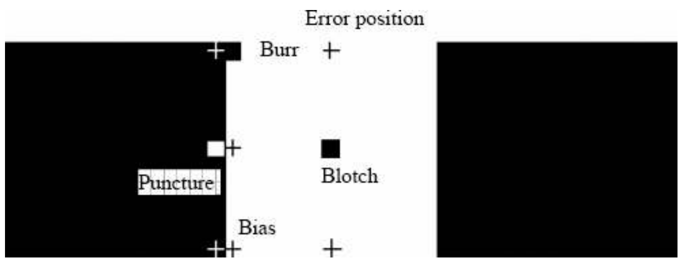


Fig. 2.1 Examples of mask error (showing puncture, burr, blotch, and mask bias).

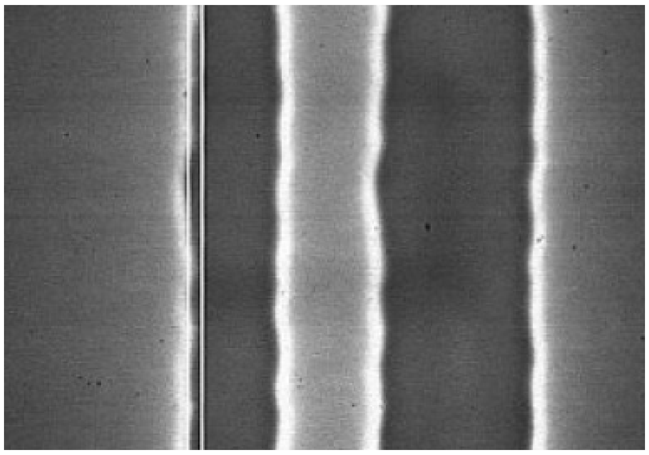


Fig. 2.2 A typical LER effect on poly.

bigger gate widths than their gate lengths, gate width variations are not as dominant as gate length variations. However, for minimal-width transistors, gate width variations can significantly impact transistor performance.

Part of the transistor gate width variations comes from shallow trench isolation (STI) polish [35]. STI is a method used for electrical isolation of microstructures during IC manufacturing. It is a replacement of the old Local Oxidation of Silicon (LOCOS) for technologies beyond $0.35\,\mu$m. STI uses a nitride mask to protect transistor regions and expose isolation regions. It etches trenches into a wafer, and then fills the trenches with oxide. Afterwards, it uses chemical mechanical polishing (CMP) to polish away any excess oxide. During the etching and filling step, some parts of the device area might be consumed (as shown in Figure 2.3) and, therefore, the effective width of the poly becomes smaller. This effect is especially significant for narrow-width transistors.

Another source of gate width variations is from the rounding effect due to patterning limitations, as shown in Figure 2.4. Rounding affects both transistor gate length and gate width. In Figure 2.4, the effective gate width for the left-most transistor is increased by the rounding

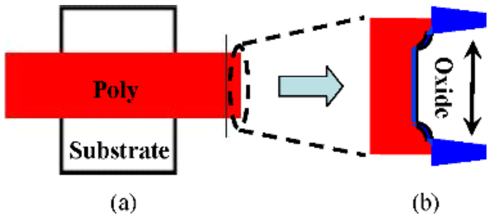


Fig. 2.3 Top view of shallow trench isolation. (a) Top view of the drawn poly. (b) Right edge of the ploy where part of its area is consumed by the etching and filling step.

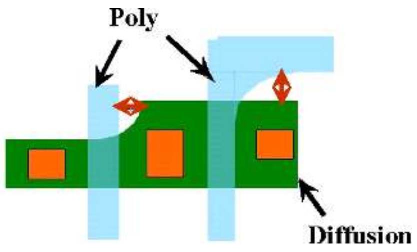


Fig. 2.4 Example of rounding effect.

effect at diffusion. Meanwhile, the effective gate length is increased by the rounding effect at poly for the right-most transistor. To mitigate the problem, strict design rules must be applied so that the rounding effect can be minimized.

Threshold voltage is another important device parameter that exhibits significant variations. Threshold voltage variations consist of two major components: (1) die-to-die variations and (2) random within-die variations. The die-to-die variations mainly come from wafer-level non-uniformity (e.g., non-uniform temperature distribution during thermal oxidation). On the other hand, the random within-die variations are mainly caused by random channel dopant fluctuations and poly dopant fluctuations. In today's leading-edge manufacturing technologies, there are only about 500 dopant atoms in one MOSFET channel. The fluctuation of the number of dopant atoms affects the threshold voltage of each device independently, and the resulting threshold voltage variation can be approximated as a zero-mean Normal distribution whose standard deviation is inversely proportional to the square root of the effective gate width and the effective gate length [65, 84, 111]:

$$\sigma\left(\Delta V_{\mathrm{TH}}^{\mathrm{Random}}\right) \sim \frac{1}{\sqrt{W_{\mathrm{Eff}} \cdot L_{\mathrm{Eff}}}}, \tag{2.1}$$

where W_{Eff} and L_{Eff} are the effective gate width and length, respectively.

2.2 Back-End of Line (BEOL) Variations

BEOL variations refer to the variations of metal interconnects. For example, metal width, metal thickness, and inter layer dielectric can have both systematic and random variations. Subsequently, the electrical parameters of interconnects including both resistance and capacitance exhibit corresponding variability.

At 90 nm technology node, metal thickness variations can be up to 30%~40% of the nominal thickness. Such large-scale variations can significantly impact the resistance and capacitance of a wire and lead to severe timing and/or signal integrity issues. Metal thickness variations depend on wire width and metal density, and have short range interactions [82]. Thickness variations are influenced by process maturity, location of the metal layer in process stack and other process operating conditions. One important source of metal thickness variations is the copper loss during chemical mechanical polishing (CMP). Such copper loss is a function of metal pattern density (erosion) and metal width (dishing) as shown in Figure 2.5.

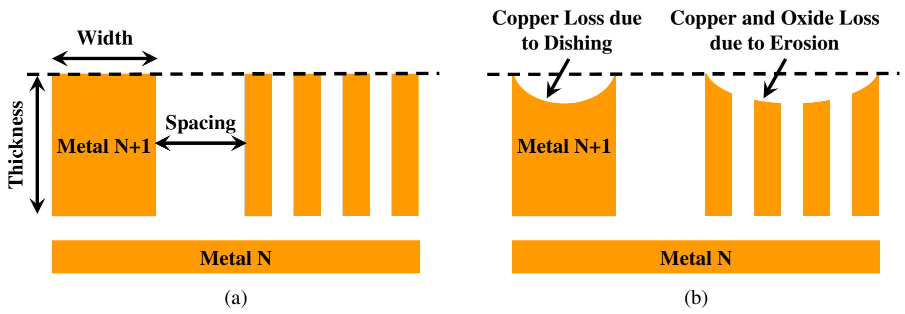

Width
Thickness
Spacing
Metal N+1
Metal N
(a)
Copper Loss due to Dishing
Copper and Oxide Loss due to Erosion
Metal N+1
Metal N
(b)

Fig. 2.5 Metal thickness variations due to chemical mechanical polishing. (a) Ideal case after copper CMP. (b) Realistic case after copper CMP.

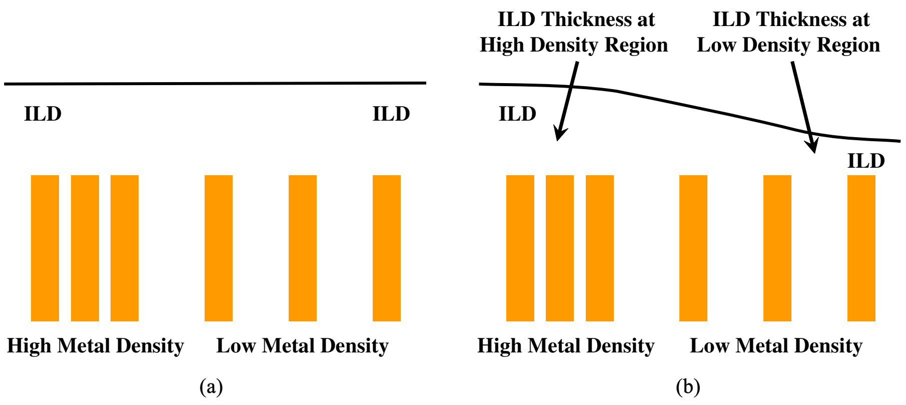


Fig. 2.6 ILD thickness variations due to chemical mechanical polishing. (a) ILD thickness after oxide deposition but before oxide CMP. (b) ILD thickness after oxide CMP.

In addition to metal thickness, inter-layer dielectric (ILD) thickness is another important process parameter that varies significantly in nano-scale technologies. ILD is the dielectric that separates two adjacent metal layers and it strongly impacts the parasitic capacitance of metal interconnects. ILD thickness variations come from the fluctuations of the oxide CMP polishing rate that are influenced by the lower-layer metal density [110]. It can be modeled as a function of the underlying metal pattern density and the pad planarization length, as shown in Figure 2.6.

The third major component of BEOL variations is the metal width/spacing variation. Metal width/spacing variations are primarily due to lithography effects. The width variation of a critical wire segment (also known as selective process biasing) is primarily determined by the metal width and the spacing to its neighbors. Process engineers typically measure width variations in silicon and create a two-dimensional table to model width variations as a function of metal width and spacing. This table is typically incorporated into the process-technology file that can be further used by the capacitance-and-resistance extraction engine to account for the effect of width variations.

2.3 Variation Characterization

Modern IC manufacturing typically involves hundreds of, or even thousands of, steps, and, therefore, the true causes of manufacturing variations are highly complicated. For example, they can be caused by fluctuations in equipment conditions, fluctuations in wafers, consumable material characteristics, layout and topography interactions with the process, and their combinations. Generally, process variations manifest themselves in both temporal and spatial manner. In this chapter, we classify process variations into different categories based on their statistical or geometrical properties. We also briefly describe the test structures for variation characterization. Due to the difference in their underlying nature, variations in different categories typically require different test structures to characterize them.

2.3.1 Statistical Categorization

From statistical point of view, process variations can be classified into two broad categories: *systematic variations* and *random variations*. Systematic variations refer to the variation components that can be attributed to a specific deterministic cause. It is usually dependent on the component position in the die and its surrounding properties such as metal density. For example, gate length and width variations contain systematic components that mainly come from the fluctuations of focus and exposure of the lithography system for different layout patterns. Part of metal thickness variations is caused by CMP and, therefore, is systematic.

Systematic variations are deterministic and can be predicted by process simulations (e.g., lithograph simulation). However, accurately predicting systematic variations may not be feasible for many practical designs because of the following two reasons. First, accurate process simulations can be extremely expensive and may not be applicable to large-size circuits. Second, a lot of design information may not be available at the earlier design stages, further limiting the accuracy of process simulation. For example, placement and routing information is not available at schematic design phase and, therefore, it is difficult to predict systematic variations for schematic design. From this point of view, systematic variations can be viewed as the repeatable but unpredictable modeling error due to the lack of computational resource and/or design information.

On the other hand, random variations refer to the variation components that correspond to unrepeatable uncertainties due to insufficient process control and intrinsic fluctuations. An important example here is the threshold voltage variation caused by doping fluctuation. In modern IC technologies, there are only about a few hundred dopant atoms in one MOSFET channel. It would be extremely difficult, if not impossible, to precisely count the dopant atoms during manufacturing. Therefore, the fluctuations of the number of dopant atoms introduce random threshold voltage variations for MOSFETs.

As IC technologies are scaled to smaller feature sizes in the future, both systematic and random variations may further increase. To reduce systematic variations, IC designers start to utilize restricted layout patterns for both digital circuits [48, 86] and analog circuits [121, 123]. Random mismatches, however, can hardly be controlled by circuit layout. As feature sizes become smaller and each transistor contains fewer atoms in its gate channel, random mismatches are expected to become the dominant variation component in 65 nm technologies and beyond.

2.3.2 Geometrical Categorization

According to the spatial scale of their manifestation, process variations can be classified into four different levels: lot-to-lot, wafer-to-wafer (within-lot), die-to-die (within-wafer), and within-die [109]. The first three variations are also called the *inter-die variations*: they model the common/average variations across the die. The within-die variations are also called the *intra-die variations* or *on-chip variations*: they model the individual, but spatially correlated, local variations within the same die. Intra-die variations further consist of two different components: correlated variations and *independent mismatches*. The spatial correlation of intra-die variations is distance-dependent. Such a spatial correlation occurs mainly because systematic variations are modeled as random variables [109]. Figure 2.7 summarizes the geometrical categorization of process variations.

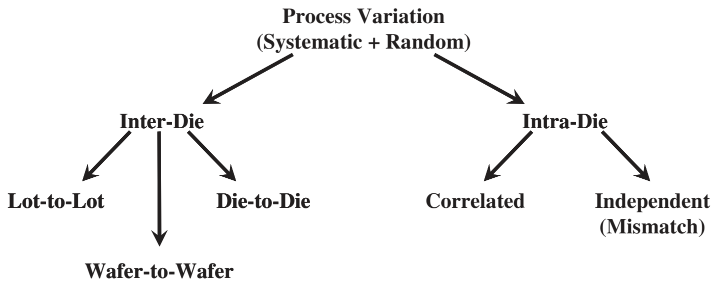


Fig. 2.7 Categorization of process variations according to geometrical scale.

2.3.3 Test Structures for Variation Characterization

Device parameters (e.g., gate length, gate width, etc.) can be directly measured from the chip. However, such measurement is typically expensive and sometimes even impossible. To avoid directly physical/mechanical measurement, various test structures are designed for process characterization. These test structures can translate the device parameters of interest to a set of electrical characteristics (e.g., ring oscillator frequency, I–V curve) that are easy to measure.

Ring oscillator is an important test structure that has been widely used to characterize inter-die variations and long-range correlated intra-die variations [11, 77, 79, 80]. The performance of interest of a ring oscillator is its oscillation frequency. The digital output of a ring oscillator can be easily delivered out of chip and its frequency can be accurately measured without any distortion. For this reason, ring oscillators can be easily distributed at both wafer level and die level. The frequency information of different ring oscillators is then collected to extract the spatial map of process variations.

Most ring oscillators, however, are not sensitive to random mismatches, since the random per-transistor variations can be averaged out when cascading a large number of inverter blocks to create a ring oscillator. Recently, transistor array has been proposed as a promising approach for mismatch characterization [75]. The basic idea here is to have a great number of regular transistors and measure the I–V curve for each of them. A major design challenge for transistor array is to build high-performance analog circuit to accurately deliver analog voltage and current signals out of chip for measurement through a limited number of I/O pins.

2.4 Variation Modeling

To facilitate statistical analysis and optimization of integrated circuits, variation models must be carefully created to abstract and approximate the physical phenomenon during IC manufacturing. The accuracy of these models directly impact the quality of the final statistical analysis and optimization results at chip level. In this chapter, we briefly review the statistical models at both device level and chip level. We do not explicitly distinguish systematic and random variations and statistically model both of them as random variables. It should be noted, however, that systematic variations are not truly random. They are modeled statistically, because accurately predicting their exact values is not feasible due to the lack of computational resource and/or design information.

2.4.1 Device-Level Variation Modeling

Commercial IC foundries started to incorporate manufacturing variation information into their device models long time ago. The early variation-aware device models are mostly corner-based. Namely, in addition to offering device models for nominal process conditions, additional models are provided for a number of process corners. Basic process corners, for example, include FF (fast PMOS and fast NMOS), FS (fast PMOS and slow NMOS), SF (slow PMOS and fast NMOS), and SS (slow PMOS and slow NMOS).

While corner models have been widely used in the past, they suffer from a few major limitations that become increasingly critical in nanoscale technologies [71]. First, it is not guaranteed that the worst-case performance always occurs at one of these corners. Different circuits with different topologies and performance metrics typically show different sensitivities with respect to process variations and, therefore, reach the worst case at different process corners. In other words, the "realistic" worst-case corner should be topology-dependent and performance-dependent. Figure 2.8 shows the relative performance variations for an

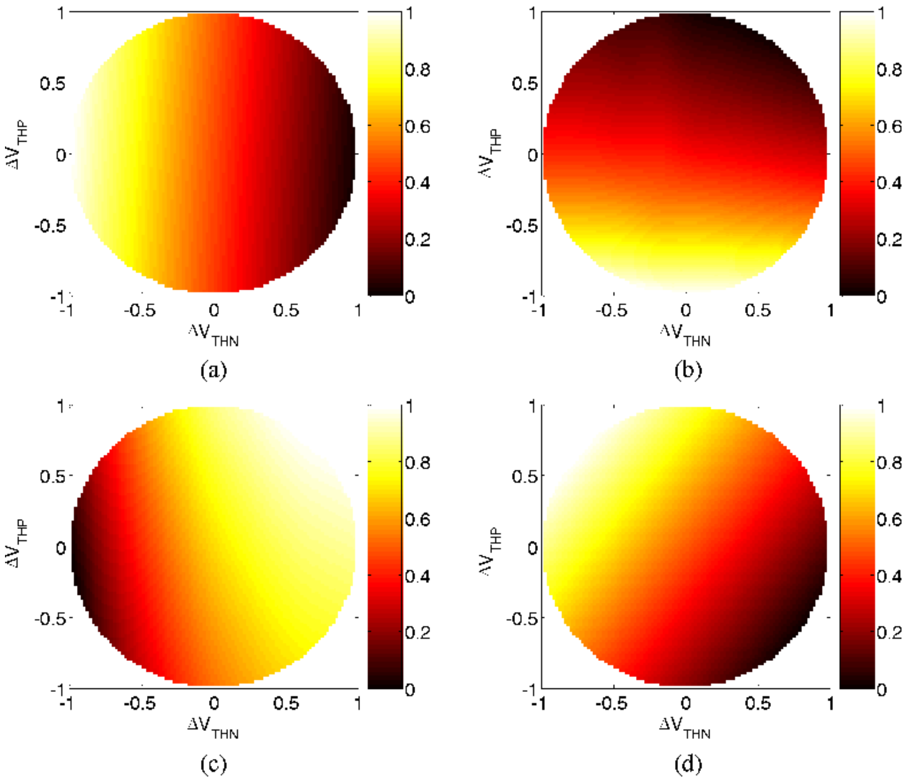


Fig. 2.8 Performance variations (normalized to [0, 1]) of an industrial voltage-controlled oscillator design in a commercial $0.13\,\mu$m CMOS process. (a) Center frequency. (b) Gain. (c) Output voltage swing. (d) Power.

industrial voltage-controlled oscillator design in a commercial $0.13\,\mu$m CMOS process. The color maps in Figure 2.8 indicate the performance sensitivities with respect to inter-die V_{THP} (threshold voltage of PMOS) and V_{THN} (threshold voltage of NMOS) variations. Studying Figure 2.8, one would notice that the performance gradients are completely different for the four performance metrics of interest.

Second, corner models are typically created for inter-die variations only. Ignoring intra-die variations can result in either pessimistic or erroneous results, depending on the circuit topology and performance metric of interest. Figure 2.9(a) shows a simple digital path that consists of several logic gates. Corner models assume that all gate delays in this path are fully correlated and they reach the worst case

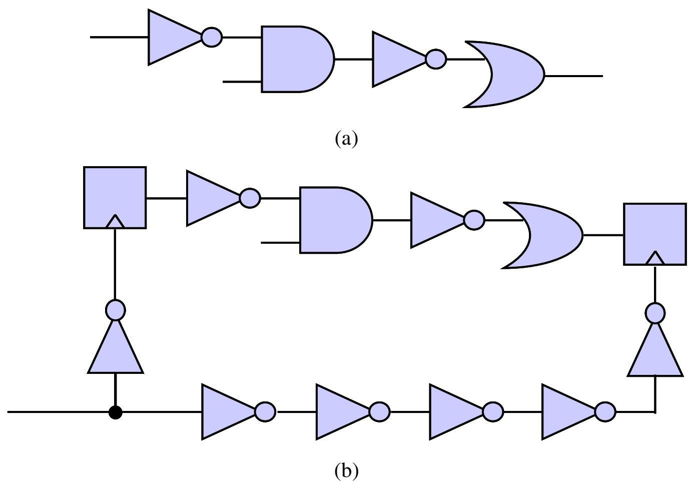


Fig. 2.9 Ignoring intra-die variations can result in either pessimistic or erroneous results. (a) A simple digital path (pessimistic results). (b) A simple sequential digital circuit with D flip–flops (erroneous results).

simultaneously, thereby yielding pessimistic results. Such pessimistic results may lead to unnecessary over-design which implies large chip area and/or high power consumption for digital circuits.

While Figure 2.9(a) gives an example of pessimism, Figure 2.9(b) shows another practical case where corner models yield erroneous results if they are not correctly used. In this example, the clock path delay (D_{CLK}) must be greater than the data path delay (D_{DATA}) to avoid setup timing failure. Since D_{CLK} and D_{DATA} are not fully correlated due to intra-die variations, simulating both D_{CLK} and D_{DATA} at the same corner (e.g., FF or SS) incorrectly assumes that D_{CLK} and D_{DATA} perfectly track each other (i.e., simultaneously increase or decrease), thereby yielding erroneous results. In this example, the true worst case occurs when the clock path is fast and the data path is slow.

In addition to the digital circuit examples in Figure 2.9, intra-die variations are even more important for analog circuits. Many basic analog building blocks (e.g., differential pair, current mirror, switched-capacitor amplifier, etc.) rely on device matching to achieve the correct analog functionality. In other words, these circuits are designed to be robust to inter-die variations, but they are extremely sensitive to intra-die variations (particularly, device mismatches). In these cases, intra-die variations must be carefully considered for worst-case performance analysis.

There have been various ways to improve corner models, e.g., replacing manufacturer-defined corners by application-specific corners [101] (also called user-defined corners). The basic idea here is to look at the performance change with respect to process variations to determine the specific location in the random variation space that yields the worst-case performance value, as shown in Figure 2.10. In this case, intra-die variations can also be incorporated to determine the worst-case corner. Considering intra-die variations for corner extraction, however, will increase the problem complexity, as additional random variables must be utilized to model intra-die variations, thereby resulting in a higher-dimensional variation space.

Given a fixed circuit topology and a number of pre-defined performance metrics, the aforementioned application-specific corners are not fixed; instead, they depend on the design variable values of the circuit (e.g., transistor sizes, bias current, etc.). In practice, a great number of corners may be required to capture the worst-case performance over a reasonable range of design variable values. For example, it is not uncommon to end up with more than 1000 application-specific corners for an industrial mixed-signal design in 65 nm technologies. Such

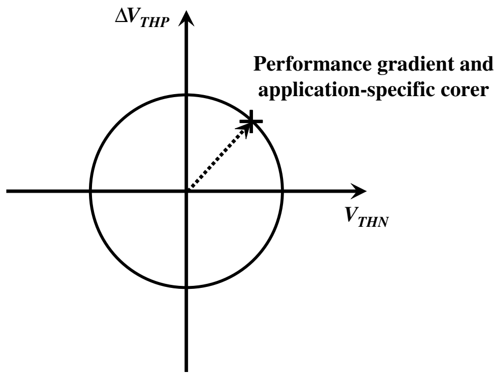


Fig. 2.10 Extract application-specific worst-case corner based on performance gradient.

```
vary ε1 dist = gauss std = 1
vary ε2 dist = gauss std = 1

model NMOS b~4
+ type = n
+ tox = 4e-9 + 1e-10*ε1 – 1.3e-10*ε2 + ...
+ vth0 = 0.6 + 0.24*ε1 + 0.3*ε2 + ...
+ ...
```

Fig. 2.11 A simple example of statistical device model where ε_1 and ε_2 are used to model inter-die variations.

a large number of corners can result in expensive simulation cost during verification.

To address the fundamental limitations of corner models, many commercial IC foundries start to provide statistical device models [104]. In these models, a number of random variables are defined to capture both inter-die and intra-die variations and the device model parameters are represented as functions of these random variables. Taking Figure 2.11 as an example, two independent random variables ε_1 and ε_2 are extracted by principal component analysis (more details in Section 2.4.3) to model the correlated inter-die variations of T_{OX} and V_{TH}. It should be noted that an industrial statistical device model can be much more complicated than the simple example in Figure 2.11, as a great number of pre-defined random variables must be utilized to capture all process variations with spatial correlation.

2.4.2 Chip-Level Correlation Modeling

As shown in Figure 2.7, process variations consist of three major components: inter-die variations, on-chip long-range correlated variations, and independent device mismatches. Good statistical device models should accurately capture all these three components. At chip level, the long-range correlated variations are most difficult to model. The challenging problem here is how to use the silicon data measured by test structures to accurately extract the spatial correlation for the entire chip that consists of millions of transistors. To solve this problem, the following two questions must be properly addressed: (1) What is the correct correlation function that is distance-dependent? (2) How can we efficiently and robustly handle the large problem size?

One useful approach for modeling correlated intra-die variations is to partition the entire die into a number of grids [18], as shown in Figure 2.12. The intra-die variations in the same grid are fully correlated, while those in close (far-away) grids are strongly (weakly) correlated. Taking Figure 2.12 as an example, the gates a and b are in the same grid and, therefore, their process parameters are fully correlated. The gates a and c lie in two neighboring grids and their process parameters are strongly correlated. The gates a and d sit far away from each other and their process parameters are weekly correlated.

The authors in [2] proposed another hierarchical approach for modeling correlated intra-die variations. The key idea is to hierarchically divide the entire chip into a number of regions using a multi-level quad-tree partition, as shown in Figure 2.13. At each level i, the die area is partitioned into 2^i by 2^i rectangles. The 0th level, for example, contains one rectangle only that covers the entire chip. An independent random variable ε_{ij} is assigned to each region $(i,\ j)$ to model a portion of the total intra-die variations. The overall variation of a gate k is expressed as the sum of the individual components ε_{ij}

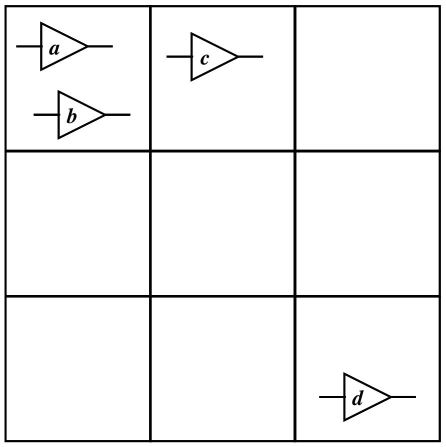


Fig. 2.12 Grid-based model for spatial correlation.

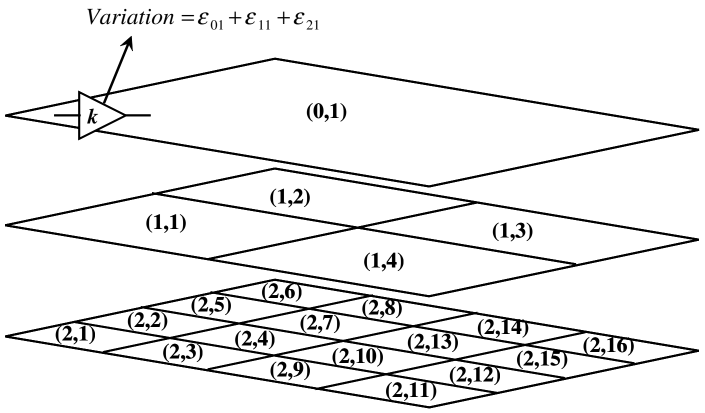


Fig. 2.13 Hierarchical model for spatial correlation.

over all levels of the regions that overlap with the location of the gate k.

The techniques proposed in [2] and [18] have been successfully applied to many practical problems. Since both techniques rely on partition, the trade-off between model accuracy and model complexity can be easily controlled by the number of total partitions. However, such partition-based approaches suffer from one major limitation: the approximated correlation function is not continuous in distance. Discontinuity will appear at the boundary of every individual region.

Most recently, several techniques have been proposed to address the aforementioned discontinuity problem. The authors in [119] proposed an efficient numerical algorithm to extract the spatial correlation function based on measurement data. The correlation extraction is formulated as a nonlinear optimization problem that can be robustly and efficiently solved by a projection-based algorithm. Bhardwaj et al. proposed an alternative algorithm to address the similar problem [10]. The algorithm proposed in [10] is based on the Karhunen–Loève expansion borrowed from stochastic process theory. It attempts to find a compact set of nonlinear basis functions to approximate the two-dimensional spatial correlation.

2.4.3 Principal Component Analysis

While many physical variations (e.g., ΔT_{OX}, ΔV_{TH}, etc.) are correlated, most statistical analysis algorithms can handle independent random variations much more easily than correlated variations. For example, it is easy to draw independent sampling points from a random number generator for Monte Carlo analysis (more details in Chapter 3). If one wants to create correlated random samples, the computational cost will be significantly increased [92]. For this reason, there is a need to mathematically represent correlated physical variations by a set of independent random variables. Principal component analysis (PCA) [98] is a statistical method that finds a compact set of independent factors to represent a *multivariate Normal distribution*. The random variables from a multivariate Normal distribution are also called *jointly Normal*.

Given N process parameters $X = [x_1, x_2, \ldots, x_N]^T$, the process variation $\Delta X = X - X_0$, where X_0 is the mean value of X, is often approximated as a zero-mean multivariate Normal distribution. The correlation of ΔX can be represented by a symmetric, positive semi-definite covariance matrix R. PCA decomposes R as

$$R = V \cdot \Sigma \cdot V^T, \tag{2.2}$$

where $\Sigma = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_N)$ contains the eigenvalues of R, and $V = [V_1, V_2, \ldots, V_N]$ contains the corresponding eigenvectors that are orthonormal, i.e., $V^T V = I$ (I is the identity matrix). Based on Σ and V, PCA defines a set of new random variables:

$$\Delta Y = \Sigma^{-0.5} \cdot V^T \cdot \Delta X. \tag{2.3}$$

These new random variables in ΔY are called the principal components or factors. It is easy to verify that all elements in $\Delta Y = [\Delta y_1, \Delta y_2, \ldots, \Delta y_N]^T$ are uncorrelated and satisfy the standard Normal distribution $N(0,1)$ (i.e., zero mean and unit standard deviation):

$$\begin{aligned} E\left(\Delta Y \cdot \Delta Y^T\right) &= E\left(\Sigma^{-0.5} \cdot V^T \cdot \Delta X \cdot \Delta X^T \cdot V \cdot \Sigma^{-0.5}\right) \\ &= \Sigma^{-0.5} \cdot V^T \cdot E\left(\Delta X \cdot \Delta X^T\right) \cdot V \cdot \Sigma^{-0.5} \\ &= \Sigma^{-0.5} \cdot V^T \cdot V \cdot \Sigma \cdot V^T \cdot V \cdot \Sigma^{-0.5} = I, \end{aligned} \tag{2.4}$$

where $E(\bullet)$ stands for the expected value operator. In addition, jointly Normal random variables are mutually independent if and only if they are uncorrelated [98]. Therefore, Equation (2.4) also implies that all elements in $\Delta Y = [\Delta y_1, \Delta y_2, \ldots, \Delta y_N]^T$ are mutually independent.

The essence of PCA can be interpreted as a coordinate rotation of the space defined by the original random variables. In addition, if the magnitudes of the eigenvalues $\{\lambda_i\}$ decrease quickly, it is possible to use a small number of random variables, i.e., a small subset of principal components, to approximate the original N-dimensional space. More details of PCA can be found in [98].

2.4.4 Non-Normal Process Variations

In most practical cases, process parameters are modeled as a multivariate Normal distribution, which allows us to apply many mathematical techniques to simplify the statistical analysis problem. PCA, for example, can decompose correlated jointly Normal variables into independent ones. If the random variables are non-Normal, mutual independence must be checked by high-order moments and, therefore, PCA cannot be applied to such cases.

There are a few cases where process variations are non-Normal, as observed from the measurement data. For example, systematic variations often exhibit non-Normal distributions, since they are not truly random. In these cases, if the non-Normal variations are mutually independent, they can be either directly incorporated into statistical analysis engine (e.g., for Monte Carlo analysis) or converted to Normal distributions by a nonlinear transform. Next, we briefly describe the nonlinear transform that converts independent non-Normal distributions to independent Normal distributions.

Given a set of random variables $\Delta X = [\Delta x_1, \Delta x_2, \ldots, \Delta x_N]^T$, we assume that all these random variables $\{\Delta x_i; i = 1, 2, \ldots, N\}$ are non-Normal and mutually independent. A set of one-dimensional functions $\{\Delta y_i = g_i(\Delta x_i);\ i = 1, 2, \ldots, N\}$ can be constructed to convert ΔX to $\Delta Y = [\Delta y_1, \Delta y_2, \ldots, \Delta y_N]^T$ such that $\{\Delta y_i;\ i = 1, 2, \ldots, N\}$ are Normal [98]. The transform function $g_i(\bullet)$ can be found via the following two steps [98].

First, the random variable Δx_i is converted to a uniform distribution $\Delta u_i \in [0,1]$ by defining

$$\Delta u_i = \mathrm{cdf}_{\Delta x_i}(\Delta x_i), \tag{2.5}$$

where $\mathrm{cdf}_{\Delta x_i}(\bullet)$ is the cumulative distribution function of Δx_i. Since any cumulative distribution function is monotonic, we have

$$P\left(\Delta u_i \le t\right) = P\left[\Delta x_i \le \mathrm{cdf}_{\Delta x_i}^{-1}(t)\right], \tag{2.6}$$

where $P(\bullet)$ denotes the probability and $\mathrm{cdf}_{\Delta x_i}^{-1}(\bullet)$ stands for the inverse function of $\mathrm{cdf}_{\Delta x_i}(\bullet)$. Therefore, the cumulative distribution function of Δu_i is equal to

$$\mathrm{cdf}_{\Delta u_i}(t) = P(\Delta u_i \le t) = P\left[\Delta x_i \le \mathrm{cdf}_{\Delta x_i}^{-1}(t)\right] = \mathrm{cdf}_{\Delta x_i}\left[\mathrm{cdf}_{\Delta x_i}^{-1}(t)\right] = t. \tag{2.7}$$

Equation (2.7) shows the fact that Δu_i is a uniform distribution.

Second, we convert the uniform distribution Δu_i to a standard Normal distribution Δy_i by defining

$$\Delta y_i = \mathrm{cdf}_{N(0,1)}^{-1}(\Delta u_i), \tag{2.8}$$

where $\mathrm{cdf}_{N(0,1)}(\bullet)$ is the cumulative distribution function of standard Normal distribution. Given the uniform distribution Δu_i, the following equation proves why Δy_i defined in (2.8) is a standard Normal distribution

$$\mathrm{cdf}_{\Delta y_i}(t) = P(\Delta y_i \le t) = P\left[\Delta u_i \le \mathrm{cdf}_{N(0,1)}(t)\right] = \mathrm{cdf}_{N(0,1)}(t). \tag{2.9}$$

Since the random variables $\{\Delta x_i;\ i = 1,2,\ldots,N\}$ are mutually independent, the random variables $\{\Delta y_i;\ i = 1,2,\ldots,N\}$ are also mutually independent and their joint probability density function is given by

$$\begin{aligned}&\mathrm{pdf}_{\Delta Y}(\Delta y_1, \Delta y_2, \ldots, \Delta y_N)\\ &\quad = \mathrm{pdf}_{\Delta y_1}(\Delta y_1) \cdot \mathrm{pdf}_{\Delta y_2}(\Delta y_2) \cdots \mathrm{pdf}_{\Delta y_N}(\Delta y_N).\end{aligned} \tag{2.10}$$

It is easy to verify that the random variables in ΔY constitute a multivariate Normal distribution [98].

The aforementioned nonlinear transform approach, however, is not applicable to correlated non-Normal distributions. At first glance, the nonlinear transform approach seems still valid. The one-dimensional nonlinear functions $\{\Delta y_i = g_i(\Delta x_i);\ i = 1,2,\ldots,N\}$ can be constructed to convert each non-Normal random variable Δx_i to a Normal variable Δy_i. Unfortunately, after the nonlinear transform is performed, the random variables $\{\Delta y_i;\ i = 1,2,\ldots,N\}$ are *not* mutually independent and their joint probability density function is not equal to the product of the marginal probability density functions, i.e.,

$$\begin{aligned}&\mathrm{pdf}_{\Delta Y}(\Delta y_1,\Delta y_2,\ldots,\Delta y_N)\\&\quad\neq \mathrm{pdf}_{\Delta y_1}(\Delta y_1)\cdot \mathrm{pdf}_{\Delta y_2}(\Delta y_2)\cdots \mathrm{pdf}_{\Delta y_N}(\Delta y_N).\end{aligned}\tag{2.11}$$

In this case, the random variables $\{\Delta y_i;\ i = 1,2,\ldots,N\}$ are not guaranteed to be a multivariate Normal distribution. In other words, *even if the random variables $\{\Delta y_i;\ i = 1,2,\ldots,N\}$ are marginally Normal, they might not be jointly Normal* [98]. This property can be understood from the following example described in [98].

Consider two random variables Δy_1 and Δy_2, and their joint probability density function:

$$\begin{aligned}\mathrm{pdf}_{\Delta Y}(\Delta y_1,\Delta y_2) = {} & \mathrm{pdf}_1(\Delta y_1)\cdot \mathrm{pdf}_2(\Delta y_2)\\ & \cdot\big\{1+\rho\cdot[2\cdot \mathrm{cdf}_1(\Delta y_1)-1]\\ & \cdot[2\cdot \mathrm{cdf}_2(\Delta y_2)-1]\big\},\end{aligned}\tag{2.12}$$

where $|\rho| < 1$, and $\mathrm{pdf}_1(\Delta y_1)$ and $\mathrm{pdf}_2(\Delta y_2)$ are two probability density functions with respective cumulative distribution functions $\mathrm{cdf}_1(\Delta y_1)$ and $\mathrm{cdf}_2(\Delta y_2)$. It is easy to verify that [98]

$$\begin{gathered}\mathrm{pdf}_{\Delta Y}(\Delta y_1,\Delta y_2)\geq 0\\ \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\mathrm{pdf}_{\Delta Y}(\Delta y_1,\Delta y_2)\cdot d\Delta y_1\cdot d\Delta y_2 = 1.\end{gathered}\tag{2.13}$$

Equation (2.13) shows that the function in (2.12) is a valid joint probability density function. In addition, directly integrating the joint probability density function yields [98]:

$$\begin{aligned}&\int_{-\infty}^{+\infty}\mathrm{pdf}_{\Delta Y}(\Delta y_1,\Delta y_2)\cdot d\Delta y_2 = \mathrm{pdf}_1(\Delta y_1)\\&\int_{-\infty}^{+\infty}\mathrm{pdf}_{\Delta Y}(\Delta y_1,\Delta y_2)\cdot d\Delta y_1 = \mathrm{pdf}_2(\Delta y_2)\end{aligned}\tag{2.14}$$

implying that $\mathrm{pdf}_1(\Delta y_1)$ and $\mathrm{pdf}_2(\Delta y_2)$ in (2.12) are the marginal probability density functions of Δy_1 and Δy_2, respectively. In particular, let $\mathrm{pdf}_1(\Delta y_1)$ and $\mathrm{pdf}_2(\Delta y_2)$ be Normal distributions. In this case, both Δy_1 and Δy_2 are marginally Normal; however, their joint probability density function in (2.12) is not a multivariate Normal distribution.

Correlated non-Normal random variables must be characterized by their joint probability density function, thereby making them extremely difficult to handle in statistical analysis. Even Monte Carlo simulation becomes impractical, if not impossible, in such cases, since it is difficult to draw random samples from a general, multi-dimensional joint probability density function [92, 94]. In addition, using correlated non-Normal distributions also increases the difficulty of process characterization, because extracting the multi-dimensional joint probability density function from silicon testing data is not trivial. For these reasons, how to efficiently model and analyze correlated non-Normal process variations remains an open topic in the IC design community.

2.5 Manufacturing Yield

The large-scale process variations significantly impact circuit performance in nano-scale technologies. From the product point of view, they directly affect the manufacturing yield. *Yield* is defined as the proportion of the manufactured chips that function correctly. As process variations become relatively large in 65 nm technologies and beyond, yield becomes one of the top concerns for today's integrated circuit design. In this sub-section, we briefly review several important concepts related to yield.

First of all, it is important to note that the yield of a specific product manufactured with a specific technology is *not* static [61]. Yield is typically low when a new process is initially developed. It must be substantially improved before high-volume production is started. Otherwise, the manufacturing cost would be extremely expensive or even unaffordable. The yield improvement could be achieved via various tunings by both design engineers and process engineers. Such a tuning procedure is called *yield learning*, as shown in Figure 2.14.

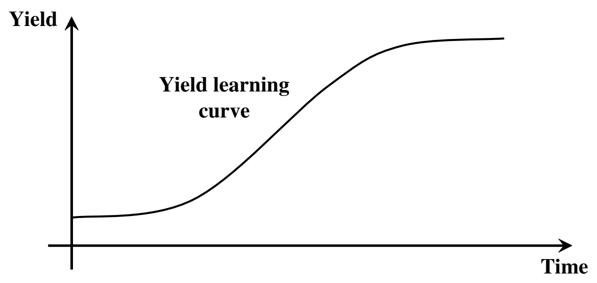


Fig. 2.14 Yield learning curve for a specific product manufactured with a specific technology.

The *faults* in a manufacturing process can be classified into two broad categories: *catastrophic* faults (due to physical and structural defects such as open, short, etc.) and *parametric* faults (due to parametric variations in process parameters such as V_{TH}, T_{OX}, etc.). Both catastrophic faults and parametric faults play important roles in integrated circuit manufacturing. In general, it is difficult to emphasize one of them while completely ignoring the other. There are several important observations of catastrophic and parametric faults for most commercial manufacturing processes.

First, catastrophic faults are often critical at the initial stage when a new technology node is developed. As the manufacturing technology becomes mature, catastrophic faults can be significantly reduced and parametric faults become increasingly important. Second, catastrophic faults are most important for large-size circuits that consume large silicon areas. For example, digital system-on-chip (SOC) designs and memory circuits significantly suffer from catastrophic faults. Finally, as process variations become relatively large due to technology scaling, parametric faults are becoming increasingly crucial at 90 nm technologies and beyond.

In summary, the relative importance of catastrophic faults and parametric faults varies from time to time as manufacturing technologies become more and more mature. It is also circuit-dependent and process-dependent. Therefore, for a specific design, both catastrophic and parametric faults must be carefully studied for yield learning.

3 Transistor-Level Statistical Methodologies

Transistor-level statistical analysis and optimization techniques focus on a single circuit block consisting of a few hundred devices. A circuit block can be, for example, a standard library cell, an interconnect wire, an analog amplifier, etc. Analyzing these circuit blocks involve transistor-level simulation using SPICE-like engine (e.g., Cadence SPECTRE, Synopsys HSPICE, etc.). For nano-scale IC technologies, even though a circuit block is small, the corresponding statistical analysis and optimization problem is not trivial mainly because of the following reasons:

- *Device models are extremely complex at 65 nm technologies and beyond.* The state-of-the-art BSIM4 model contains more than 20,000 lines of C code to describe the behavior of a *single* MOS transistor! It, in turn, results in expensive device model evaluation for transistor-level simulation. In many practical applications, it is not uncommon that more than 50% of the simulation time is consumed by device model evaluation.
- *Large-scale process variations must be characterized by complex statistical models.* A statistical device model typically

contains a great number of random variables to capture various device-level variations, thereby further increasing device model complexity. For example, a commercial 65 nm CMOS process may contain more than 300 independent random variables to model inter-die variations. If device mismatches are simultaneously considered, it will introduce more than 10 additional random variables for *every* transistor!

- *Interconnect models become increasingly sophisticated.* As the continuous device scaling pushes operational frequency to GHz region, a physical on-chip interconnect model can consist of hundreds of (or even thousands of) RC elements. In many high-speed applications (e.g., micro-processor package, high-speed data link, RF transceiver, etc.), parasitic inductance starts to play an important role, which further complicates both parasitic extraction and transistor-level simulation.

In this chapter, we review various statistical techniques that address the transistor-level analysis and optimization problem. Several recently-developed methodologies, including projection-based performance modeling (PROBE) and asymptotic probability extraction (APEX), will be discussed in detail.

3.1 Monte Carlo Analysis

Monte Carlo analysis is an important technique for statistical circuit analysis [92, 94]. It attempts to estimate the probability distribution of the performance of interest (e.g., gain, bandwidth, power, etc.) via three steps: (1) generating a set of random samples for process parameters, (2) running transistor-level simulations and evaluating the performance values at all sampling points, and (3) estimating the performance distribution by bin-based histogram or advanced kernel smoothing methods [102].

The relative cost of the aforementioned three steps is application-dependent. In general, it is difficult to identify which one of these steps dominates the overall computational time. For transistor-level

Monte Carlo analysis, the second step is often the most time-consuming part, since it involves a large number of expensive transistor-level simulations.

The accuracy of Monte Carlo analysis depends on the number of random samples. In practice, a huge number of sampling points are required to achieve sufficient accuracy, which is one of the major limitations of Monte Carlo analysis. Next, we will discuss this accuracy issue in detail.

3.1.1 Accuracy of Monte Carlo Analysis

Monte Carlo analysis is a statistical sampling technique. Theoretically, we cannot get identical results from two separate Monte Carlo simulations. It, therefore, implies that Monte Carlo accuracy must be analyzed by statistical methodologies. We will use the following simple example to show such statistical analysis.

Assume that x is a random variable with standard Normal distribution (i.e., zero mean and unit variance) and we attempt to estimate its mean value by Monte Carlo analysis. For this purpose, we randomly pick up M sampling points $\{x_1, x_2, \ldots, x_M\}$ and estimate the mean value by

$$\mu_x = \frac{1}{M} \cdot \sum_{i=1}^{M} x_i, \tag{3.1}$$

where μ_x is called a *point estimator* of the mean value [81].

Since $\{x_1, x_2, \ldots, x_M\}$ are drawn from a random number generator that follows the probability distribution of x, all $\{x_1, x_2, \ldots, x_M\}$ are standard Normal distributions. In addition, all $\{x_1, x_2, \ldots, x_M\}$ should be mutually independent, if the random number generator is sufficiently good, i.e., the period of its pseudo-random sequence is sufficiently large. Given these two assumptions, we can theoretically calculate the first two moments of μ_x

$$E(\mu_x) = E\left(\frac{1}{M} \cdot \sum_{i=1}^{M} x_i\right) = \frac{1}{M} \cdot \sum_{i=1}^{M} E(x_i) = 0 \tag{3.2}$$

$$E\left(\mu_x^2\right) = E\left[\frac{1}{M^2}\cdot\left(\sum_{i=1}^{M} x_i\right)^2\right] = \frac{1}{M^2}\cdot\sum_{i=1}^{M} E\left(x_i^2\right) = \frac{1}{M}. \tag{3.3}$$

Equation (3.2) demonstrates that the expected value of μ_x is exactly equal to the mean value of x. Therefore, the point estimator in (3.1) is called an *unbiased estimator* [81]. The variance value in (3.3) has a twofold meaning. First, given a finite value of M, the estimator μ_x does not yield a deterministic value. In other words, we cannot get identical results from two separate Monte Carlo simulations. Second, as M increases, the variance in (3.3) decreases and, therefore, the accuracy of Monte Carlo analysis is improved.

The accuracy of Monte Carlo analysis can be mathematically specified by a *confidence interval* [92]. The α-level confidence interval is defined as an interval where the statistical measurement (e.g., the estimator μ_x in the aforementioned example) falls corresponding to a given probability α. For a fixed value of α, the α-level confidence interval shrinks (meaning that Monte Carlo analysis becomes increasingly accurate), as the number of random samples increases. However, due to the statistical nature of Monte Carlo analysis, there is always a small probability (i.e., $1-\alpha$) that the statistical measurement will fall outside the confidence interval. From this point of view, even if a large number of samples are used, Monte Carlo analysis may still (although unlikely) yield a "wrong" result.

For a given accuracy specification defined by confidence interval, we can calculate the required number of Monte Carlo sampling points (i.e., M). In the aforementioned example, since all $\{x_1, x_2, \ldots, x_M\}$ are Normal distributions, the estimator μ_x in (3.1) is a linear combination of multiple Normal distributions and, therefore, is also Normal [81]. The 99.7%-level confidence interval is corresponding to the $\pm 3\sigma$ boundary of μ_x. If we require the 99.7%-level confidence interval to be $[-0.1, 0.1]$, the required number of sampling points is equal to

$$M \geq \left(\frac{3}{0.1}\right)^2 = 900. \tag{3.4}$$

If we require the 99.7%-level confidence interval to be $[-0.01, 0.01]$, the required number of sampling points is equal to:

$$M \geq \left(\frac{3}{0.01}\right)^2 = 90{,}000. \tag{3.5}$$

Studying (3.4) and (3.5), one would notice that Monte Carlo error is only reduced by 10×, if the number of random sampling points is increased by 100×. To achieve sufficient accuracy, a great number of sampling points are required, which is one of the major limitations of Monte Carlo analysis. In many practical problems, a typical selection of the number of random samples is around 1,000 $\sim$ 10,000.

One major advantage of Monte Carlo analysis is that its accuracy is independent of the underlying problem dimension (i.e., the number of random variables in the stochastic system) [92, 94]. This property implies that we do not have to increase the number of Monte Carlo sampling points as problem dimension becomes increasingly large, e.g., when both inter-die variations and device mismatches must be simultaneously considered for statistical analysis. For this reason, Monte Carlo analysis can be more attractive than other techniques (e.g., response surface modeling methods) for large-dimension problems. A detailed comparison between Monte Carlo analysis and other techniques will be given in Section 3.2.4.

3.1.2 Latin Hypercube Sampling

As discussed in 3.1.1, Monte Carlo analysis often requires a large number of random sampling points, thereby resulting in expensive computational cost. There are various techniques to control Monte Carlo samples to reduce the overall analysis cost. Instead of directly drawing random samples from a random number generator, these methods attempt to create sampling points from a controlled random sequence such that the estimation accuracy can be improved. *Latin hypercube sampling* (LHS) [64, 83] is one of these fast Monte Carlo techniques that we will discuss in detail in this sub-section.

The key idea of Latin hypercube sampling [64, 83] is to make sampling point distribution close to the probability distribution function of the random variable that we try to sample. In the one-dimensional

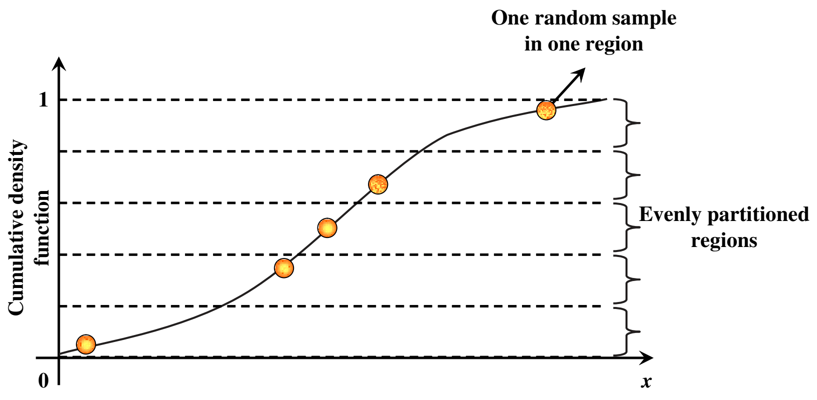

One random sample in one region
1
0
Cumulative density function
Evenly partitioned regions
x

Fig. 3.1 Five one-dimensional Latin hypercube samples for a single random variable x.

case where we attempt to generate M sampling points for a random variable x, Latin hypercube sampling consists of two steps. First, the cumulative distribution function $\mathrm{cdf}_x(x)$ is evenly partitioned into M regions. Second, a single sampling point is randomly selected in each region. Figure 3.1 shows a simple example where five Latin hypercube sampling points are created for the random variable x. In this example, Latin hypercube sampling guarantees to select five sampling points from five different regions and, therefore, it eliminates the possibility that many sampling points come from the same small local region. Since Latin hypercube sampling distributes the sampling points all over the random space, it is more efficient than direct random sampling.

The efficacy of Latin hypercube sampling can be demonstrated by the following simple example. Assume that x is a random variable with standard Normal distribution (i.e., zero mean and unit variance) and we want to estimate its mean value by (3.1) based on Monte Carlo analysis. To compare Latin hypercube sampling with direct random sampling, we estimate the mean value of x by both sampling schemes. A number of experiments are conducted with different number of sampling points ($M = 10, 100, 1{,}000$, and 10,000) such that we can study the convergence rate of the error. In addition, given a fixed number of sampling points (e.g., $M = 10$), 50 independent Monte Carlo analyses are conducted to predict the probability distribution of the estimation error.

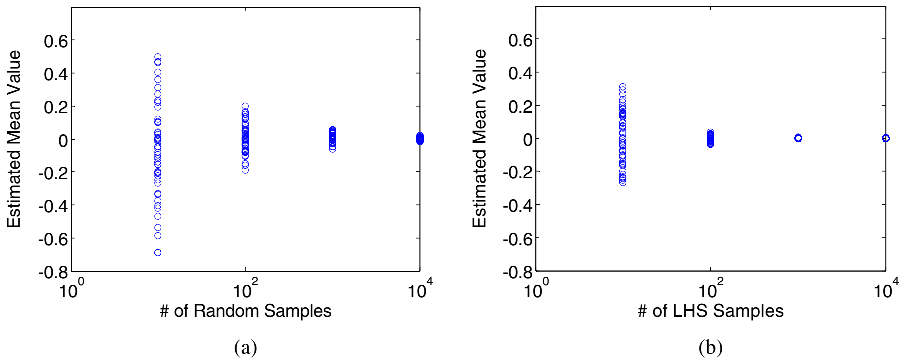


Fig. 3.2 Comparison of direct random sampling and Latin hypercube sampling. (a) Estimated mean values from random samples. (b) Estimated mean values from Latin hypercube samples.

Figure 3.2 shows the Monte Carlo analysis results and provides two important observations. First, both sampling schemes yield smaller error if a larger number of sampling points are utilized. Second, but most importantly, Latin hypercube sampling consistently results in smaller error (measured by the variance of the estimated mean value) than direct random sampling in this example.

The aforementioned one-dimensional Latin hypercube sampling can be easily extended to two-dimensional cases. Assume that we want to generate M two-dimensional Latin hypercube samples for two independent random variables x_1 and x_2. (Correlated Normal random variables can be decomposed to independent random variables by principal component analysis.) We first create two independent one-dimensional Latin hypercube sampling sets for x_1 and x_2, respectively. Each sampling set consists of M samples. Next, we randomly combine the samples in these two sets to create M two-dimensional pairs. Figure 3.3 shows a simple example where five two-dimensional Latin hypercube sampling points are created for the independent random variables x_1 and x_2. As shown in Figure 3.3, there is only a single grid filled with one sampling point in each row (or column), and the sampling point is randomly selected within that grid. As such, Latin hypercube sampling distributes its samples all over the two-dimensional random space. A high-dimensional Latin hypercube sampling can be similarly constructed [64, 83].

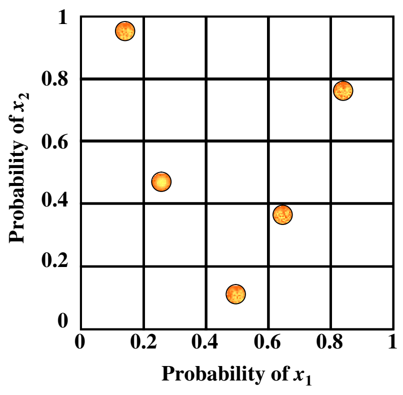


Fig. 3.3 Five two-dimensional Latin hypercube samples for two independent random variables x_1 and x_2.

3.1.3 Importance Sampling

Importance sampling is another commonly-used approach to speed-up Monte Carlo analysis [46, 92, 94]. The basic idea is to distort the original probability density function $\mathrm{pdf}_x(x)$ to reduce the variance of the estimator. Mathematically, the expected value of a given performance function $f(x)$ is defined as

$$E(f) = \int_{-\infty}^{+\infty} f(x) \cdot \mathrm{pdf}_x(x) \cdot dx. \tag{3.6}$$

Direct Monte Carlo analysis estimates $E(f)$ by drawing random sampling points from $\mathrm{pdf}_x(x)$. Importance sampling, however, attempts to find a distorted probability density function $\mathrm{pdf}_y(y)$ such that choosing random sampling points from $\mathrm{pdf}_y(y)$ yields an estimation of $E(f)$ with smaller variance.

$$\begin{aligned} E(f) &= \int_{-\infty}^{+\infty} f(y) \cdot \frac{\mathrm{pdf}_y(y)}{\mathrm{pdf}_y(y)} \cdot \mathrm{pdf}_x(y) \cdot dy \\ &= \int_{-\infty}^{+\infty} f(y) \cdot \frac{\mathrm{pdf}_x(y)}{\mathrm{pdf}_y(y)} \cdot \mathrm{pdf}_y(y) \cdot dy. \end{aligned} \tag{3.7}$$

Equation (3.7) implies that if random sampling points are selected from the distorted probability density function $\mathrm{pdf}_y(y)$, the expected value of $f(y) \cdot \mathrm{pdf}_x(y)/\mathrm{pdf}_y(y)$ is exactly equal to $E(f)$. Therefore, $E(f)$ can be estimated by the following unbiased estimator:

$$\mu_f = \frac{1}{M} \cdot \sum_{i=1}^{M} f(y_i) \cdot \frac{\mathrm{pdf}_x(y_i)}{\mathrm{pdf}_y(y_i)} \quad \left[y_i \sim \mathrm{pdf}_y(y)\right], \tag{3.8}$$

where y_i is the ith random sampling point selected from the probability density function $\mathrm{pdf}_y(y)$, and M is the total number of random samples.

It can be proven that if the distorted probability density function $\mathrm{pdf}_y(y)$ is proportional to $f(y) \cdot \mathrm{pdf}_x(y)$, i.e.,

$$\frac{f(y) \cdot \mathrm{pdf}_x(y)}{\mathrm{pdf}_y(y)} = k \;\Rightarrow\; \mathrm{pdf}_y(y) = \frac{1}{k} \cdot f(y) \cdot \mathrm{pdf}_x(y), \tag{3.9}$$

where k is a constant, the Monte Carlo analysis using importance sampling has minimal error, i.e., the variance of the estimator μ_f in (3.8) is minimized [94]. In this "ideal" case, no matter which value is randomly selected for y, the function $f(y) \cdot \mathrm{pdf}_x(y)/\mathrm{pdf}_y(y)$ is always equal to the constant k and, therefore, the variance of the estimator μ_f in (3.8) is equal to 0.

The ideal case in (3.9), however, cannot be easily applied to practical applications. It can be extremely difficult to draw random samples from a general probability density function $\mathrm{pdf}_y(y) = f(y) \cdot \mathrm{pdf}_x(y)/k$, especially if the random variable y is multi-dimensional [92, 94]. Even in the one-dimensional case, selecting random sampling points from $\mathrm{pdf}_y(y)$ requires to know its cumulative distribution function [81, 92]:

$$\mathrm{cdf}_y(y) = \int_{-\infty}^{y} \mathrm{pdf}_y(y) \cdot dy = \frac{1}{k} \cdot \int_{-\infty}^{y} f(y) \cdot \mathrm{pdf}_x(y) \cdot dy \tag{3.10}$$

implying that the integral in (3.6) must be known in advance and there would hence be no reason to run Monte Carlo analysis at all!

Based on these discussions, a practical selection of the distorted probability density function $\mathrm{pdf}_y(y)$ must satisfy the following two constraints:

- *Easy to sample.* Random samples must be created from $\mathrm{pdf}_y(y)$ easily.

- *Minimizing estimator variance.* $\mathrm{pdf}_y(y)$ must be close to $f(y) \cdot \mathrm{pdf}_x(y)/k$ as much as possible so that the estimator variance (i.e., the error) can be minimized.

Finding the optimal probability density function $\mathrm{pdf}_y(y)$ for importance sampling is not trivial in many practical applications. It is often application-dependent and must be constructed by experience, which is one of the key limitations of the importance sampling technique. For integrated circuits, one important application of importance sampling is to estimate the rare failure events of memory circuits, as reported in [46]. In such applications, the optimal probability density function $\mathrm{pdf}_y(y)$ is optimized such that the rare failure events will occur with a much higher probability and, therefore, they can be easily captured by Monte Carlo analysis.

3.1.4 Quasi Monte Carlo Analysis

An alternative strategy for better controlling Monte Carlo samples is to use so-called *low-discrepancy* samples which are deterministically chosen to "more uniformly" sample the statistical distribution [92]. The technique is called *Quasi Monte Carlo* (QMC) and is widely used in many application domains. For example, it is a standard method in the computational finance world for evaluating complex financial instruments under various forms of statistical uncertainty. The technique is applicable to the world of scaled semiconductor problems as well: speedups of 10–50$\times$ have been demonstrated in [105] compared with a direct Monte Carlo simulation.

A number of deterministic sequences can be constructed for Quasi Monte Carlo analysis such that the *divergence* (a measure of Monte Carlo analysis error) decreases in the order of $O[(\log M)^{N-1}/M]$ [92], where M is the number of random samples and N is the underlying problem dimension (i.e., the number of random variables in the stochastic system). Compared with the direct Monte Carlo analysis that has a divergence rate of $O[1/\mathrm{sqrt}(M)]$, Quasi Monte Carlo analysis can offer a substantial gain, when N is small and M is large. However, for high-dimensional problems where N is large, the benefit of using Quasi Monte Carlo analysis may be difficult to justify.

3.2 Statistical Performance Modeling

As previously discussed, Monte Carlo analysis requires a huge number of sampling points to achieve sufficient accuracy. In practice, repeatedly running transistor-level simulations for so many times is often time-consuming or even infeasible for large-size circuits. For example, simulating an industrial phase-locked loop (PLL) or analog-to-digital converter (ADC) at one single sampling point may take more than one month on a stand-alone machine!

To overcome this difficulty, *response surface modeling* (also referred to as *performance modeling*) has been widely used to reduce the computational cost. The key idea here is to approximate the performance of interest (e.g., delay, power, gain, etc.) as a polynomial function of the process parameters that are modeled as random variables (e.g., V_{TH}, T_{OX}, etc.). Such a response surface model establishes an analytical dependence between device-level variations and circuit-level performance so that statistical analysis can be further applied to estimate the performance variation efficiently.

3.2.1 Response Surface (Performance) Modeling

Given a fixed circuit design, the circuit performance f can be approximated as a linear response surface model of process parameters [69, 73]:

$$f(X) = B^T X + C, \tag{3.11}$$

where $X = [x_1, x_2, \ldots, x_N]^T$ represents the random variables to model process variations, $B \in R^N$ and $C \in R$ stand for the model coefficients, and N is the total number of the random variables of concern.

The linear approximation in (3.11) is efficient and accurate when process variations are sufficiently small. However, the recent advances in IC technologies suggest a need to revisit this assumption. As IC technologies are scaled to finer feature sizes, process variations become relatively larger. As reported in [73], the gate length variation can reach $\pm 35\%$ at 90 nm technologies and beyond. It, in turn, implies the importance of applying high-order (e.g., quadratic) response surface models to achieve high approximation accuracy [69]. Note that applying quadratic response surface models is especially important for analog circuits, since many analog performances can be strongly nonlinear in the presence of large-scale manufacturing variations. A quadratic response surface model has the form of [69]:

$$f(X) = X^T AX + B^T X + C, \tag{3.12}$$

where $C \in R$ is the constant term, $B \in R^N$ contains the linear coefficients, and $A \in R^{N\times N}$ contains the quadratic coefficients.

The unknown model coefficients in (3.11) and (3.12) can be determined by solving the over-determined linear equations at a number of sampling points [69]:

$$B^T X_i + C = \tilde{f}_i \quad (i = 1,2,\ldots,S) \tag{3.13}$$

$$X_i^T AX_i + B^T X_i + C = \tilde{f}_i \quad (i = 1,2,\ldots,S), \tag{3.14}$$

where X_i and $\tilde{f}_i$ are the value of X and the exact value of f for the ith sampling point, respectively, and S is the total number of sampling points.

Quadratic response surface model is much more accurate than linear response surface model in many practical applications. Figure 3.4 shows the circuit schematic of a low noise amplifier designed in a commercial

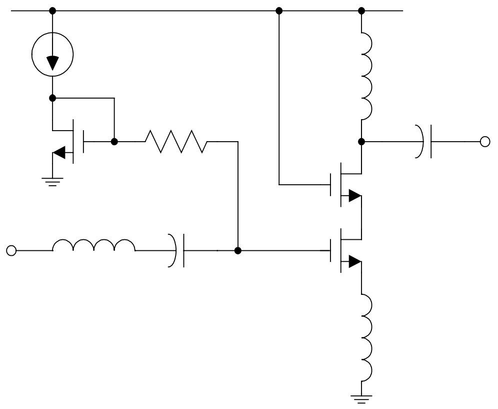


Fig. 3.4 Circuit schematic of a low noise amplifier designed in a commercial 0.25μm BiCMOS process.

0.25 μm BiCMOS process. For this low noise amplifier, the variations of both MOS transistors and passive components (capacitors and inductors) are modeled. The probability distributions and the correlation information of these variations are provided in the process design kit. After principal component analysis, eight principal factors are identified to capture all process variations. The performance of the low noise amplifier is characterized by eight different performance metrics. We approximate these performance metrics by both linear and quadratic response surface models. Table 3.1 shows the approximation error for both models. In this example, the quadratic modeling error is 7.5$\times$ smaller than the linear modeling error on average. In addition, it should be noted that the nonlinear terms in the quadratic models are expected to become increasingly important, as process variations become larger in scaled IC technologies.

Using quadratic response surface model, however, significantly increases the modeling cost. It is straightforward to verify that the number of unknown coefficients in (3.12) is $O(N^2)$, where N is the total number of the random variables to model process variations. If the total number of random variables reaches 100, a quadratic approximation will result in a 100×100 quadratic coefficient matrix containing 10,000 coefficients! The overall computational cost of quadratic response surface modeling consists of two portions:

- *Simulation cost.* i.e., the cost for running a transistor-level simulator to determine the performance value $\tilde{f}_i$ at every sampling point X_i. The number of simulation samples should be greater than the number of unknown coefficients in order

Table 3.1 Response surface modeling error for low noise amplifier.

Performance	Linear (%)	Quadratic (%)
F0	1.76	0.14
S11	6.40	1.32
S12	3.44	0.61
S21	2.94	0.34
S22	5.56	3.47
NF	2.38	0.23
IIP3	4.49	0.91
Power	3.79	0.70

to uniquely solve the linear equations in (3.14). Therefore, at least $O(N^2)$ sampling points are required to fit the quadratic model in (3.12). In practical applications, the number of samples is generally selected to be significantly larger than the unknown coefficient number to avoid over-fitting. The simulation cost is typically the dominant portion of the overall computational cost, if the transistor-level simulation is expensive for a given circuit.

- *Fitting cost.* i.e., the cost for solving the over-determined linear equations in (3.14). For the quadratic model in (3.12), the fitting cost is on the order of $O(N^6)$.

The high computational cost of quadratic response surface modeling becomes one of the most challenging problems recently, especially because intra-die variations (e.g., device mismatches) play an increasingly important role in nano-scale technologies. These intra-die variations must be modeled by many additional random variables, thereby significantly increasing the number of unknown model coefficients. This makes quadratic response surface modeling much more expensive or even infeasible in many practical nano-scale problems.

There are several techniques available to reduce quadratic response surface modeling cost. *Projection pursuit* is one of the interesting and useful techniques in this domain. The original work on projection pursuit was proposed by mathematicians in the early of 1980's [33]. The idea was recently adapted and tuned by Li et al. [57] to create an efficient projection-based extraction algorithm (PROBE) for quadratic response surface modeling. We will discuss the PROBE algorithm in detail in the next sub-section.

3.2.2 Projection-Based Performance Modeling

3.2.2.1 Mathematic Formulation

The key disadvantage of the traditional quadratic response surface modeling is the need to compute all elements of the matrix A in (3.12). This matrix is often sparse and rank-deficient in many practical problems. Therefore, instead of finding the full-rank matrix A, PROBE [57] approximates A by another low-rank matrix A_L. Such a low-rank approximation problem can be stated as follows: *given a matrix A, find another matrix A_L with rank $p <$ rank (A) such that their difference $\|A_L - A\|_F$ is minimized.* Here, $\| \bullet \|_F$ denotes the Frobenius norm, which is the square root of the sum of the squares of all matrix elements. Without loss of generality, we assume that A is symmetric in this paper, since any asymmetric quadratic form $X^T AX$ can be converted to an equivalent symmetric form $0.5 \cdot X^T(A + A^T)X$ [38].

From matrix theory [38], for any symmetric matrix $A \in R^{N\times N}$, the optimal rank-p approximation with the least Frobenius-norm error is:

$$A_L = \sum_{i=1}^{p} \lambda_i P_i P_i^T, \tag{3.15}$$

where λ_i is the ith dominant eigenvalue and $P_i \in R^N$ is the ith dominant eigenvector. The eigenvectors in (3.15) define an orthogonal projector $P_1P_1^T + P_2P_2^T + \cdots + P_pP_p^T$, and every column in A_L is the *projection* of every column in A onto the subspace span$\{P_1, P_2, \ldots, P_p\}$. PROBE uses this orthogonal projector for quadratic response surface modeling, which is intuitively illustrated in Figure 3.5.

The main advantage of the rank-p projection is that, for approximating the matrix $A \in R^{N\times N}$ in (3.12), only $\lambda_i \in R$ and $P_i \in R^N$ $(i = 1, 2, \ldots, p)$ should be determined, thus reducing the number of problem unknowns to $O(pN)$. In many practical applications, p is significantly less than N and the number of unknown coefficients that PROBE needs to solve is almost a linear function of N. Therefore,

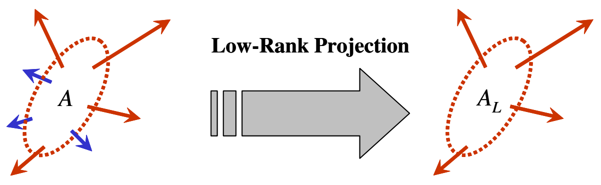


Fig. 3.5 PROBE identifies the most critical directions where process variations significantly impact a given circuit performance and then fits a quadratic response surface model along these directions [57].

compared with the problem size $O(N^2)$ of the traditional quadratic modeling, PROBE is much more efficient and can be applied to large-size problems.

3.2.2.2 Coefficient Fitting via Implicit Power Iteration

Since the matrix A in (3.12) is not known in advance, we cannot use the traditional matrix computation algorithm to compute the dominant eigenvalues λ_i and eigenvectors P_i that are required for low-rank approximation. One approach for finding the optimal rank-p model is to solve the following optimization problem for the unknown coefficients λ_i and P_i $(i = 1, 2, \ldots, p)$ and B, C:

$$\begin{aligned} &\text{minimize } \psi = \sum_i \left[X_i^T \left(\sum_{j=1}^{p} \lambda_j P_j P_j^T \right) X_i + B^T X_i + C - \tilde{f}_i \right]^2 \\ &\text{subject to } \|P_j\|_2 = 1 \quad (j = 1, \ldots, p), \end{aligned} \tag{3.16}$$

where $\|\bullet\|_2$ denotes the 2-norm of a vector.

Compared with (3.12), Equation (3.16) approximates the matrix A by $\lambda_1 P_1 P_1^T + \lambda_2 P_2 P_2^T + \cdots + \lambda_p P_p P_p^T$. Therefore, we can expect that minimizing the cost function Ψ in (3.16) will converge λ_i and P_i to the dominant eigenvalues and eigenvectors of the original matrix A, respectively. Unfortunately, Ψ in (3.16) is a 6th-order polynomial and may not be convex. In addition, the constraint set in (3.16) is specified by a quadratic equation and is not convex either. Therefore, the optimization in (3.16) is not a convex programming problem and there is no efficient optimization algorithm that can guarantee finding the global optimum for Ψ.

Instead of solving the non-convex optimization problem in (3.16), PROBE utilizes an *implicit power iteration* method to efficiently extract the unknown coefficients λ_i and P_i $(i = 1, 2, \ldots, p)$. The implicit power iteration solves a sequence of over-determined linear equations and exhibits robust convergence. In what follows, we first describe the implicit power iteration algorithm for rank-one approximation, and then extend it to rank-p approximation.

A. Rank-One Implicit Power Iteration

Algorithm 3.1 rank-one implicit power iteration.

(1) Start from a set of sampling points $\{X_i, \tilde{f}_i;\ i = 1,2,\ldots,S\}$, where S is the total number of sampling points.

(2) Randomly select an initial vector $Q_0 \in R^N$ and set $k = 1$.

(3) Compute $Q_{k-1} = Q_{k-1}/\|Q_{k-1}\|_2$.

(4) Solve the over-determined linear equations for Q_k, B_k, and C_k:

$$X_i^T Q_k Q_{k-1}^T X_i + B_k^T X_i + C_k = \tilde{f}_i \quad (i = 1,2,\ldots,S). \tag{3.17}$$

(5) If the residue:

$$\psi_k(Q_k, B_k, C_k) = \sum_i \left(X_i^T Q_k Q_{k-1}^T X_i + B_k^T X_i + C_k - \tilde{f}_i \right)^2 \tag{3.18}$$

is unchanged, i.e.,

$$|\psi_k(Q_k, B_k, C_k) - \psi_{k-1}(Q_{k-1}, B_{k-1}, C_{k-1})| < \varepsilon, \tag{3.19}$$

where ε is a pre-defined error tolerance, then go to Step (6). Otherwise, $k = k + 1$ and return Step (3).

(6) The rank-one response surface model is:

$$f_1(X) = X^T Q_k Q_{k-1}^T X + B_k^T X + C_k. \tag{3.20}$$

Algorithm 3.1 outlines the implicit power iteration algorithm for rank-one approximation. This algorithm repeatedly solves a sequence of over-determined linear equations until convergence is identified. Next, we explain why the implicit power iteration yields the optimal rank-one approximation $A_L = \lambda_1 P_1 P_1^T$. Note that Step (4) in Algorithm 3.1 approximates the matrix A by $Q_k Q_{k-1}^T$, where Q_{k-1} is determined in the previous iteration step. Finding such an optimal approximation is equivalent to solving the over-determined linear equations:

$$Q_k Q_{k-1}^T = A. \tag{3.21}$$

The least-square-error solution for (3.12) is given by [38]:

$$Q_k = AQ_{k-1} \cdot \left(Q_{k-1}^T Q_{k-1}\right)^{-1} = AQ_{k-1}. \tag{3.22}$$

In (3.22), $Q_{k-1}Q_{k-1}^T = ||Q_{k-1}||_2^2 = 1$, since Q_{k-1} is normalized in Step (3) of Algorithm 3.1. Equation (3.22) reveals an interesting fact that solving the over-determined linear equations in Step (4) "implicitly" computes the matrix-vector product AQ_{k-1}, which is the basic operation required in the traditional power iteration for dominant eigenvector computation [38].

Given an initial vector:

$$Q_0 = \alpha_1 P_1 + \alpha_2 P_2 + \cdots + \alpha_N P_N, \tag{3.23}$$

where Q_0 is represented as the linear combination of all eigenvectors of A, the kth iteration step yields:

$$Q_k = A^k Q_0 = \alpha_1 \lambda_1^k P_1 + \alpha_2 \lambda_2^k P_2 + \cdots + \alpha_N \lambda_N^k P_N. \tag{3.24}$$

In (3.24), we ignore the normalization $Q_{k-1} = Q_{k-1}/||Q_{k-1}||_2$ which is nothing else but a scaling factor. This scaling factor will not change the direction of Q_k. As long as $\alpha_1 \neq 0$ in (3.23), i.e., P_1 is not orthogonal to the initial vector Q_0, $\alpha_1\lambda_1^k P_1$ (with $|\lambda_1| > |\lambda_2| > \cdots$) will become more and more dominant over other terms. Q_k will asymptotically approach the direction of P_1, as shown in Figure 3.6.

After the iteration in Algorithm 3.1 converges, we have $Q_{k-1} = Q_{k-1}/||Q_{k-1}||_2 = P_1$ and $Q_k = AQ_{k-1} = \lambda_1 P_1$. $Q_k Q_{k-1}^T$ is the optimal rank-one approximation $A_L = \lambda_1 P_1 P_1^T$. Thus the aforementioned implicit power iteration extracts the unknown coefficients λ_1 and P_1

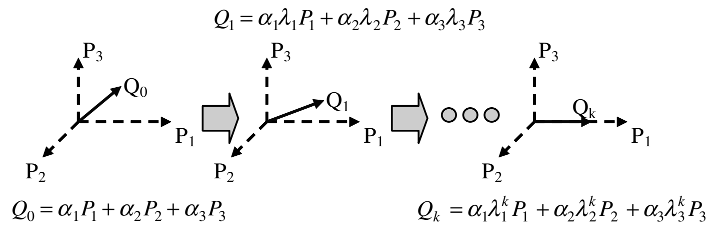

$Q_1 = \alpha_1\lambda_1 P_1 + \alpha_2\lambda_2 P_2 + \alpha_3\lambda_3 P_3$
P3
Q0
P1
P2
P3
Q1
P1
P2
P3
Qk
P1
P2
$Q_0 = \alpha_1 P_1 + \alpha_2 P_2 + \alpha_3 P_3$
$Q_k = \alpha_1\lambda_1^k P_1 + \alpha_2\lambda_2^k P_2 + \alpha_3\lambda_3^k P_3$

Fig. 3.6 Convergence of the implicit power iteration in a three-dimensional space.

with guaranteed convergence, but in an implicit way (i.e., without knowing the original matrix A). This "implicit" property is the key difference between the implicit power iteration and the traditional explicit power iteration in [38].

The above discussion demonstrates that the implicit power iteration is provably convergent if A is symmetric. For an asymmetric A, Q_{k-1} and Q_k should iteratively converge to the directions of the dominant left and right singular vectors of A to achieve the optimal rank-one approximation. However, the global convergence of the implicit power iteration is difficult to prove in this case.

B. Rank-p Implicit Power Iteration

Algorithm 3.2 rank-*p* implicit power iteration.

```
(1) Start from a set of sampling points $\{X_i, \tilde{f}_i;\ i = 1,2,\ldots,S\}$,
    where $S$ is the total number of sampling points.
(2) For each $k = \{1,2,\ldots,p\}$
(3) Apply Algorithm 3.1 to compute the rank-one approximation
    $g_k(X)$.
(4) Update the sampling points:
        $\tilde{f}_i = \tilde{f}_i - g_k(X_i) \quad (i = 1,2,\ldots,S).$   (3.25)
(5) End For.
(6) The rank-$p$ response surface model is
        $f_p(X) = g_1(X) + g_2(X) + \cdots + g_p(X).$   (3.26)
```

Algorithm 3.2 shows the implicit power iteration algorithm for rank-p approximation. Assuming that the unknown function can be approximated as the full-rank quadratic form in (3.12), Algorithm 3.2 first extracts its rank-one approximation:

$$g_1(X) = X^T(\lambda_1 P_1 P_1^T)X + B^T X + C. \tag{3.27}$$

Then, the component of $g_1(X)$ is subtracted from the full-rank quadratic function in Step (4) of Algorithm 3.2, yielding:

$$f(X) - g_1(X) = X^T \left(\sum_{i=2}^{N} \lambda_i P_i P_i^T \right) X. \tag{3.28}$$

Now λ_2 and P_2 become the dominant eigenvalue and eigenvector of the quadratic function in (3.28), and they are extracted by the rank-one implicit power iteration to generate $g_2(X)$. The rank-one implicit power iteration and the subtraction are repeatedly applied for p times until the rank-p approximation $f_p(X)$ is achieved.

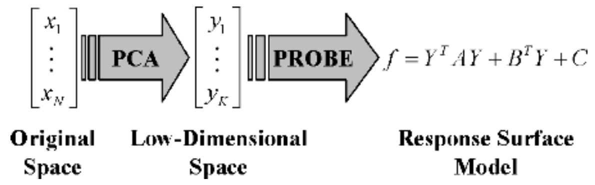
Algorithm 3.2 assumes a given approximation rank p. In practical applications, the value of p can be iteratively determined based on approximation error. For example, starting from a low-rank approximation, p should be iteratively increased if the modeling error remains large.

3.2.2.3 PROBE vs. Traditional Techniques

There are several traditional techniques, such as *principal component analysis* [98], *variable screening* [60], and *projection pursuit* [33], which can be applied to reduce the computational cost of response surface modeling. In this sub-section, we compare PROBE with these traditional techniques and highlight their difference.

A. PROBE vs. Principal Component Analysis

As discussed in Chapter 2, principal component analysis (PCA) [98] is a statistical method that can reduce the number of random variables required to approximate a given high-dimensional random space. Given an N-dimensional multivariate Normal distribution $X = [x_1, x_2, \ldots, x_N]^T$ and their covariance matrix R, PCA computes the dominant eigenvalues and eigenvectors of R, and then constructs a set of principal components $Y = [y_1, y_2, \ldots, y_K]^T$, where $K < N$, to approximate the original N-dimensional random space. After PCA, circuit performances can be approximated as functions of the principal components $\{y_i;\ i = 1, 2, \ldots, K\}$ using response surface modeling. Since the



$\begin{bmatrix} x_1 \\ \vdots \\ x_N \end{bmatrix}$
PCA
$\begin{bmatrix} y_1 \\ \vdots \\ y_K \end{bmatrix}$
PROBE
$f = Y^T AY + B^T Y + C$
Original Space
Low-Dimensional Space
Response Surface Model

Fig. 3.7 Combine PCA and PROBE to achieve minimal computational cost for quadratic response surface modeling.

number of the principal components $\{y_i;\ i = 1,2,\ldots,K\}$ is less than the number of the original variables $\{x_i;\ i = 1,2,\ldots,N\}$, PCA reduces the dimension size.

The result of PCA is uniquely determined by the covariance matrix R of the random variables $\{x_i;\ i = 1,2,\ldots,N\}$. It is independent of a specific circuit performance f. In contrast, PROBE reduces the problem dimension by carefully analyzing the dependence between a specific performance f and the random process variations. PROBE will eliminate (or keep) one eigenvector P_i if and only if f is strongly (or weakly) dependent on P_i. From this point of view, PCA and PROBE rely on completely different mechanisms for dimension reduction. In practice, both PCA and PROBE should be applied sequentially to achieve the minimal modeling cost, as shown in Figure 3.7.

B. *PROBE vs. Variable Screening*

Variable screening [60] is another traditional approach for reducing the response surface modeling cost. Given a circuit performance f, variable screening applies fractional factorial experimental design [66] and tries to identify a subset (hopefully small) of the variables that have much greater influence on f than the others. Compared with variable screening, PROBE also does a similar "screening," but with an additional coordinate rotation, as shown in Figure 3.8. The additional coordinate rotation offers more flexibility to filter out unimportant variables, thereby achieving better modeling accuracy and/or cheaper modeling cost. From this point of view, PROBE can be viewed as a *generalized variable screening* which is an extension of the traditional variable screening in [60].

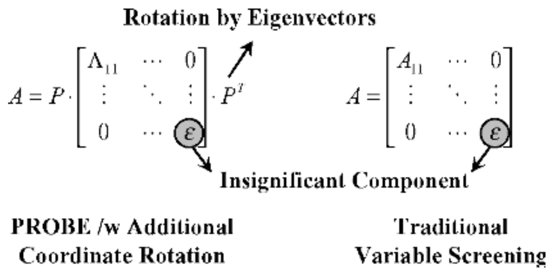

Rotation by Eigenvectors
$A = P \cdot \begin{bmatrix} \Lambda_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \varepsilon \end{bmatrix} \cdot P^T$
$A = \begin{bmatrix} A_{11} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \varepsilon \end{bmatrix}$
Insignificant Component
PROBE /w Additional Coordinate Rotation
Traditional Variable Screening

Fig. 3.8 Compared with variable screening, PROBE offers more flexibility to filter out unimportant variables by an additional coordinate rotation.

C. PROBE vs. Projection Pursuit

Projection pursuit [33] attempts to approximate the unknown high-dimensional nonlinear function by the sum of several smooth low-dimensional functions. The authors in [33] utilize the one-dimensional projection:

$$f(x) = g_1(P_1^T X) + g_2(P_2^T X) + \cdots , \tag{3.29}$$

where $g_i(\bullet)$ is the pre-defined one-dimensional nonlinear function and $P_i \in R^N$ defines the projection space. One of the main difficulties in the traditional projection pursuit is to find the optimal projection vectors P_i $(i = 1, 2, \ldots, p)$. The authors in [33] apply local optimization with heuristics to search for the optimal P_i. Such an optimization can easily get stuck at a local minimum. The PROBE algorithm is actually a special case of the traditional projection pursuit, where all $g_i(\bullet)$'s are quadratic functions. In this case, the theoretical solution of the optimal projection vectors P_i $(i = 1, 2, \ldots, p)$ is known, i.e., they are determined by the dominant eigenvalues and eigenvectors of the original quadratic coefficient matrix A. These dominant eigenvalues and eigenvectors can be extracted by the aforementioned implicit power iteration quickly and robustly.

D. PROBE vs. Full-Rank Quadratic Modeling

The rank-p implicit power iteration in Algorithm 3.2 requires running the rank-one implicit power iteration for p times. Each rank-one approximation needs to solve $2N + 1$ unknown coefficients, for which the required number of samples is on the order of $O(N)$ and solving the over-determined linear equations in Step (4) of Algorithm 3.1 has a complexity of $O(N^3)$. Therefore, a rank-p approximation requires $O(pN)$ simulation samples in total and the overall computational cost for the rank-p implicit power iteration in Algorithm 3.2 is $O(pN^3)$. In many practical applications, p is much less than N and, therefore, PROBE is much more efficient than the traditional full-rank quadratic modeling that requires $O(N^2)$ simulation samplings and has a fitting cost of $O(N^6)$ for solving the over-determined linear equations.

Figure 3.9 compares the response surface modeling complexity between PROBE and the traditional full-rank quadratic modeling. As shown in Figure 3.9, if the total number of random process parameters reaches 200, a full-rank quadratic model contains more than 20,000 unknown coefficients while the rank-one PROBE approximation reduces the coefficient number to about 400, thereby achieving significant speed-up in computational time.

To compare PROBE with other traditional techniques, a physical implementation of the ISCAS'89 S27 benchmark circuit is created using a commercial CMOS 90 nm process. Given a set of fixed gate sizes, the longest path delay in the benchmark circuit (shown in Figure 3.10) is a function of process variations. The probability distributions and the correlation information of all process variations are obtained from the

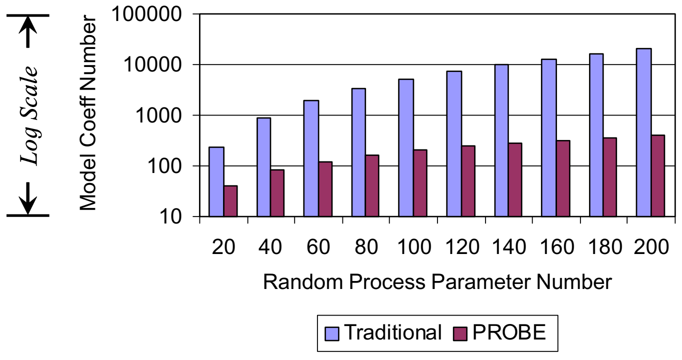


Fig. 3.9 Rank-one PROBE approximation significantly reduces the number of model coefficients compared with the traditional full-rank quadratic response surface modeling.

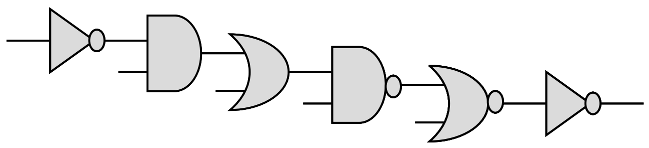


Fig. 3.10 Schematic of the longest path in the ISCAS'89 S27 benchmark circuit.

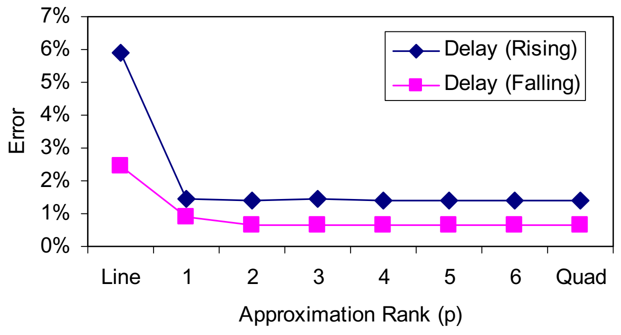


Fig. 3.11 Response surface modeling error for path delay.

process design kit. After principal component analysis, six principal factors are identified to represent these variations.

Figure 3.11 shows the response surface modeling error when the path delays of both rising and falling transitions are approximated as the linear, rank-p quadratic (PROBE), and traditional full-rank quadratic models. It is shown in Figure 3.11 that as p increase, the rank-p modeling error asymptotically approaches the full-rank quadratic modeling error. However, after $p > 2$, further increases in p do not have a significant impact on reducing error. It, in turn, implies that a rank-two model, instead of the full-rank quadratic model with rank six, is sufficiently accurate in this example.

In summary, PROBE utilizes a new projection scheme to facilitate the trade-off between modeling accuracy and computational cost. An implicit power iteration algorithm is presented to find the optimal projection space and solve the unknown model coefficients. By using the implicit power iteration, PROBE significantly reduces the modeling cost (both the required number of sampling points and the linear equation size), thereby facilitating scaling to much larger problem sizes.

The response surface models generated by PROBE can be incorporated into a statistical analysis/optimization environment for accurate and efficient yield analysis/optimization, which will be discussed in detail in Sections 3.3 and 3.4.

3.2.3 Design of Experiments

One of the key problems for response surface modeling is to optimally select the locations of sampling points such that a minimal number of samples can be used to achieve good modeling quality. Such a sample selection problem belongs to the broad research area called *design of experiments* (DOE) [66]. An experiment is defined as a test or a series of tests in which purposeful changes are made to the input variables of a process or system so that we may observe and identify the reasons for changes that may be observed in the output response. To fit the response surface model $f(X)$, we need to come up with the optimal scheme to change X (i.e., the experiment) such that the changes of f can be observed and used to accurately determine the unknown model coefficients.

To further illustrate the importance of DOE, Figure 3.12 shows two "bad" examples of sampling schemes. In Figure 3.12(a), no perturbation is applied to x_2 and therefore such a DOE cannot estimate the dependence between the performance function $f(x_1, x_2)$ and the variable x_2. On the other hand, the DOE in Figure 3.12(b) varies both x_1 and x_2, and is sufficient for fitting a linear response surface model.

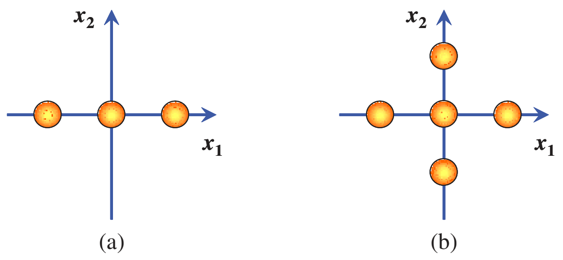


Fig. 3.12 Two "bad" examples of design of experiments. (a) x_2 is not varied and the impact of x_2 cannot be estimated for a linear model. (b) x_1 and x_2 are not varied simultaneously and the cross-product term x_1x_2 cannot be estimated for a quadratic model.

However, the DOE in Figure 3.12(b) does not vary x_1 and x_2 simultaneously and therefore it cannot estimate the cross-product term x_1x_2 for a quadratic response surface model.

Most DOE techniques for response surface modeling utilize one of the following two sampling schemes:

- *Deterministic sampling.* Sampling points are deterministically selected based on the underlying model template (e.g., linear model, quadratic model, etc.). These sampling points are optimized to minimize modeling error and, therefore, the DOE quality can be well controlled. However, it is not trivial to find an efficient deterministic sampling scheme for a general, complicated model template (e.g., when applying projection pursuit [33, 57]).
- *Random sampling.* Sampling points are randomly generated based on the probability distribution of random variables. For example, the Monte Carlo analysis discussed in Section 3.1 is one of the possible approaches to create random sampling points. The random sampling method is general and easy to implement. It is useful especially when an efficient deterministic sampling scheme is difficult to find for a given model template. However, the statistical nature of random sampling makes it difficult to achieve "guaranteed" quality in practical applications.

3.2.4 Performance Modeling vs. Monte Carlo Analysis

Both response surface modeling and Monte Carlo analysis are widely used for statistical circuit analysis. The relative efficiency of these two methods is application-dependent. In general, if the number of random process parameters is small, response surface modeling is often more efficient than Monte Carlo analysis. In such cases, only a small number of sampling points are required to fit the model. Otherwise, if the number of random process parameters is large (e.g., when both inter-die variations and device mismatches are simultaneously considered), Monte Carlo analysis is often preferable. Note that the accuracy of Monte Carlo analysis is independent of the underlying problem dimension, according to our discussion in Section 3.1.1. This property implies that we do not have to increase the number of Monte Carlo sampling points as problem dimension becomes increasingly large. For this reason, Monte Carlo analysis can be quite attractive for large-dimension problems.

3.3 Statistical Performance Analysis

The objective of statistical performance modeling is to estimate the parametric yield, given the response surface models extracted in Section 3.2. One straightforward approach for statistical performance analysis is to run Monte Carlo analysis based on response surface models. Such a model-based Monte Carlo analysis is fast, since it does not require any additional transistor-level simulations. However, an even faster statistical analysis engine is required for a number of practical applications, e.g., when parametric yield estimation must be repeatedly performed within the inner loop of an optimization flow [24, 28, 51, 97]. For this reason, various more efficient statistical analysis techniques have been developed. In what follows, we will first review the parametric yield estimation for a single performance constraint and then extend our discussion to multiple performance constraints.

Any single performance constraint can be expressed as the following standard form:

$$f(X) \leq 0, \tag{3.30}$$

where f is the performance of interest, $X = [x_1, x_2, \ldots, x_N]^T$ represents the random variables to model process variations, and N is the total number of the random variables of concern. We assume that all random variables in X are mutually independent and they satisfy the standard Normal distribution $N(0,1)$ (i.e., zero mean and unit standard deviation). Correlated Normal random variables can be decomposed to independent ones by PCA. The standard form in (3.30) is ready to handle several extensions. For example, $f(X) \leq f_0$ and $f(X) \geq f_0$ can be expressed as $f(X) - f_0 \leq 0$ and $-f(X) + f_0 \leq 0$, respectively.

Given the performance constraint in (3.30), the parametric yield is defined as

$$\text{Yield} = P(f \leq 0), \tag{3.31}$$

where $P(\bullet)$ denotes the probability. If the function $f(X)$ in (3.30) is approximated as the linear response surface model in (3.11), the parametric yield estimation problem is trivial. In this case, the performance f is Normal and its mean and variance can be, respectively, calculated by

$$\mu_f = C \tag{3.32}$$

$$\sigma_f^2 = \|B\|_2^2, \tag{3.33}$$

where $\|\bullet\|_2$ denotes the 2-norm of a vector. Since a Normal distribution is uniquely determined by its mean and variance, the parametric yield can be easily estimated based on the cumulative distribution function of the Normal distribution of f.

As shown in Table 3.1, the linear response surface model in (3.11) may be inaccurate due to the large-scale process variations in nanoscale technologies. For this reason, a quadratic response surface model is often required in many practical applications to provide high modeling accuracy. Using quadratic response surface model, however, brings about new challenges due to the nonlinear mapping between the process variations X and the circuit performance f. The distribution of f is no longer Normal, unlike the case of the linear model. In the next subsection, we describe an asymptotic probability extraction algorithm (APEX [52, 54]) to efficiently estimate the random performance distribution (and therefore the parametric yield) for a given quadratic response surface model.

3.3.1 Asymptotic Probability Extraction

Given the quadratic response surface model in (3.12), the objective of APEX is to estimate the probability density function $\text{pdf}_f(f)$ and the cumulative distribution function $\text{cdf}_f(f)$ for the performance f. APEX applies moment matching to approximate the characteristic function of f (i.e., the Fourier transform of the probability density function [81])

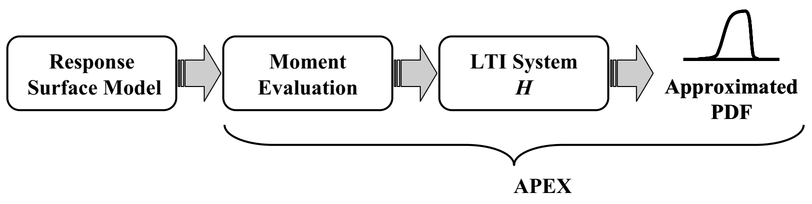


Fig. 3.13 Overall flow of APEX.

by a rational function H. H is conceptually considered to be of the form of the transfer function of a linear time-invariant (LTI) system, and the $\text{pdf}_f(f)$ and the $\text{cdf}_f(f)$ are approximated by the impulse response and the step response of the LTI system H, respectively. Figure 3.13 shows the overall flow of APEX.

3.3.1.1 Mathematic Formulation

APEX attempts to find an Mth order LTI system H whose impulse response $h(t)$ and step response $s(t)$ are the optimal approximations for $\text{pdf}_f(f)$ and $\text{cdf}_f(f)$, respectively. The variable t in $h(t)$ and $s(t)$ corresponds to the variable f in $\text{pdf}_f(f)$ and $\text{cdf}_f(f)$. The optimal approximation is determined by matching the first $2M$ moments between $h(t)$ and $\text{pdf}_f(f)$ for an Mth order approximation. In this sub-section, we first describe the mathematical formulation of the APEX algorithm. Next, APEX is linked to probability theory and we explain why it is efficient in approximating PDF/CDF functions. Finally, we show that the moment-matching method utilized in APEX is asymptotically convergent when applied to quadratic response surface models.

Define the *time moments* [15, 87] for a given circuit performance f whose probability density function is $\text{pdf}_f(f)$ as follows:

$$s_k = \frac{(-1)^k}{k!} \int_{-\infty}^{+\infty} f^k \cdot \text{pdf}_f(f) \cdot df. \tag{3.34}$$

In (3.34), the definition of time moments is identical to the traditional definition of moments in probability theory except for the scaling factor $(-1)^k/k!$.

Similarly, the time moments can be defined for an LTI system H [15, 87]. Given an Mth order LTI system whose transfer function and impulse response are

$$H(s) = \sum_{i=1}^{M} \frac{a_i}{s - b_i} \quad \text{and} \quad h(t) = \begin{cases} \sum\limits_{i=1}^{M} a_i e^{b_i t} & (\text{if } t \geq 0) \\ 0 & (\text{if } t < 0) \end{cases}. \tag{3.35}$$

The time moments of H are defined as [15, 87]:

$$s_k = \frac{(-1)^k}{k!} \int_{-\infty}^{+\infty} t^k \cdot h(t) \cdot dt = -\sum_{i=1}^{M} \frac{a_i}{b_i^{k+1}}. \tag{3.36}$$

In (3.35), the poles $\{b_i;\ i = 1,2,\ldots,M\}$ and residues $\{a_i;\ i = 1, 2,\ldots,M\}$ are the $2M$ unknowns that need to be determined. Matching the first $2M$ moments in (3.34) and (3.36) yields the following $2M$ nonlinear equations:

$$\begin{aligned} -\left(\frac{a_1}{b_1} + \frac{a_2}{b_2} + \cdots + \frac{a_M}{b_M}\right) &= s_0 \\ -\left(\frac{a_1}{b_1^2} + \frac{a_2}{b_2^2} + \cdots + \frac{a_M}{b_M^2}\right) &= s_1 \\ \vdots \qquad & \quad \vdots \\ -\left(\frac{a_1}{b_1^{2M}} + \frac{a_2}{b_2^{2M}} + \cdots + \frac{a_M}{b_M^{2M}}\right) &= s_{2M-1}. \end{aligned} \tag{3.37}$$

Equation (3.37) can be solved using the algorithm proposed in [15, 87]. It first solves the poles $\{b_i\}$ and then the residues $\{a_i\}$. In what follows, we briefly describe this two-step algorithm for solving (3.37).

In order to solve the poles $\{b_i\}$ in (3.37), the authors in [15, 87] first formulate the following linear equations:

$$-\begin{bmatrix} s_0 & s_1 & \cdots & s_{M-1} \\ s_1 & s_2 & \cdots & s_M \\ \vdots & \vdots & \vdots & \vdots \\ s_{M-1} & s_M & \cdots & s_{2M-2} \end{bmatrix} \cdot \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{M-1} \end{bmatrix} = \begin{bmatrix} s_M \\ s_{M+1} \\ \vdots \\ s_{2M-1} \end{bmatrix}. \tag{3.38}$$

After solving (3.38) for $\{c_i;\ i = 0,1,\ldots,M-1\}$, the poles $\{b_i\}$ in (3.37) are equal to the reciprocals of the roots of the following characteristic polynomial:

$$c_0 + c_1 b^{-1} + c_2 b^{-1} + \cdots + c_{M-1} b^{-M+1} + b^{-M} = 0. \tag{3.39}$$

The detailed proof of (3.38) and (3.39) can be found in [15, 87].

After the poles $\{b_i\}$ are known, substitute $\{b_i\}$ into (3.37) and the residues $\{a_i\}$ can be solved by using the first M moments:

$$-\begin{bmatrix} b_1^{-1} & b_2^{-1} & \cdots & b_M^{-1} \\ b_1^{-2} & b_2^{-2} & \cdots & b_M^{-2} \\ \vdots & \vdots & \vdots & \vdots \\ b_1^{-M} & b_2^{-M} & \cdots & b_M^{-M} \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_M \end{bmatrix} = \begin{bmatrix} s_0 \\ s_1 \\ \vdots \\ s_{M-1} \end{bmatrix}. \tag{3.40}$$

The aforementioned algorithm assumes that the poles $\{b_i\}$ are distinct. Otherwise, if repeated poles exist, the unknown poles and residues must be solved using a more comprehensive algorithm described in [15, 87].

Once the poles $\{b_i\}$ and residues $\{a_i\}$ are determined, the probability density function $\mathrm{pdf}_f(f)$ is optimally approximated by $h(t)$ in (3.35), and the cumulative distribution function $\mathrm{cdf}_f(f)$ is optimally approximated by the step response:

$$s(t) = \int_0^t h(\tau) d\tau = \begin{cases} \displaystyle\sum_{i=1}^{M} \frac{a_i}{b_i} \cdot (e^{b_i t} - 1) & (\text{if } t \geq 0) \\ 0 & (\text{if } t < 0) \end{cases}. \tag{3.41}$$

The aforementioned moment-matching method was previously applied to IC interconnect order reduction [15, 87] and is related to the *Padé approximation* in linear control theory [12]. Next, we will explain why such a moment-matching approach is efficient in approximating PDF/CDF functions.

In probability theory, given a random variable f whose probability density function is $\mathrm{pdf}_f(f)$, the characteristic function is defined as the Fourier transform of $\mathrm{pdf}_f(f)$ [81]:

$$\Phi(\omega) = \int_{-\infty}^{+\infty} \mathrm{pdf}_f(f) \cdot e^{j\omega f} \cdot df = \int_{-\infty}^{+\infty} \mathrm{pdf}_f(f) \cdot \sum_{k=0}^{+\infty} \frac{(j\omega f)^k}{k!} \cdot df. \tag{3.42}$$

Substituting (3.34) into (3.42) yields:

$$\Phi(\omega) = \sum_{k=0}^{+\infty} s_k \cdot (-j\omega)^k. \tag{3.43}$$

Equation (3.43) implies an important fact that the time moments defined in (3.34) are related to the Taylor expansion of the characteristic function at the expansion point $\omega = 0$. Matching the first $2M$ moments in (3.37) is equivalent to matching the first $2M$ Taylor expansion coefficients between the original characteristic function $\Phi(\omega)$ and the approximated rational function $H(s)$.

To explain why the moment-matching approach is efficient, we first need to show two important properties that are described in [81]:

Theorem 3.1. A characteristic function has the maximal magnitude at $\omega = 0$, i.e., $|\Phi(\omega)| \leq \Phi(0) = 1$.

Theorem 3.2. A characteristic function $\Phi(\omega) \to 0$ when $\omega \to \infty$.

Figure 3.14 shows the characteristic functions for several typical random distributions. The above two properties imply an interesting

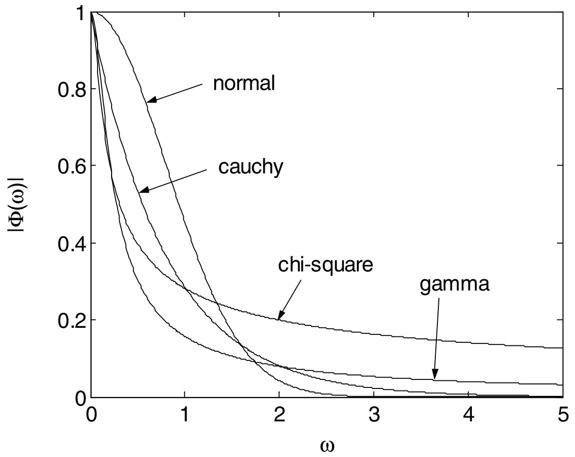


Fig. 3.14 The characteristic functions of several typical distributions.

fact: namely, given a random variable f, the magnitude of its characteristic function decays as ω increases. Therefore, the optimally approximated $H(s)$ in (3.35) is a low-pass system. It is well-known that a Taylor expansion is accurate around the expansion point. Since a low-pass system is mainly determined by its behavior in the low-frequency range (around $\omega = 0$), it can be accurately approximated by matching the first several Taylor coefficients at $\omega = 0$, i.e., the moments. In addition, the rational function form utilized in APEX is an efficient form to approximate the transfer function $H(s)$ of a low-pass system. These conclusions have been verified in other applications (e.g., IC interconnect order reduction [15, 87]) and they provide the theoretical background to explain why APEX works well for PDF/CDF approximation.

We have intuitively explained why the moment-matching approach is efficient in approximating PDF/CDF functions. However, there is a theoretical question which might be raised: given a random variable, can the PDF/CDF functions always be uniquely determined by its moments? In general, the answer is *no*. It has been observed in mathematics that some probability distributions cannot be uniquely determined by their moments. One example described in [4] is the following probability density function:

$$\mathrm{pdf}_f(f) = \begin{cases} \dfrac{e^{-0.5\cdot[\ln(f)]^2}}{\sqrt{2\pi}f} \cdot \{1 + a \cdot \sin[2\pi \cdot \ln(f)]\} & (\text{if } f > 0) \\ 0 & (\text{if } f \le 0) \end{cases}, \qquad (3.44)$$

where $a \in [-1, 1]$. It can be verified that all moments of the probability density function $\mathrm{pdf}_f(f)$ in (3.44) are independent of a, although varying a changes $\mathrm{pdf}_f(f)$ significantly [4]. It, in turn, implies that the probability density function in (3.44) cannot be uniquely determined by its moments.

However, there are special cases for which the moment problem is guaranteed to converge, i.e., the PDF/CDF functions are uniquely determined by the moments. The following Carleman theorem states one of those special cases and gives a sufficient condition for the convergence of the moment problem.

Theorem 3.3 (Carleman [4]). A probability distribution on the interval $(-\infty, +\infty)$ can be uniquely determined by its moments

$\{m_k;\ k = 1, 2, \ldots\}$ if:

$$\sum_{k=1}^{+\infty} (m_{2k})^{\frac{-1}{2k}} = \infty. \tag{3.45}$$

Based on the Carleman theorem, it can be proven that the moment-matching approach utilized in APEX is asymptotically convergent when applied to quadratic response surface models. Namely, given a quadratic response surface model f that is a quadratic function of normally distributed random variables, the probability distribution of f can be uniquely determined by its moments $\{m_k;\ k = 1, 2, \ldots, K\}$ when K approaches infinity, i.e., $K \to +\infty$. The asymptotic convergence of APEX can be formally stated by the following theorem. The detailed proof of Theorem 3.4 is given in [52].

Theorem 3.4. Given the quadratic response surface model $f(X)$ in (3.12) where all random variables in X are mutually independent and satisfy the standard Normal distribution $N(0,1)$, the probability distribution of f can be uniquely determined by its moments $\{m_k;\ k = 1, 2, \ldots\}$.

APEX is derived from the classical moment problem [4] that has been widely studied by mathematicians for over one hundred years, focusing on the theoretical aspects of the problem, e.g., the existence and uniqueness of the solution. Details of these theoretical results can be found in [4] or other recent publications, e.g., [103]. APEX aims to solve the moment problem efficiently, i.e., to improve the approximation accuracy and reduce the computational cost for practical applications. Next, we will discuss several important implementation algorithms in detail. A complete discussion of all implementation issues can be found in [52].

3.3.1.2 Binomial Moment Evaluation

A key operation required in APEX is the computation of the high order time moments defined in (3.34) for a given random variable f. Such a time moment evaluation is equivalent to computing the expected values of $\{f^k;\ k = 0, 1, \ldots, 2M - 1\}$. Given the quadratic response surface model in (3.12), f^k can be represented as a high order polynomial of X:

$$f^k(x) = \sum_i c_i \cdot x_1^{\alpha_{1i}} \cdot x_2^{\alpha_{2i}} \cdots x_N^{\alpha_{Ni}}, \tag{3.46}$$

where x_i is the ith element in the vector X, c_i is the coefficient of the ith product term, and α_{ij} is the positive integer exponent. Since the random variables in X are mutually independent, we have:

$$E(f^k) = \sum_i c_i \cdot E\left(x_1^{\alpha_{1i}}\right) \cdot E\left(x_2^{\alpha_{2i}}\right) \cdots E\left(x_N^{\alpha_{Ni}}\right). \tag{3.47}$$

In addition, remember that all random variables in X satisfy the standard Normal distribution $N(0,1)$, which yields [81]:

$$E(x^k) = \begin{cases} 1 & (\text{if } k = 0) \\ 0 & (\text{if } k = 1, 3, 5, \ldots) \\ 1 \cdot 3 \cdots (k-1) & (\text{if } k = 2, 4, 6, \ldots) \end{cases}. \tag{3.48}$$

Substituting (3.48) into (3.47), the expected value of f^k can be determined.

The above computation scheme is called *direct moment evaluation*. The key disadvantage of such a moment evaluation is that, as k increases, the total number of the product terms in (3.47) will exponentially increase, thereby quickly making the computation infeasible. To overcome this difficulty, a novel *binomial moment evaluation* scheme is developed in [52]. It recursively computes the high order moments without explicitly constructing the high order polynomial f^k in (3.47). The binomial moment evaluation consists of two steps: quadratic model diagonalization and moment evaluation.

A. Quadratic Model Diagonalization

The first step of binomial moment evaluation is to remove the cross product terms in the quadratic response surface model (3.12), thereby yielding a much simpler, but equivalent, quadratic model. According to matrix theory [38], any symmetric matrix $A \in R^{N \times N}$ can be diagonalized as:

$$A = U \cdot \Lambda \cdot U^T, \tag{3.49}$$

where $\Lambda = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_N)$ contains the eigenvalues of A and $U = [U_1, U_2, \ldots, U_N]$ is an orthogonal matrix (i.e., $U^T U = I$) containing the eigenvectors. Define the new random variables $Y = [y_1, y_2, \ldots, y_N]^T$ as follows:

$$Y = U^T \cdot X. \tag{3.50}$$

Substituting (3.50) into (3.12) yields:

$$f(Y) = Y^T \cdot \Lambda \cdot Y + Q^T \cdot Y + C = \sum_{i=1}^{N} (\sigma_i \cdot y_i^2 + q_i \cdot y_i) + C, \tag{3.51}$$

where y_i is the ith element in the vector Y and $Q = [q_1, q_2, \ldots, q_N]^T = U^T B$. Equation (3.51) implies that there is no cross product term in the quadratic model after the diagonalization. In addition, the following theorem guarantees that the random variables $\{y_i;\ i = 1, 2, \ldots, N\}$ defined in (3.50) are also independent and satisfy the standard Normal distribution $N(0,1)$. The detailed proof of Theorem 3.5 can be found in [52].

Theorem 3.5. Given a set of independent random variables $\{x_i;\ i = 1, 2, \ldots, N\}$ satisfying the standard Normal distribution $N(0,1)$ and an orthogonal matrix U, the random variables $\{y_i;\ i = 1, 2, \ldots, N\}$ defined in (3.50) are independent and satisfy the standard Normal distribution $N(0,1)$.

B. Moment Evaluation

Next, the simplified quadratic model (3.51) will be used for fast moment evaluation. Based on (3.51), a set of new random variables can be defined:

$$\begin{aligned} g_i &= \sigma_i \cdot y_i^2 + q_i \cdot y_i \\ h_l &= \sum_{i=1}^{l} g_i + C = \sum_{i=1}^{l} \left(\sigma_i \cdot y_i^2 + q_i \cdot y_i\right) + C. \end{aligned} \tag{3.52}$$

Comparing (3.52) with (3.51), it is easy to verify that when $l = N$, $h_N = f$. Instead of computing the high order moments of f directly, the binomial moment evaluation iteratively computes the moments of $\{h_l;\ l = 1,2,\ldots,N\}$, as shown in Algorithm 3.3.

Algorithm 3.3 binomial moment evaluation.

```
(1) Start from $h_0 = C$ and compute $E(h_0^k) = C^k$ for each $k = \{0,1,\ldots,2M-1\}$. Set $l = 1$.
(2) For each $k = \{0,1,\ldots,2M-1\}$
(3) Compute:
```

$$\begin{aligned} E(g_l^k) &= E[(\sigma_l \cdot y_l^2 + q_l \cdot y_l)^k] \\ &= \sum_{i=0}^{k} \binom{k}{i} \cdot \sigma_l^i q_l^{k-i} \cdot E(y_l^{k+i}) \qquad (3.53) \\ E(h_l^k) &= E[(h_{l-1} + g_l)^k] \\ &= \sum_{i=0}^{k} \binom{k}{i} \cdot E(h_{l-1}^i) \cdot E(g_l^{k-i}). \qquad (3.54) \end{aligned}$$

```
(4) End For.
(5) If $l = N$, then go to Step (6). Otherwise, $l = l + 1$ and return Step (2).
(6) For each $k = \{0,1,\ldots,2M-1\}$, $E(f^k) = E(h_N^k)$.
```

Step (3) in Algorithm 3.3 is the key operation required by the binomial moment evaluation. In Step (3), both (3.53) and (3.54) utilize the binomial theorem to calculate the binomial series. Therefore, this algorithm is referred to as binomial moment evaluation. In (3.53), the expected value $E(y_l^{k+i})$ can be easily evaluated using the closed-form expression (3.48), since y_l satisfies the standard Normal distribution $N(0,1)$. Equation (3.54) utilizes the property that h_{l-1} and g_l are independent, because h_{l-1} is a function of $\{y_i;\ i = 1,2,\ldots,l-1\}$, g_l is a function of y_l, and all $\{y_i;\ i = 1,2,\ldots,N\}$ are mutually independent (see Theorem 3.5). Therefore, $E(h_{l-1}^i \cdot g_l^{k-i}) = E(h_{l-1}^i) \cdot E(g_l^{k-i})$, where the values of $E(h_{l-1}^i)$ and $E(g_l^{k-i})$ are already computed in the previous steps.

The main advantage of the binomial moment evaluation is that, unlike the direct moment evaluation in (3.47), *it does not explicitly construct the high order polynomial* f^k. Therefore, unlike the direct moment evaluation where the total number of product terms exponentially increases with k, both $E(g_l^k)$ in (3.53) and $E(h_l^k)$ in (3.54) contain at most $2M$ product terms. Since $k = \{0, 1, \ldots, 2M-1\}$ and $l = \{0, 1, \ldots, N\}$ for an Mth order APEX approximation with N independent random variables, the total number of $E(g_l^k)$ and $E(h_l^k)$ that need to be computed is $O(MN)$. In addition, the matrix diagonalization in (3.49) is only required once and has a complexity of $O(N^3)$. Therefore, the computational complexity of the aforementioned algorithm is $O(M^2N) + O(N^3)$. In most circuit-level applications, N is small (around $5 \sim 100$) after principal component analysis, and selecting $M = 7 \sim 10$ provides sufficient accuracy for moment matching. With these typical values of M and N, the binomial moment evaluation is extremely fast, as is demonstrated in [52, 54].

It should be noted that as long as the circuit performance f is approximated as the quadratic model in (3.12) and the process variations are jointly Normal, the binomial moment evaluation yields the *exact* high order moment values (except for numerical errors). There is no further assumption or approximation made by the algorithm. For non-Normal process variations, however, the binominal moment evaluation algorithm cannot be easily applied.

In summary, the binomial moment evaluation utilizes statistical independence theory to efficiently compute the high order moments that are required by the moment matching of APEX. Compared with the direct moment evaluation in (3.46)–(3.48) whose computational complexity is $O(N^M)$, the binomial moment evaluation reduces the complexity to $O(M^2N) + O(N^3)$.

3.3.1.3 Reverse Evaluation

In many practical applications, such as robust circuit optimization, the best-case performance (e.g., the 1% point on CDF) and the worst-case performance (e.g., the 99% point on CDF) are two important metrics of interest. As discussed in Section 3.3.1.1, APEX matches the first

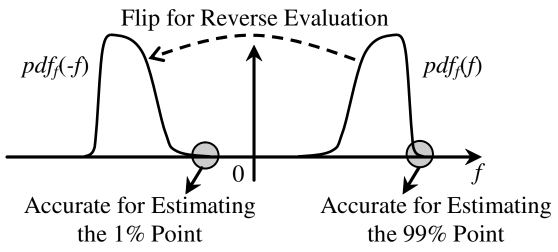

Flip for Reverse Evaluation
$pdf_f(-f)$
$pdf_f(f)$
0
f
Accurate for Estimating the 1% Point
Accurate for Estimating the 99% Point

Fig. 3.15 Illustration of the reverse evaluation.

$2M$ Taylor expansion coefficients between the original characteristic function $\Phi(\omega)$ and the approximated rational function $H(s)$. Recall that a Taylor expansion is most accurate around the expansion point $\omega = 0$. According to the final value theorem of Laplace transform, accurately approximating $\Phi(\omega)$ at $\omega = 0$ provides an accurate $\text{pdf}_f(f)$ at $f \to \infty$. It, in turn, implies that the moment-matching approach can accurately estimate the 99% point of the random distribution, as shown in Figure 3.15.

The above analysis motivates us to apply a reverse evaluation scheme for accurately estimating the 1% point. As shown in Figure 3.15, the reverse evaluation flips the original $\text{pdf}_f(f)$ to $\text{pdf}_f(-f)$. The 1% point of the original $\text{pdf}_f(f)$ now becomes the 99% point of the flipped $\text{pdf}_f(-f)$ which can be accurately evaluated by APEX.

3.3.1.4 A Digital Circuit Example

To compare APEX with other traditional techniques, a physical implementation of the ISCAS'89 S27 benchmark circuit is created using a commercial CMOS 90 nm process. Given a set of fixed gate sizes, the longest path delay in the benchmark circuit (shown in Figure 3.10) is a function of process variations. The probability distributions and the correlation information of process variations are obtained from the process design kit. After PCA, six principal factors are identified to model these variations. As shown in Figure 3.11, to approximate the delay variation, the response surface modeling error is 5.92% for linear model and 1.39% for quadratic model (4.5× difference). It is worth noting that while the linear modeling error in this example is not extremely

Table 3.2 Computational cost for moment evaluation.

	Direct		Binomial
Moment order	# of product terms	Computational time (Sec.)	Computational time (Sec.)
1	28	1.00×10^{-2}	0.01
3	924	3.02×10^{0}	0.01
5	8008	2.33×10^{2}	0.01
6	18564	1.57×10^{3}	0.01
7	38760	8.43×10^{3}	0.01
8	74613	3.73×10^{4}	0.02
10	—	—	0.02
15	—	—	0.04
20	—	—	0.07

large, as IC technologies are scaled to finer feature sizes, process variations will become relatively larger, thereby making the nonlinear terms in the quadratic model even more important.

Table 3.2 compares the computational cost for the traditional direct moment evaluation and the binomial moment evaluation. During the direct moment evaluation, the number of product terms increases exponentially, thereby making the computation task quickly infeasible. The binomial moment evaluation, however, is extremely fast and achieves more than $10^6\times$ speedup over the direct moment evaluation in this example.

Figure 3.16 shows the cumulative distribution functions for two different approximation orders. In Figure 3.16, the "exact" cumulative

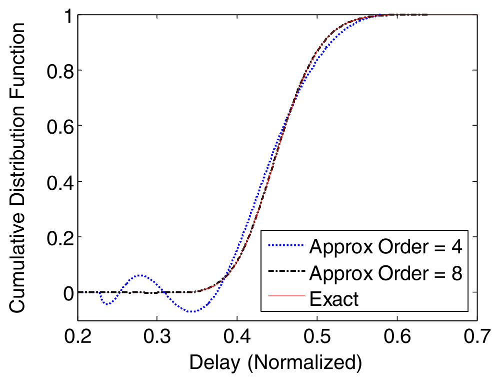


Fig. 3.16 The approximated cumulative distribution function of delay.

distribution function is evaluated by a Monte Carlo simulation with 10^6 samples. Note that the CDF obtained from the low order approximation (Order = 4) is not accurate and contains numerical oscillations. However, once the approximation order is increased to 8, these oscillations are eliminated and the approximated CDF asymptotically approaches the exact CDF. Similar behavior has been noted in moment matching of interconnect circuits [15, 87].

Table 3.3 compares the estimation accuracy and speed for four different probability extraction approaches: linear regression, Legendre approximation, Monte Carlo simulation with 10^4 samples, and APEX. The linear regression approach approximates the performance f by a best-fitted linear model with least-squared error, resulting in a Normal probability distribution for f. The Legendre approximation is often utilized in traditional mathematics. It expands the unknown probability density function by the Legendre polynomials and determines the expansion coefficients based on moment matching.

The delay values at several specific points of the cumulative distribution function are estimated by these probability extraction techniques. The 1% point and the 99% point, for example, denote the best-case delay, and the worst-case delay respectively. After the cumulative distribution function is obtained, the best-case delay, the worst-case delay and all other specific points on CDF can be easily found using a binary search algorithm. These delay values are compared with the Monte Carlo simulation results with 10^6 samples. Their relative difference is used as a measure of the estimation error for accuracy comparison, as shown in Table 3.3. The computational cost in Table 3.3 is the total

Table 3.3 Estimation error (compared against Monte Carlo with 10^6 samples) and computational cost.

	Linear	Legendre	MC (10^4 Samples)	APEX
1% Point	1.43%	0.87%	0.13%	0.04%
10% Point	4.63%	0.02%	0.22%	0.01%
25% Point	5.76%	0.12%	0.04%	0.03%
50% Point	6.24%	0.04%	0.13%	0.02%
75% Point	5.77%	0.03%	0.26%	0.02%
90% Point	4.53%	0.15%	0.38%	0.03%
99% Point	0.18%	0.77%	0.86%	0.09%
Cost (Sec.)	0.04	0.16	1.56	0.18

computational time for estimating the unknown probability density function (PDF) and cumulative distribution function (CDF).

Note from Table 3.3 that the linear regression approach has the largest error. APEX achieves a speedup of 8.7× over the Monte Carlo simulation with 10^4 samples, while still providing better accuracy. In this example, applying reverse evaluation on $\mathrm{pdf}_f(-f)$ reduces the 1% point estimation error by 4×, from 0.20% to 0.04%. This observation demonstrates the efficacy of the reverse evaluation method described in 3.3.1.3.

In summary, we describe an asymptotic probability extraction (APEX) method for estimating the non-Normal random distribution resulting from quadratic response surface modeling. Applying APEX results in better accuracy than a Monte Carlo simulation with 10^4 samples and achieves up to 10× more efficiency. In Sections 3.3.3 and 3.3.4, we will further discuss how to utilize APEX as a fundamental tool to solve various parametric yield estimation problems.

3.3.2 Convolution-Based Probability Extraction

In addition to APEX, an alternative approach for probability extraction is based on numerical convolution [125]. In this sub-section, we first show the mathematic formulation of the convolution-based technique, and then compare these two methods (i.e., APEX vs. convolution) in Section 3.3.2.2.

3.3.2.1 Mathematic Formulation

Given the diagonalized quadratic model in (3.51) and the quadratic functions $\{g_i;\ i = 1,2,\ldots,N\}$ defined in (3.52), the performance f can be expressed as a linear function of $\{g_i;\ i = 1,2,\ldots,N\}$ and the constant term C:

$$f(g_1, g_2, \ldots, g_N) = \sum_{i=1}^{N} g_i + C. \tag{3.55}$$

The quadratic functions $\{g_i;\ i = 1,2,\ldots,N\}$ in (3.52) can be re-written as the form of

$$g_i = a_i \cdot (y_i + b_i)^2 + c_i, \tag{3.56}$$

where

$$a_i = \sigma_i \quad b_i = \frac{q_i}{2\sigma_i} \quad c_i = -\frac{q_i^2}{4\sigma_i}. \tag{3.57}$$

Since the random variable y_i satisfies the standard Normal distribution $N(0,1)$, the probability density function of g_i can be analytically calculated [125]:

$$\mathrm{pdf}_{g_i}(g_i) = \begin{cases} 0 & (a_i > 0, g_i < c_i) \\ \dfrac{\exp\left\{-\left[\sqrt{(g_i-c_i)/a_i}-b_i\right]^2/2\right\}+\exp\left\{-\left[\sqrt{(g_i-c_i)/a}+b_i\right]^2/2\right\}}{2\cdot\sqrt{2\pi\cdot a_i\cdot(g_i-c_i)}} & (a_i > 0, g_i \geq c_i) \end{cases} \tag{3.58}$$

$$\mathrm{pdf}_{g_i}(g_i) = \begin{cases} \dfrac{\exp\left\{-\left[\sqrt{(g_i-c_i)/a_i}-b_i\right]^2/2\right\}+\exp\left\{-\left[\sqrt{(g_i-c_i)/a}+b_i\right]^2/2\right\}}{2\cdot\sqrt{2\pi\cdot a_i\cdot(g_i-c_i)}} & (a_i > 0, g_i \leq c_i) \\ 0 & (a_i < 0, g_i > c_i) \end{cases} \tag{3.59}$$

In addition, $\{g_i;\ i = 1,2,\ldots,N\}$ are mutually independent, because $\{y_i;\ i = 1,2,\ldots,N\}$ are mutually independent (see Theorem 3.5) and g_i is a function of y_i. Therefore, the probability density function of f is determined by

$$\mathrm{pdf}_f(f) = \mathrm{pdf}_{g_1}(f - C) \otimes \mathrm{pdf}_{g_2}(f - C) \otimes \cdots \otimes \mathrm{pdf}_{g_N}(f - C), \tag{3.60}$$

where $\otimes$ denotes the operator of convolution, i.e.,

$$f(t) \otimes g(t) = \int_{-\infty}^{+\infty} f(t - \tau) \cdot g(\tau) \cdot d\tau. \tag{3.61}$$

Note that the probability density function in (3.60) is shifted by the constant term C of the performance function (3.55). The convolutions in (3.60) can be computed by multiple, one-dimensional numerical integrations [125].

3.3.2.2 APEX vs. Convolution

For a practical application, either APEX or the convolution-based approach could be applied for probability extraction. There are two general factors that one should consider when comparing these two algorithms.

First, the convolution-based approach can be difficult to apply, if the variances of $\{g_i;\ i = 1, 2, \ldots, N\}$ in (3.56) are widely spread. In such cases, some of the g_i's have narrow probability distributions, while the others have wide probability distributions, thereby making the numerical convolution (3.60) difficult to compute. Second, APEX is most suitable for estimating the extreme values of a probability distribution, i.e., the best-case and worst-case performance values. The reverse evaluation method discussed in Section 3.3.1.3 should be selectively applied, depending on the distribution tail of interest. APEX, however, cannot produce accurate results for both distribution tails simultaneously. For this reason, the convolution-based approach is preferable, if one wants to estimate the complete probability density function including both tails.

3.3.3 Multiple Performance Constraints with Normal Distributions

The aforementioned algorithms can only be applied to a single performance metric, while the parametric yield value of most analog, digital and mixed-signal circuits is defined by multiple performance constraints. For example, a simple analog operational amplifier typically has more than ten performance metrics, including gain, bandwidth, power, phase margin, gain margin, output swing, slew rate, common-mode rejection ratio, common-mode input range, etc. For most digital circuits, delay and power (both dynamic power and leakage power) are two most important performance metrics. Therefore, the final parametric yield cannot be simply determined by a single performance constraint; instead, it depends on multiple performance constraints. Similar to (3.30), all performance constraints can be expressed as the following standard form

$$f_k(X) \leq 0 \quad (k = 1, 2, \ldots, K), \tag{3.62}$$

where f_k is the kth performance of interest, $X = [x_1\ x_2, \ldots, x_N]^T$ represents the random variables to model process variations, K is the total number of performance constraints, and N is the total number of random process variations.

Given the performance constraints in (3.62), the parametric yield is determined by

$$\text{Yield} = P(f_1 \leq 0\ \ \&\ \ f_2 \leq 0\ \ \&\ \ \cdots\ \ \&\ \ f_K \leq 0), \tag{3.63}$$

where $P(\bullet)$ denotes the probability. The probability in (3.63) depends on all performance distributions as well as their *correlations*. The following simple example demonstrates why performance correlations play an important role here.

Consider two performance metrics f_1 and f_2, and assume that both of them satisfy the standard Normal distribution $N(0,1)$ (i.e., zero mean and unit variance). Substituting $K = 2$ into (3.63) yields:

$$\text{Yield} = P(f_1 \leq 0\ \ \&\ \ f_2 \leq 0). \tag{3.64}$$

To demonstrate the importance of correlation, we consider two extreme cases. First, if f_1 and f_2 are fully correlated, the parametric yield is

$$\text{Yield} = P(f_1 \leq 0) = P(f_2 \leq 0) = 0.5. \tag{3.65}$$

The probability in (3.65) is equal to 0.5, because both f_1 and f_2 are standard Normal distributions. On the other hand, if f_1 and f_2 are mutually independent, we have:

$$\text{Yield} = P(f_1 \leq 0) \cdot P(f_2 \leq 0) = 0.25. \tag{3.66}$$

Comparing (3.65) and (3.66), we notice that different correlation values result in completely different parametric yield values in this example. In practical applications, multiple performance metrics may be neither fully correlated nor mutually independent, rendering a challenging parametric yield estimation problem. Next, we review several techniques that address this yield estimation problem. In this sub-section, we assume that all performance functions are approximated as linear response surface models and, therefore, all performance distributions are Normal. More complicated cases where performance distributions are non-Normal will be discussed in Section 3.3.4.

Given the linear response surface models for all performances of interest:

$$f_k(X) = B_k^T X + C_k \quad (k = 1, 2, \ldots, K), \tag{3.67}$$

where $B_k \in R^N$ and $C_k \in R$ are the linear model coefficients for the kth performance function, the parametric yield in (3.63) can be expressed as:

$$\begin{aligned}\text{Yield} = P(&B_1^T X + C_1 \le 0 \ \& \ B_2^T X + C_2 \le 0 \\ &\& \ \cdots \ \& \ B_K^T X + C_K \le 0).\end{aligned} \tag{3.68}$$

Define the *feasible space* as:

$$F = \left\{ X \middle| B_k^T X + C_k \le 0 \quad (k = 1, 2, \ldots, K) \right\}. \tag{3.69}$$

The feasible space in (3.69) is a *polytope*, since all performance models are linear. The parametric yield in (3.68) is equal to the integral of the probability density function $\text{pdf}_X(X)$ over the feasible space:

$$\text{Yield} = \int_F \text{pdf}_X(X) \cdot dX. \tag{3.70}$$

The integral in (3.70) is N-dimensional. Accurately calculating it by numerical integration can be extremely expensive, if N is large. One practical approach is to compute the parametric yield in (3.70) by Monte Carlo simulation. Details on Monte Carlo analysis can be found in Section 3.1. Other than Monte Carlo analysis, an alternative approach is to approximate the feasible space in (3.69) as an ellipsoid (either the maximal inscribed ellipsoid shown in Figure 3.17(a) or the minimal circumscribed ellipsoid shown in Figure 3.17(b)) and then integrate the multi-dimensional probability density function over the approximated ellipsoid. Such an ellipsoid approximation has been widely used in both analog and digital applications [1, 6, 45, 99, 117]. Next, we will discuss the ellipsoid approximation technique in detail.

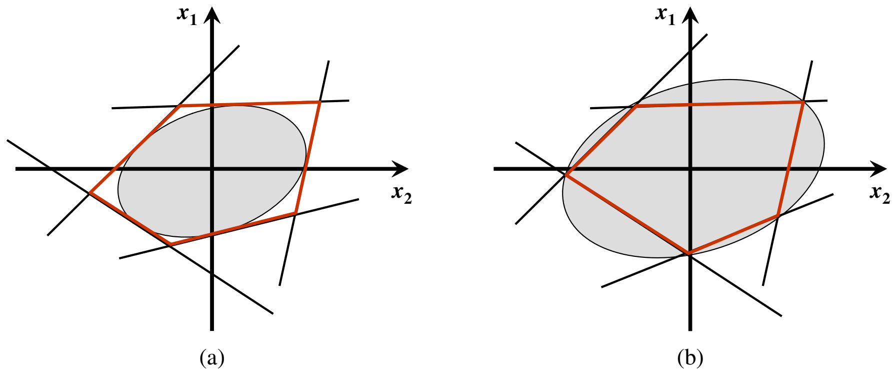


Fig. 3.17 Approximate the feasible space by an ellipsoid for parametric yield estimation. (a) Maximal inscribed ellipsoid yields the lower bound of parametric yield. (b) Minimal circumscribed ellipsoid yields the upper bound of parametric yield.

3.3.3.1 Maximal Inscribed Ellipsoid

An ellipsoid in the N-dimensional space can be defined as [13]:

$$\Omega = \{X = W \cdot u + d | \, \|u\|_2 \le 1\}, \tag{3.71}$$

where $u \in R^N$ and $d \in R^N$ are N-dimensional vectors and $W \in R^{N \times N}$ is an N-by-N symmetric, positive definite matrix. Substituting (3.71) into (3.69) yields:

$$\sup_{\|u\|_2 \le 1} (B_k^T \cdot W \cdot u + B_k^T \cdot d + C_k) = \|W \cdot B_k\|_2 + B_k^T \cdot d + C_k \le 0$$
$$(k = 1, 2, \ldots, K), \tag{3.72}$$

where sup(•) denotes the supremum (i.e., the least upper bound) of a set. In addition, it can be shown that the volume of the ellipsoid in (3.71) is proportional to det(W) [13], where det(•) denotes the determinant of a matrix. Therefore, finding the maximal inscribed ellipsoid can be formulated as the following optimization problem:

$$\begin{array}{ll} \text{maximize} & \log[\det(W)] \\ \text{subject to} & \|W \cdot B_k\|_2 + B_k^T \cdot d + C_k \le 0 \quad (k = 1, 2, \ldots, K). \end{array} \tag{3.73}$$

The nonlinear optimization in (3.73) attempts to find the optimal values of W and d such that the ellipsoid in (3.71) has the maximal volume.

Each nonlinear constraint in (3.73) consists of two parts: (1) the matrix norm $||W \cdot B_k||_2$ and (2) the linear function $B_k^T \cdot d + C_k$. Both the matrix norm and the linear function are convex [13], and the non-negative weighted sum of these two convex functions remains convex [13]. For this reason, each nonlinear constraint in (3.73) is a sub-level set of a convex function, which is a convex set. The final constraint set is the intersection of all convex sub-level sets and, therefore, is convex. In addition, by taking the log transformation of the determinant $\det(W)$, the cost function in (3.73) is concave [13]. Based on these observations, the optimization in (3.73) is a convex programming problem, since it maximizes a concave function over a convex constraint set. Several robust and efficient algorithms for solving (3.73) can be found in [13].

As shown in Figure 3.17(a), the maximal inscribed ellipsoid does not cover the entire feasible space. Therefore, the yield value estimated from the maximal inscribed ellipsoid is a lower bound of the actual parametric yield.

3.3.3.2 Minimal Circumscribed Ellipsoid

Another way to define an N-dimensional ellipsoid is in the form of [13]:

$$\Omega = \left\{X \mid \|WX + d\|_2^2 \le 1\right\}, \tag{3.74}$$

where $d \in R^N$ is an N-dimensional vector and $W \in R^{N\times N}$ is an N-by-N symmetric, positive definite matrix. To guarantee that a polytope is located inside an ellipsoid, it is necessary and sufficient to force all vertexes of the polytope in the ellipsoid. In addition, it can be shown that the volume of the ellipsoid in (3.74) is proportional to $\det(W^{-1})$ [13]. Therefore, if the polytope in (3.69) has a set of vertexes $\{X_i;\ i = 1,2,\ldots,S\}$ where S is the total number of vertexes, finding the minimal circumscribed ellipsoid can be formulated as the following optimization problem:

$$\begin{array}{ll} \text{minimize} & \log[\det(W^{-1})] \\ \text{subject to} & \|WX_i + d\|_2^2 \le 1 \quad (i = 1,2,\ldots,S). \end{array} \tag{3.75}$$

The nonlinear optimization in (3.75) attempts to find the optimal values of W and d such that the ellipsoid in (3.74) has the minimal volume.

Each nonlinear constraint in (3.75) is a positive, semi-definite quadratic function which is convex. The final constraint set is the intersection of all convex sub-level sets and, therefore, is convex. In addition, by taking the log transformation of the determinant $\det(W^{-1})$, the cost function in (3.75) is convex [13]. For these reasons, the optimization in (3.75) is a convex programming problem and it can be solved robustly and efficiently by various algorithms [13].

As shown in Figure 3.17(b), the feasible space does not cover the entire minimal circumscribed ellipsoid. Therefore, the yield value estimated from the minimal circumscribed ellipsoid is an upper bound of the actual parametric yield.

3.3.3.3 Parametric Yield Estimation over Ellipsoid

Once the feasible space is approximated as an ellipsoid (either the maximal inscribed ellipsoid in Section 3.3.3.1 or the minimal circumscribed ellipsoid in Section 3.3.3.2), the next important step is to integrate the probability density function $\text{pdf}_X(X)$ over the approximated ellipsoid for yield estimation:

$$\text{Yield} = \int_{\Omega} \text{pdf}_X(X) \cdot dX. \tag{3.76}$$

Again, directly computing the multi-dimensional integral in (3.76) is not trivial. However, the integration problem in (3.76) can be converted to a probability extraction problem that is easy to solve.

The ellipsoid representations in (3.71) and (3.74) can be, respectively, converted to the form of:

$$\begin{aligned}
&\begin{cases} X = W \cdot u + d \\ \|u\|_2 \leq 1 \end{cases} \Rightarrow \begin{cases} u = W^{-1} \cdot (X - d) \\ u^T u \leq 1 \end{cases} \\
&\qquad \Rightarrow (X - d)^T \cdot (W^{-1})^T W^{-1} \cdot (X - d) \leq 1 \\
&\qquad \Rightarrow X^T \cdot (W^{-1})^T W^{-1} \cdot X - 2 \cdot d^T \cdot (W^{-1})^T W^{-1} \\
&\qquad\quad \cdot X + d^T \cdot (W^{-1})^T W^{-1} \cdot d \leq 1
\end{aligned} \tag{3.77}$$

$$\begin{aligned}
\|WX + d\|_2^2 \leq 1 &\Rightarrow (WX + d)^T \cdot (WX + d) \leq 1 \\
&\Rightarrow X^T \cdot W^T W \cdot X + 2d^T \cdot W^T W \\
&\quad \cdot X + d^T \cdot W^T W \cdot d \leq 1.
\end{aligned} \tag{3.78}$$

In other words, both ellipsoids can be represented by the following standard form:

$$f(X) = X^T AX + B^T X + C \leq 1. \tag{3.79}$$

The function $f(X)$ in (3.79) is quadratic. The parametric yield in (3.76) is equal to the probability:

$$\text{Yield} = P(f \leq 1) = \text{cdf}_f(f = 1). \tag{3.80}$$

Given the quadratic function $f(X)$ in (3.79), its cumulative distribution function $\text{cdf}_f(f)$ can be extracted using either APEX (Section 3.3.1) or the convolution-based technique (Section 3.3.2). After that, the yield value in (3.80) can be easily calculated.

In summary, given a set of performance constraints that are approximated as linear response surface models, the aforementioned ellipsoid technique approximates the feasible space (i.e., a polytope) by a maximal inscribed ellipsoid or a minimal circumscribe ellipsoid, as shown in Figure 3.17. As such, the lower bound or the upper bound of the parametric yield can be easily estimated.

3.3.4 Multiple Performance Constraints with Non-Normal Distributions

If the performance functions are approximated as quadratic response surface models, the parametric yield estimation problem becomes much more difficult. Unlike the linear modeling case where the feasible space is a convex polytope, the quadratic mapping between the performance of interest and the random process parameters makes the feasible space much more complicated. In general, when quadratic response surface models are applied, the feasible space may be non-convex or even discontinuous. Therefore, the ellipsoid approximation in Section 3.3.3 is no longer applicable.

In this sub-section, we describe a MAX(•) approximation technique for efficient parametric yield estimation of multiple correlated non-Normal performance distributions [55]. The key idea is to conceptually map multiple performance constraints to a single *auxiliary constraint* using MAX(•) operations. Given a number of performance constraints in (3.62), we define the additional auxiliary constraint as

$$f_{\text{aux}}(X) = \text{MAX}[f_1(X), f_2(X), \ldots, f_K(X)]. \qquad (3.81)$$

It is straightforward to verify that the parametric yield in (3.63) can be uniquely determined by the auxiliary constraint, i.e.,

$$\text{Yield} = P[f_{\text{aux}}(X) \leq 0]. \qquad (3.82)$$

Even if all performance functions $\{f_k(X);\ k = 1, 2, \ldots, K\}$ are approximated as quadratic response surface models, the auxiliary constraint in (3.81) is not necessarily quadratic, due to the nonlinearity of the MAX($\bullet$) operator. However, it is possible to approximate the auxiliary constraint f_{aux} as a quadratic function of X. Such a MAX($\bullet$) approximation problem was widely studied for statistical static timing analysis [18, 19, 23, 113, 115, 124], and it has recently been tuned for analog/RF applications in [55]. Various algorithms for MAX($\bullet$) approximation are available and they will be discussed in detail in Chapter 4.

Once the auxiliary constraint in (3.81) is approximated as a quadratic function, its cumulative distribution function can be easily estimated using either APEX (Section 3.3.1) or the convolution-based technique (Section 3.3.2) to calculate the parametric yield in (3.82).

Next, we will use the low noise amplifier in Figure 3.4 as an example to demonstrate the efficacy of the aforementioned parametric yield estimation algorithm. The performance of the low noise amplifier is characterized by 8 specifications, as shown in Table 3.1. For testing and comparison purpose, we randomly select 100 sets of different specification values and the parametric yield is estimated for each set of these specifications. Figure 3.18 compares the yield estimation accuracy for two different approaches: the traditional linear approximation and the quadratic approximation. The parametric yield values estimated by both techniques are compared against the Monte Carlo analysis results with 10^4 random samples. Their absolute difference is used as a measure of the estimation error for accuracy comparison.

As shown in Figure 3.18, the traditional linear approximation cannot accurately capture the parametric yield of the low noise amplifier and the maximal absolute yield estimation error reaches 11%. The quadratic approximation achieves much better accuracy and it reduces

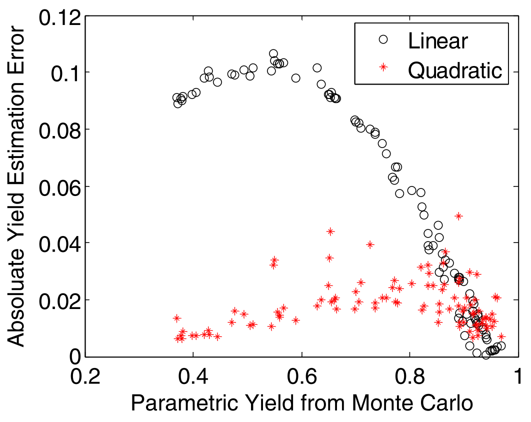


Fig. 3.18 Absolute parametric yield estimation error for low noise amplifier.

the maximal error to 5%. On average, the quadratic approximation is 3× more accurate than the traditional linear approximation in this example.

3.4 Statistical Transistor-Level Optimization

The analysis techniques described in the previous sub-sections eventually facilitate statistical transistor-level optimization in the presence of large-scale process variations. In this sub-section, we focus on the optimization problems for analog, digital and mixed-signal circuits. The techniques described in this sub-section mainly target for circuit-level blocks, e.g., operational amplifier (analog), flip–flop (digital), sense amplifier (memory), etc., although several of these algorithms have been successfully applied to large-scale integrated systems (e.g., analog-to-digital converters, phase-locked loop, etc.) that consist of a large number of transistors.

The purpose of statistical optimization is to improve parametric yield. It takes a fixed circuit topology as input, statistically predicts random performance distributions, and optimizes the device sizes to leave sufficient and necessary performance margins to accommodate large-scale process variations. In what follows, we first briefly review several nominal and corner-based transistor-level optimization techniques and explain their limitations in Section 3.4.1. Next, we extend our discussions to statistical transistor-level optimization in Section 3.4.2 and show a two-step statistical transistor-level optimization flow in Section 3.4.3.

3.4.1 Nominal and Corner-Based Transistor-Level Optimization

Most algorithms for transistor-level optimization fall into one of the following two categories: *equation-based* optimization [21, 31, 37, 41, 43, 44, 47, 62, 96, 114], and *simulation-based* optimization [32, 39, 49, 50, 74, 85, 88, 107]. The equation-based approaches utilize analytic performance models, where each circuit-level performance (e.g. gain, bandwidth, etc.) is approximated as an analytical function of design variables (e.g. transistor sizes, bias current, etc.). These analytical models can be manually derived by hand analysis [21, 31, 41, 43, 44, 47, 62, 96, 114], or automatically generated by symbolic analysis [37]. The equation-based optimization is extremely fast; however it only offers limited accuracy since generating analytic equations for complicated performance metrics (e.g., nonlinear distortion) is not trivial and requires various simplifications that ignore many non-idealities. In contrast, the simulation-based methods run numerical simulations to measure circuit performance. They are much more accurate, but also much more expensive, than the equation-based approaches. Note that achieving high accuracy for performance evaluation is extremely important for statistical optimization. The process variations in today's IC technologies typically introduce $20\% \sim 30\%$ variations on circuit performance. If the performance evaluation error is not sufficiently smaller than this value, the parametric yield cannot be accurately predicted and optimized. A detailed review on transistor-level optimization can be found in [36].

Nominal transistor-level optimization typically minimizes a cost function (e.g., power) by pushing many other performance constraints to their boundaries. Therefore, a nominally-optimized design can easily violate a number of performance specifications, if process variations are considered, as shown in Figure 3.19. Most importantly, nominal

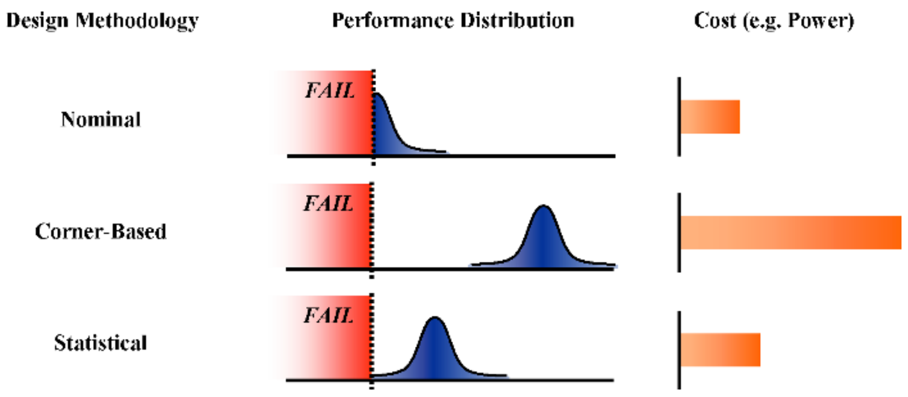


Fig. 3.19 Nominal optimization pushes performance to specification boundary, corner-based optimization often results in significant over-design, and statistical optimization leaves sufficient and necessary performance margin to accommodate process variations.

optimization often results in unstable bias condition for analog circuits and, therefore, makes the optimized circuits extremely sensitive to process variations. This issue can be intuitively explained by the following simple example.

Figure 3.20 shows the circuit schematic of a simple common-source amplifier that consists of three devices: the NMOS transistor M, the resistor R, and the capacitor C. Assume that we are interested in the following optimization problem for this amplifier circuit:

$$\begin{array}{ll} \text{minimize} & \text{Power} \\ \text{subject to} & \text{Gain} \geq 10. \end{array} \tag{3.83}$$

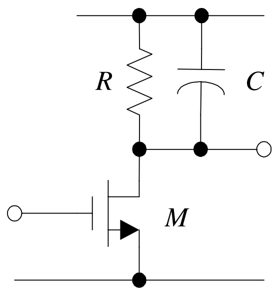


Fig. 3.20 Circuit schematic of a common-source amplifier.

It is easy to verify that the performances Power and Gain can be approximated by the first-order analytical equations [91]:

$$\begin{aligned} \text{Power} &= V_{\text{DD}} \cdot I_{\text{B}} \\ \text{Gain} &= \frac{2 \cdot I_{\text{B}} \cdot R}{V_{\text{GS}} - V_{\text{TH}}} \end{aligned} \tag{3.84}$$

where V_{DD} denotes the power supply voltage, I_{B} represents the bias current, and V_{GS} is the gate-to-source voltage of the NMOS transistor. Substituting (3.84) into (3.83) yields:

$$\begin{aligned} &\text{minimize} \quad V_{\text{DD}} \cdot I_{\text{B}} \\ &\text{subject to} \quad \frac{2 \cdot I_{\text{B}} \cdot R}{V_{\text{GS}} - V_{\text{TH}}} \geq 10. \end{aligned} \tag{3.85}$$

Studying (3.85), one would notice that the bias current I_{B} must be minimized to reduce power. Meanwhile, as I_{B} is decreased, $V_{\text{GS}} - V_{\text{TH}}$ must be minimized to satisfy the gain requirement. The nominal optimization, therefore, results in the bias condition:

$$V_{\text{GS}} \approx V_{\text{TH}}. \tag{3.86}$$

In other words, the NMOS transistor sits on the boundary between "on" and "off." In this case, a small increase in V_{TH} due to process variations can turn the transistor off and make the amplifier fail to operate.

Based on these observations, nominal circuit optimization may fail to set up a robust bias condition for analog circuits. Although the bias condition is not explicitly defined as a performance specification, it is extremely important to maintain the correct functionality of a analog circuit and, therefore, is referred to as the *implicit*, *topology-given* constraints in [32, 107]. The design space where all topology-given constraints are satisfied is called the *feasible region*. It is a small subset of the global analog design space, as shown in Figure 3.21. It is observed that, in many practical applications, analog design space is weakly nonlinear in feasible region [39, 107], while the global analog design space is strongly nonlinear and contains many local minima.

To overcome the aforementioned limitation of nominal optimization, corner-based optimization has often been utilized [21, 31, 32, 39, 41, 43, 44, 47, 49, 50, 62, 74, 85, 88, 96, 107, 114]. It optimizes

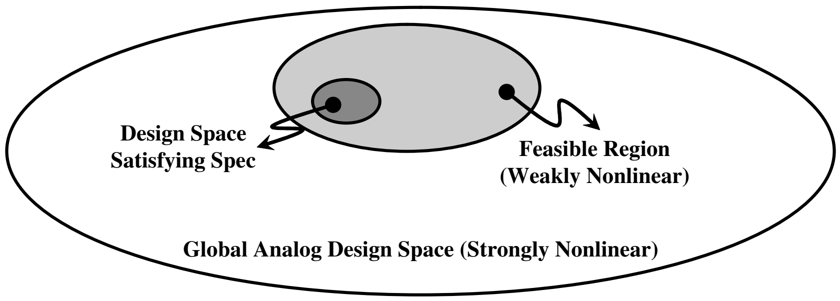


Fig. 3.21 Feasible region in analog design space.

the circuit at all process corners by combining the extreme values of all process parameters. However, the corner-based approach suffers from the problem that the total number of corners increases exponentially with the number of independent process parameters. For this reason, device mismatches cannot be efficiently handled by the corner-based methodology. Furthermore, it is not guaranteed that the worst-case performance will occur at one of these corners. As discussed in Chapter 2, the realistic worst-case performance corner is topology-dependent and performance-dependent. Simply combining the extreme values of independent process parameters is pessimistic, because it is unlikely for all process parameters to reach their worst-case values simultaneously. Therefore, corner-based optimization often results in significant over-design in practical applications, as shown in Figure 3.19.

3.4.2 Statistical Transistor-Level Optimization

During the past two decades, many statistical techniques have been proposed to address the large-scale process variation problem for integrated circuit design. The objective of these statistical methodologies is to accurately predict random performance distributions and then optimize the device sizes to leave sufficient and necessary performance margins to accommodate process variations. In other words, these statistical optimization methods attempt to improve parametric yield while simultaneously reducing design pessimism (i.e., over-design), as shown in Figure 3.19.

Most statistical transistor-level optimization techniques can be classified into four broad categories: (1) *direct yield optimization* [29, 30, 97, 116], (2) *worst-case optimization* [24, 28, 51, 58], (3) *design centering* [1, 6, 99, 117], and (4) *infinite programming* [67, 122].

The direct yield optimization methods [29, 30, 97, 116] search the design space and estimate the parametric yield for each design point by either numerical integration or Monte Carlo analysis. The design point with maximal parametric yield is then selected as the final optimal result. In many practical applications, estimating the exact parametric yield is often expensive. It, in turn, motivates the development of many other statistical optimization techniques that attempt to *approximate* the actual parametric yield by various simplifications.

The worst-case optimization approaches [28, 24, 51, 58] optimize the worst-case circuit performances, instead of nominal performances, over all process variations. For example, the worst-case optimization for a digital circuit block may have the form of:

$$\begin{array}{ll} \text{minimize} & \text{Power}_{\text{WC}} \\ \text{subject to} & \text{Delay}_{\text{WC}} \le 100ps, \end{array} \tag{3.87}$$

where Power_{WC} and Delay_{WC} denote the worst-case power and delay, respectively. These worst-case performance values are defined as the tails of the corresponding probability density functions (PDF), e.g., the 99% point of the cumulative distribution function (CDF) for the performance Delay in (3.87), as shown in Figure 3.22.

It should be noted that parametric yield cannot be uniquely determined by worst-case performance values. As shown in (3.63) and discussed in Section 3.3.3, parametric yield must be determined by all performance distributions as well as their *correlations*. While the worst-case optimization fully considers the random performance distributions, it completely ignores the correlations between different performance metrics. In general, given the performance constraints in (3.62) and the parametric yield definition in (3.63), we have:

$$\text{Yield} = P(f_1 \le 0 \;\&\; f_2 \le 0 \;\cdots\; f_K \le 0) \le P(f_k \le 0) \quad (k = 1,2,\ldots,K). \tag{3.88}$$

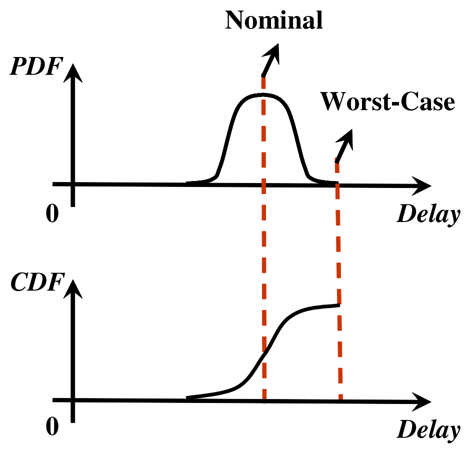


Fig. 3.22 Definition of the worst-case delay.

Namely, the worst-case optimization can only guarantee a yield upper bound and the actual parametric yield may be smaller.

The design centering approaches [1, 6, 99, 117] attempt to find the optimal design that is furthest away from all constraint boundaries and is therefore least sensitive to process variations. In design centering, the robustness of a design is assessed by the volume of the approximated ellipsoid shown in Figure 3.17. The design yielding maximal ellipsoid volume is selected as the final optimal result. As discussed in Section 3.3.3, the approximated ellipsoid estimates either the lower bound (for maximal inscribed ellipsoid) or the upper bound (for minimal circumscribed ellipsoid) of the actual parameter yield.

The infinite programming methods [67, 122] formulate the circuit optimization problem as a nonlinear infinite programming where the cost function and/or the constraints are defined over an infinite set covering process variations. The first step for such an infinite programming is to define an infinite set Ω to cover the random variation space with a given confidence level ξ, i.e.,

$$P(X \in \Omega) = \zeta, \tag{3.89}$$

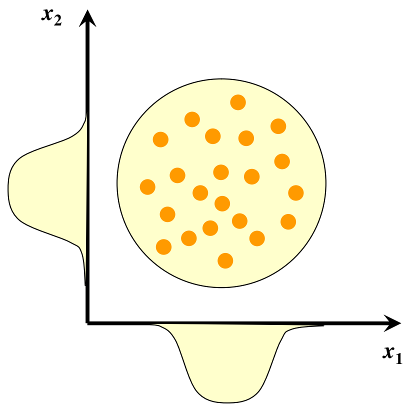


Fig. 3.23 x_1 and x_2 are independent and satisfy the standard Normal distribution $N(0,1)$. The values of x_1 and x_2 are most likely to fall in the two-dimensional ball.

where $P(\bullet)$ denotes the probability and $X = [x_1, x_2, \ldots, x_N]^T$ are the random variables to model process variations. For the special case where all random variables in X are mutually independent and satisfy the standard Normal distribution $N(0,1)$, the set Ω in (3.89) is an N-dimensional ball and its radius is determined by the confidence level ξ, as shown in Figure 3.23.

The infinite programming techniques attempt to minimize one cost function while simultaneously satisfying all performance constraints over the infinite set Ω. Taking the simple digital circuit application in (3.87) as an example, the infinite programming can be formulated as

$$\begin{array}{ll} \text{minimize} & \text{Power} \\ \text{subject to} & \sup\limits_{X \in \Omega} (\text{Delay}) \leq 100ps, \end{array} \tag{3.90}$$

where $\sup(\bullet)$ denotes the supremum (i.e., the least upper bound) of a set. Since the constraints of (3.90) are defined over an infinite set, implying that the number of constraints is infinite, the optimization in (3.90) is referred to as the infinite programming in mathematics. Given the confidence level in (3.89) and the optimization formulation in (3.90), it is easy to verify that ξ is the lower bound of the parametric yield, if a feasible solution can be found for (3.90).

3.4.3 A Two-Step Robust Design Flow

It is important to note that statistical transistor-level optimization for analog circuits should not take a nominally-optimized design as the initial starting point. As discussed in Section 3.4.1, a nominally optimized analog circuit does not have a stable bias condition and, therefore, the design space is strongly nonlinear and contains many local minima in the neighborhood. For this reason, starting from a nominally optimized analog circuit, the statistical optimization may easily get stuck at one of the local minima. In this sub-section, we will show a two-step optimization flow to achieve robust transistor-level design for practical applications.

The robust design flow described in this sub-section is facilitated by a combination of equation-based optimization and simulation-based optimization. The entire design flow consists of two steps. First, an initial design is created by the performance centering algorithm [59] based on analytical design equations. The performance centering method is a special design centering technique that can be applied to topology selection at the earlier design stages. Simplified device and parasitic models are utilized in this step to simplify design equations and speed up nonlinear optimization. This first-step optimization provides a rapid but coarse search over the global design space and finds a robust initial design sitting in the feasible region.

Next, in the second step, taking the robust initial design as the starting point, a robust analog/digital optimization algorithm (ROAD [51, 58]) is applied with detailed device/parasitic/variation models to perform a more fine-grained search and to optimize the worst-case circuit performance considering large-scale process variations. ROAD is a special worst-case optimization tool that falls in the same category as [24, 28].

3.4.3.1 Performance Centering

The objective of performance centering is to accomplish two important tasks: (1) optimally compare and select circuit topologies for a given set of design specifications; (2) quickly search the global design space and create a robust initial design for further post-tuning. In other words, performance centering attempts to quickly search the global design space and then create an initial design that can be used for detailed implementation in the later design stages. For this purpose, equation-based performance models are utilized in order to make the global search problem tractable.

Many research works [21, 31, 41, 43, 44, 47, 62, 96], have demonstrated that most circuit-level performance specifications can be cast as posynomial functions. The performances of interest include both explicit constraints (e.g., gain, delay, etc.) and implicit constraints (related to bias conditions [39, 107]). Let $Z = [z_1, z_2, \ldots, z_M]^T$ be M real and positive design variables (e.g., bias current, transistor sizes, etc.). A function f is called *posynomial* if it has the form of

$$f(Z) = \sum_i c_i z_1^{\alpha_{1i}} z_2^{\alpha_{2i}} \cdots z_M^{\alpha_{Ni}}, \tag{3.91}$$

where $c_i \in R_+$ and $\alpha_{ij} \in R$. Note that the coefficients c_i must be non-negative, but the exponents α_{ij}'s can be real values. If all performance metrics are approximated as posynomial functions, circuit sizing can be formulated as a geometric programming problem:

$$\begin{array}{lll} \text{minimize} & f_0(Z) & \\ \text{subject to} & f_k(Z) \leq 1 & (k = 1, \ldots, K) \\ & z_m > 0 & (m = 1, \ldots, M), \end{array} \tag{3.92}$$

where $f_0, f_1, \ldots, f_K$ are normalized circuit performance metrics and they are approximated as posynomial functions. The standard formulation in (3.92) is ready to handle several extensions. For example, a performance specification $f(Z) \geq 1$ can be written as $1/f(Z) \leq 1$, if $1/f(Z)$ can be approximated as a posynomial function [42]. The geometric programming problem in (3.92) attempts to find the optimal value of Z that minimizes f_0 while satisfying all other constraints $\{f_k(Z) \leq 1;\ k = 1, 2, \ldots, K\}$ and $\{z_m > 0;\ m = 1, 2, \ldots, M\}$. The optimization in (3.92) can be converted to a convex programming problem and solved in an extremely efficient way [13].

It should be noted, however, the posynomial models in (3.91) and (3.92) may be inaccurate since they are derived from simplified device models and circuit equations. The modeling error must be properly

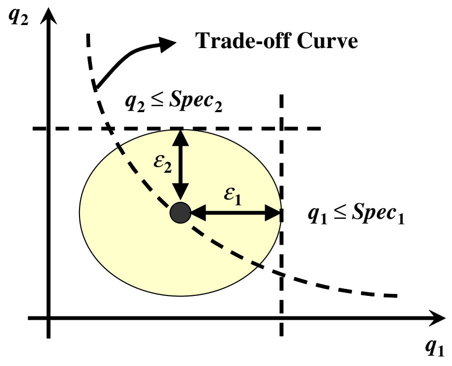


Fig. 3.24 Illustration of performance centering with two performance specifications q_1 and q_2.

considered within the optimization flow. For this purpose, a performance centering approach is developed in [59] for initial global sizing. The performance centering approach is derived from the traditional design centering methods [1, 6, 99, 117]. It attempts to center the design within the performance space and maximize the inscribed ellipsoid constrained by all performance boundaries, as shown in Figure 3.24. Importantly, the performance centering approach simultaneously maximizes the design margin for all performance constraints such that the resulting ellipsoid volume represents a *quality* measure for a given circuit topology. For example, since the models are known to be imprecise, a small ellipsoid volume indicates that the performance specifications will probably not be achievable in silicon. This ellipsoid volume, therefore, can be used as a criterion for topology selection at the earlier design stages.

Without loss of generality, we normalize all posynomial performance constraints:

$$f_k(Z) \leq 1 \quad (k = 1, \ldots, K), \tag{3.93}$$

where $\{f_k(Z);\ k = 1, 2, \ldots, K\}$ are posynomials and K is the total number of performance constraints. It should be noted that the formulation in (3.93) is substantially different from the form in (3.62). First, the performance constraints in (3.93) are represented as functions of the design variables Z (for circuit optimization), while those in (3.62) are functions of the random process variations X (for parametric yield estimation). Second, Equation (3.93) contains both explicit constraints (e.g., gain, delay, etc.) and implicit constraints (related to bias conditions [39], [107]), in order to prevent the optimization from converging to an unstable bias condition. In (3.62), however, the parametric yield is determined by explicit constraints only.

Given (3.93), the performance centering problem can be formulated as

$$\begin{array}{lll} \text{maximize} & \varepsilon_1 \cdot \varepsilon_2 \cdots \varepsilon_K & \\ \text{subject to} & \varepsilon_k = 1 - f_k(Z) & (k = 1,2,\ldots,K) \\ & \varepsilon_k > 0 & (k = 1,2,\ldots,K) \\ & z_m > 0 & (m = 1,2,\ldots,M). \end{array} \tag{3.94}$$

where $\{\varepsilon_k;\ k = 1,2,\ldots,K\}$ are a set of variables that represent the lengths of the ellipsoid axes (see Figure 3.24). The optimization in (3.94) solves the optimal values of $\{\varepsilon_k;\ k = 1,2,\ldots,K\}$ and $\{z_m;\ m = 1,2,\ldots,M\}$ by maximizing the ellipsoid volume $\varepsilon_1 \cdot \varepsilon_2 \cdots \varepsilon_K$. Such an optimization can also be generalized to use other cost functions to measure the ellipsoid size, e.g. $\varepsilon_1^2 + \varepsilon_2^2 + \cdots + \varepsilon_K^2$.

The optimization problem in (3.94) does not match the standard geometric programming form in (3.92), since it maximizes a posynomial cost function with multiple posynomial equality constraints. However, Equation (3.94) can be equivalently converted to:

$$\begin{array}{lrl} \text{minimize} & \varepsilon_1^{-1} \cdot \varepsilon_2^{-1} \cdots \varepsilon_K^{-1} & \\ \text{subject to} & f_k(Z) + \varepsilon_k \le 1 & (k = 1,2,\ldots,K) \\ & \varepsilon_k > 0 & (k = 1,2,\ldots,K) \\ & z_m > 0 & (m = 1,2,\ldots,M). \end{array} \tag{3.95}$$

Comparing (3.94) and (3.95), we note that maximizing the cost function $\varepsilon_1 \cdot \varepsilon_2 \cdots \varepsilon_K$ in (3.94) is equivalent to minimizing the cost function $\varepsilon_1^{-1} \cdot \varepsilon_2^{-1} \cdots \varepsilon_K^{-1}$ in (3.95). In addition, since maximizing $\varepsilon_1 \cdot \varepsilon_2 \cdots \varepsilon_K$ always pushes all $\{\varepsilon_k;\ k = 1,2,\ldots,K\}$ to their maximal values, the inequality constraints $\{f_k(Z) + \varepsilon_k \le 1;\ k = 1,2,\ldots,K\}$ in (3.95) will become active, i.e. reach $\{q_k(X) + \varepsilon_k = 1;\ k = 1,2,\ldots,K\}$, after the optimization. According to these observations, we conclude that the optimization problems in (3.94) and (3.95) are equivalent. This conclusion can be formally proven by using the Karush–Kuhn–Tucker optimality condition in optimization theory [9].

Equation (3.95) is a geometric programming problem and, therefore, can be solved efficiently. Solving the optimization in (3.95) yields: (1) the maximal ellipsoid volume in the performance space that can be used as a criterion to compare different circuit topologies and (2) the initial values for all design variables that can be used as the starting point for further detailed sizing.

Shown in Figure 3.25 are the circuit schematics of two operational amplifiers (Op Amp). As an example, our design objective is to select the optimal circuit topology from Figure 3.25(a) and Figure 3.25(b) for the performance specifications in Table 3.4. Both operational amplifiers are implemented with a commercial BiCMOS $0.25\,\mu$m process in this example.

We construct the posynomial performance models for both operational amplifiers, and formulate the performance centering problem as (3.95). The details of the design equations can be found in [42]. Given these posynomial equations, the performance centering problem in (3.95) can be efficiently solved using geometric programming, taking $1 \sim 2$ seconds for this amplifier design example on a SUN – 1 GHz server.

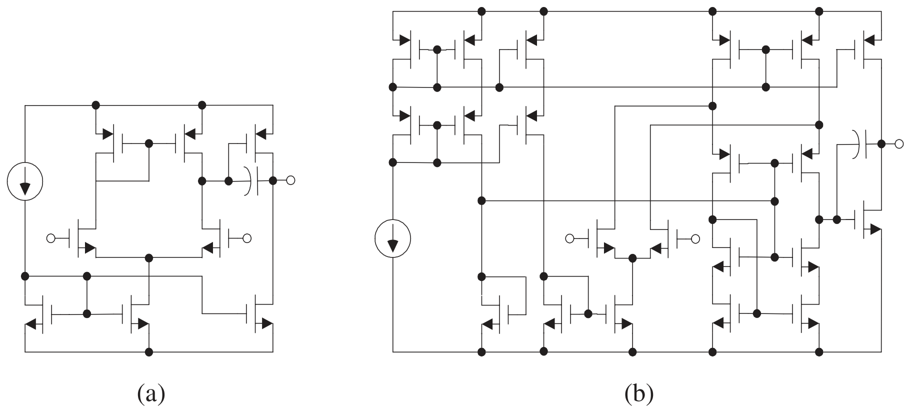


Fig. 3.25 Circuit schematics of two operational amplifiers. (a) A simple two-stage operational amplifier. (b) A folded-cascode two-stage operational amplifier.

Table 3.4 Design specifications for operational amplifier.

Performance	Specification
V_{DD} (V)	$= 2.5$
Gain (dB)	≥ 100
UGF (MHz)	≥ 10
Phase Margin (degree)	≥ 60
Slew Rate (V/μs)	≥ 20
Swing (V)	≥ 0.5
Power (mW)	≤ 20

Figure 3.26 compares the operational amplifier topologies in terms of the maximized ellipsoid volume in the performance space. As we would expect, the two-stage folded-cascode topology is better than the simple two-stage topology when the power supply voltage is sufficiently high to provide the necessary voltage headroom. In this example, we find that a sufficient voltage is 2.5 V, whereas, the folded-cascode topology appears to be inferior to the simple two-stage topology once the supply voltage is dropped to 2.0 V. Perhaps less obvious, however, we find that for extremely high gain specification, the quality measure (i.e. the ellipsoid volume) for the simple Op Amp once again falls below that for the folded-cascode Op Amp, even at a 2.0 V supply. This indicates that the folded-cascode configuration would provide a better topology for detailed implementation even at $V_{\mathrm{DD}} = 2.0$ V if the gain requirement

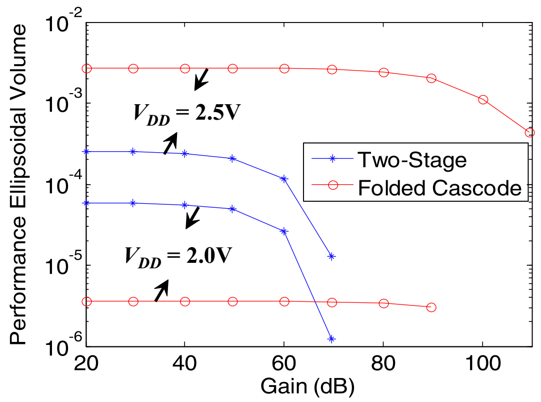


Fig. 3.26 Maximized ellipsoid volume in performance space when applying different gain and V_{DD} specifications for operational amplifier.

is high enough. Given the performance specifications in Table 3.4, the folded-cascode Op Amp topology in Figure 3.25(b) provides larger performance margin (i.e., larger ellipsoid volume) and, therefore, represents our preferred topology for these performance specifications.

It is important to note that as IC technologies continue to scale, many traditional analog circuit topologies will begin to break down owing to reduced power supply voltages and/or device non-idealities such as excessive leakage current. It is essential to understand the limitation of each topology at the earlier design stages and select the best topology for detailed circuit implementation in the later design stages.

3.4.3.2 Robust Analog/Digital Optimization (ROAD)

Once the circuit topology and the initial device sizing are determined, detailed circuit optimization based on accurate transistor-level simulation should be further applied to improve parametric yield and/or circuit performance. The robust analog/digital design tool (ROAD [51, 58]) uses a combination of response surface modeling, statistical analysis, and nonlinear programming. Figure 3.27 outlines the ROAD optimization flow which consists of three major steps.

- Run transistor-level simulation and fit the quadratic response surface models $\{f_k(Z,X);\ k = 1,2,\ldots,K\}$ for all circuit performances in the local design space, where f_i stands for

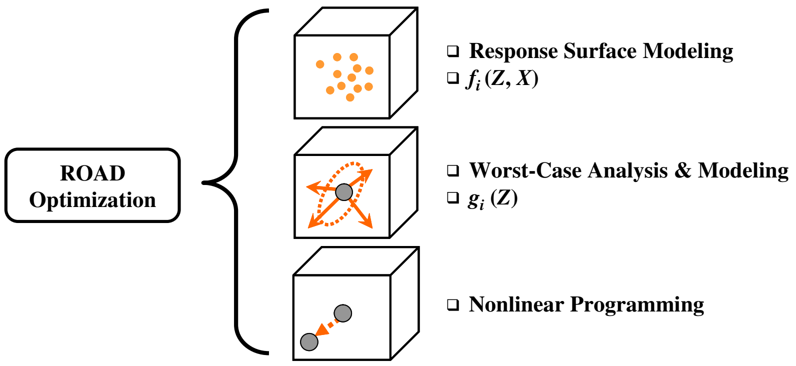


Fig. 3.27 ROAD optimization flow.

the ith circuit performance, $Z = [z_1, z_2, \ldots, z_M]^T$ contains the design variables and $X = [x_1, x_2, \ldots, x_N]^T$ contains the random process parameters.

- Based on the response surface models $\{f_k(Z,X);\ k = 1,2,\ldots,K\}$, apply statistical analysis to extract the probability distributions of all performance metrics. Fit the worst-case performance models $\{g_k(Z);\ k = 1,2,\ldots,K\}$ as functions of the design variables $\{z_m;\ m = 1,2,\ldots,M\}$. The worst-case performance g_k can be defined as the 1% or 99% point of the cumulative distribution function, for example.
- Using these pre-extracted worst-case performance models, optimize the circuit by nonlinear programming.

The fitting and optimization procedure in Figure 3.27 is repeatedly applied to successively narrowed local design spaces during ROAD iterations, as shown in Figure 3.28. Since ROAD sets up all performance constraints with process variations and the performance models become increasingly accurate in the successively narrowed local design space, it can converge to an accurate design with high parametric yield. It should be noted, however, that since ROAD is a local optimization tool, the initial starting point can have a significant impact on the quality of the final optimized design. For example, if a bad initial design is used, ROAD can even fail to converge.

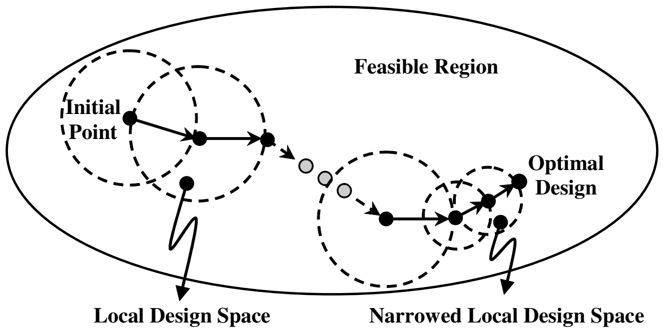


Fig. 3.28 Successive ROAD iterations.

Table 3.5 ROAD optimization result for operational amplifier.

Performance	Specification	Nominal	Worst-case
Gain (dB)	≥ 100	102.7	100.0
UGF (MHz)	≥ 10	10.9	10.0
Phase margin (degree)	≥ 60	63.5	60.0
Slew rate (V/μs)	≥ 20	20.5	19.9
Swing (V)	≥ 0.5	1.00	1.00
Power (mW)	≤ 20	0.79	0.86

We apply ROAD to the operational amplifier example in Figure 3.25(b), using the performance centering result in Section 3.4.3.1 as the initial starting point. Table 3.5 shows the nominal and worst-case performance values after ROAD optimization. The worst-case performance values are measured at the 1% or 99% points of the cumulative distribution functions that are extracted from 1000 transistor-level Monte Carlo simulations in Cadence SPECTRE. It is shown in Table 3.5 that ROAD leaves sufficient margin for each performance metric. These additional performance margins help the operational amplifier to meet performance specifications under process variations.

4

System-Level Statistical Methodologies

As we move from transistor level toward system level, one major challenge for statistical analysis and optimization stems from the underlying large problem size that has a twofold meaning. First, a full integrated system typically contains millions of transistors and, hence, the circuit size is huge. Second, to capture the intra-die variations of the entire system, a large number of random variables must be utilized. For example, if we attempt to model the per-transistor random doping variation that is expected to be the dominant variation component at 45 nm technologies and beyond [4], the total number of random variables may reach $10^3 \sim 10^6$ for large-scale integrated systems. These two issues make system-level analysis and optimization extremely difficult, especially if quadratic response surface models must be utilized to capture non-Normal performance distributions. In such cases, if the total number of random variables reaches 10^6, a quadratic approximation will result in a $10^6 \times 10^6$ quadratic coefficient matrix containing 10^{12} coefficients! The challenging problem is how to facilitate accurate and affordable statistical analysis and optimization for such a large problem size.

Recently, a number of special techniques have been proposed to address the system-level statistical analysis problem. These techniques typically utilize a hierarchical flow to partition a large-size system into multiple small pieces such that the statistical analysis and optimization problem becomes tractable. In addition, advanced numerical algorithms are proposed to efficiently manipulate the large number of random variables that model both inter-die and intra-die variations. Depending on the application of interest, various algorithms are developed to explore the trade-offs between analysis accuracy and computational complexity.

In this chapter, we focus on the statistical methodologies for full-chip digital integrated circuits. In particular, full-chip statistical timing analysis and leakage analysis will be discussed in detail.

4.1 Statistical Timing Analysis

The purpose of timing analysis is to verify the timing constraints (including both setup constraints and hold constraints) for a full-chip digital circuit. Timing analysis can be either *dynamic* or *static*. Dynamic timing analysis implies the simulation of a circuit from a given start time to a given end time. The inputs to the circuit during this period are fully specified. These input signals are called *input patterns* or *input vectors*. The circuit response is then solved by either SPICE or other fast simulators [25, 27]. Static timing analysis, on the other hand, attempts to characterize a circuit for all time, independent of its input signals [95]. It is often used to estimate the delay of an interconnected set of combinational logic blocks between the flip–flops of a digital circuit.

Timing analysis can also be classified into two broad categories: *path-based* and *block-based*. Path-based timing analysis predicts the maximal delay for a number of pre-selected logic paths. It can be conducted with the consideration of random process variations [5, 40, 45, 70, 78, 76]. The path-based technique is efficient and accurate if a few critical paths can be easily identified. For example, it has been widely used for the timing analysis of micro-processor circuits [5]. On the other hand, block-based timing analysis propagates arrival times on a timing graph in a breadth-first order. It does not require preselecting any critical paths and its computational complexity linearly scales with circuit size. The block-based technique has been widely used for the full-chip timing analysis of digital circuits where many equally critical paths may exist after timing optimization.

In this chapter, we mainly focus on block-based static timing analysis. We first introduce several important concepts for nominal block-based static timing analysis in Section 4.1.1. Next, statistical block-based static timing analysis will be discussed in detail in Section 4.1.2.

4.1.1 Nominal Block-Based Static Timing Analysis

Given a circuit netlist, block-based timing analysis translates the netlist into a *timing graph*, i.e., a weighted directed graph $G = (V, E)$ where each node $V_i \in V$ denotes a primary input, output or internal net, each edge $E_i = \langle V_m, V_n \rangle \in E$ denotes a *timing arc*, and the weight $D(V_m, V_n)$ of E_i stands for the delay value from the node V_m to the node V_n. In addition, a *source/sink* node is conceptually added before/after the primary inputs/outputs so that the timing graph can be analyzed as a single-input single-output network. Figure 4.1 shows a simple timing graph example.

There are several key concepts in nominal block-based timing analysis. They are briefly summarized as follows. It is important to note that the following terminologies are only defined for latest arrival time and required time. However, our discussions can be extended to earliest arrival time and required time easily.

- The *arrival time* (AT) at a node V_i is the latest time that the signal becomes stable at V_i. It is determined by the longest path from the source node to V_i.
- The *required time* (RT) at a node V_i is the latest time that the signal is allowed to become stable at V_i. It is determined by the longest path from V_i to the sink node.
- *Slack* is the difference between the required time and the arrival time, i.e., RT − AT. Therefore, positive slack means that the timing constraint is satisfied, while negative slack means that the timing constraint is failed.

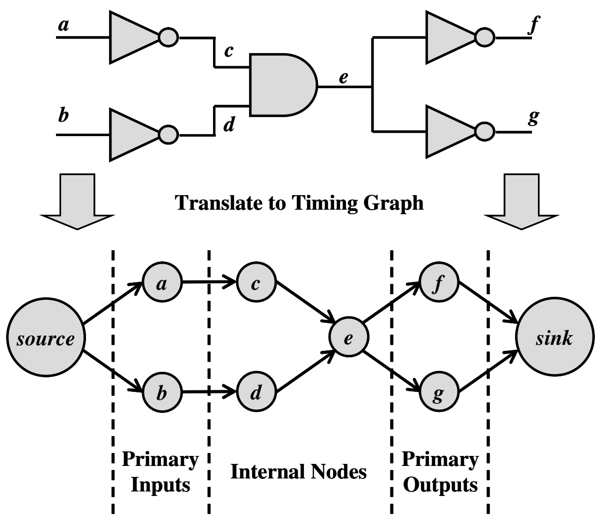


Fig. 4.1 A simple timing graph example.

- *Critical path* is the longest path between the source node and the sink node. In nominal timing analysis, all nodes along the critical path have the same (smallest) slack.

The purpose of nominal static timing analysis is to compute the arrival time, required time and slack at each node and then identify the critical path. Taking the arrival time as an example, static timing analysis starts from the source node, propagates the arrival times through each timing arc by a breadth-first traversal, and eventually reaches the sink node. Two atomic operations, i.e., SUM(•) and MAX(•) as shown in Figure 4.2, are repeatedly applied during such a traversal.

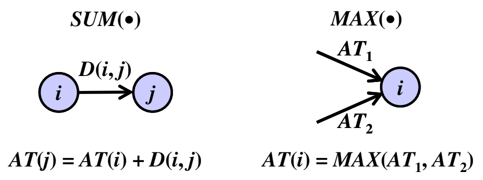


Fig. 4.2 Atomic operations for static timing analysis.

After the nominal static timing analysis is complete, the resulting critical path and slack values provide the information that is required for timing optimization. Roughly speaking, the gates and interconnects along the critical path (where the slacks are small) can be upsized to improve circuit speed, while those along the non-critical paths (where the slacks are large) can be down-sized to save chip area or power consumption. Of course, there are more subtle implications with up/down-sizing gates that can be shown as counter-examples to this over-simplification of the problem. For example, the increase in gate capacitance with upsizing may create a larger delay increase on the upstream logic stage, than the improvement in delay due to increasing the drive strength of the logic stage that is resized. Such cases are readily handled with accurate delay models and proper sensitivity information.

4.1.2 Statistical Block-Based Timing Analysis

Unlike nominal timing analysis, the gate/interconnect delays in statistical timing analysis are all modeled as random variables to account for large-scale process variations. That means, the weight $D(V_m, V_n)$ associated with a timing arc is a random variable, instead of a deterministic value. Therefore, the two atomic operations, SUM($\bullet$) and MAX($\bullet$), must handle statistical distributions. Many numerical algorithms have been proposed to perform statistical SUM($\bullet$) and MAX($\bullet$) operations. Next, we briefly review these algorithms and highlight their advantages and limitations. We will focus on the SUM($\bullet$) and MAX($\bullet$) of two random variables, since multi-variable operations can be broken down into multiple two-variable cases.

4.1.2.1 PDF/CDF Propagation

A random variable can be described by its probability density function (PDF) and cumulative distribution function (CDF). The timing analysis algorithms proposed in [3, 26] estimate the PDF's and the CDF's of arrival times and directly propagate the random distributions through SUM($\bullet$) and MAX($\bullet$). These algorithms can handle random variations with arbitrary distributions (e.g., not limited to Normal distributions);

however, all random distributions must be mutually independent such that the SUM(•) and MAX(•) operations are easy to evaluate.

Given two independent random variables x and y, the corresponding statistical SUM(•) and MAX(•) can be implemented using the following equations [3, 26]:

$$\mathrm{pdf}_{x+y}(t) = \int \mathrm{pdf}_x(t-\tau) \cdot \mathrm{pdf}_y(\tau) \cdot d\tau \tag{4.1}$$

$$\mathrm{cdf}_{\mathrm{MAX}(x,y)}(t) = \mathrm{cdf}_x(t) \cdot \mathrm{cdf}_y(t), \tag{4.2}$$

where pdf(•) and cdf(•) denote the probability density function and the cumulative distribution function, respectively. If the random variables are correlated, however, their joint probability density function cannot be simply represented as the product of all marginal probability density functions. In this case, calculating the probability distribution for SUM(•) and MAX(•) involves the numerical integration of a multi-dimensional PDF/CDF for which the computational complexity increases exponentially with the total number of random variables, thereby quickly making the computation task infeasible.

In practice, arrival times become correlated because of path-sharing (i.e., re-convergent fan-out) and/or process-sharing (i.e., correlated process variations). Figure 4.3 illustrates two simple examples for correlated arrival times. In Figure 4.3(a), the total delay is equal to

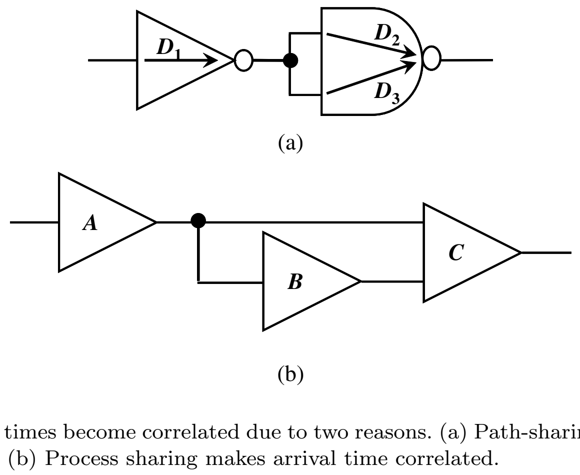


Fig. 4.3 Arrival times become correlated due to two reasons. (a) Path-sharing makes arrival time correlated. (b) Process sharing makes arrival time correlated.

MAX$(D_1 + D_2, D_1 + D_3)$, where $D_1 + D_2$ and $D_1 + D_3$ share the same component D_1 and, therefore, are correlated. On the other hand, process-sharing can occur for both inter-die and intra-die variations. Inter-die variations are shared by all logic gates on the same die, thereby making gate delays correlated. Intra-die variations contain long-range correlated components and their correlation typically decays over distance. Taking Figure 4.3(b) as an example, the delays of A and B have a higher correlation than the delays of A and C.

4.1.2.2 Delay Model Propagation

To address the aforementioned correlation problem, several statistical timing analysis techniques [18, 19, 113, 115, 124, 125] are recently developed where delay variations are approximated as the linear models:

$$x = B_x^T \cdot \varepsilon + C_x = \sum_{i=1}^{N} B_{x_i} \cdot \varepsilon_i + C_x \tag{4.3}$$

$$y = B_y^T \cdot \varepsilon + C_y = \sum_{i=1}^{N} B_{y_i} \cdot \varepsilon_i + C_y \tag{4.4}$$

or the quadratic models:

$$\begin{aligned} x &= \varepsilon^T \cdot A_x \cdot \varepsilon + B_x^T \cdot \varepsilon + C_x \\ &= \sum_{i=1}^{N} \sum_{j=1}^{i} A_{x_{ij}} \cdot \varepsilon_i \cdot \varepsilon_j + \sum_{i=1}^{N} B_{x_i} \cdot \varepsilon_i + C_x \qquad (4.5) \\ y &= \varepsilon^T \cdot A_y \cdot \varepsilon + B_y^T \cdot \varepsilon + C_y \\ &= \sum_{i=1}^{N} \sum_{j=1}^{i} A_{y_{ij}} \cdot \varepsilon_i \cdot \varepsilon_j + \sum_{i=1}^{N} B_{y_i} \cdot \varepsilon_i + C_y, \qquad (4.6) \end{aligned}$$

where $C_x, C_y \in R$ are the constant terms, $B_x, B_y \in R^N$ contain the linear coefficients, $A_x, A_y \in R^{N \times N}$ contain the quadratic coefficients, $\varepsilon = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N]^T$ contains a set of random variables to model process variations and N is the total number of these random variables. For most practical applications, $\{\varepsilon_i;\ i = 1, 2, \ldots, N\}$ can be modeled as independent standard Normal distributions. Note that correlated

Normal random variables can be decomposed to uncorrelated random variables by principal component analysis.

Given the delay models in (4.3)–(4.6), the basic operation in statistical timing analysis is to evaluate SUM(x,y) or MAX(x,y) and approximate the result as a new delay model of $\{\varepsilon_i;\ i = 1,2,\ldots,N\}$. In other words, unlike the PDF/CDF propagation discussed in Section 4.1.2.1 where only independent probability distributions are considered, the delay model propagation approximates all arrival times as functions of $\{\varepsilon_i;\ i = 1,2,\ldots,N\}$ such that the correlation information can be preserved.

For the linear delay models in (4.3) and (4.4), the SUM$(\bullet)$ operation can be easily handled by

$$x + y = (B_x + B_y)^T \cdot \varepsilon + (C_x + C_y). \tag{4.7}$$

A similar formulation can be derived for the quadratic delay models in (4.5) and (4.6):

$$x + y = \varepsilon^T \cdot (A_x + A_y) \cdot \varepsilon + (B_x + B_y)^T \cdot \varepsilon + (C_x + C_y). \tag{4.8}$$

In (4.7) and (4.8), since the SUM$(\bullet)$ operator is linear, adding two linear (or quadratic) models results in a new linear (or quadratic) model. The MAX$(\bullet)$ operator, however, is nonlinear and is much more difficult to approximate. In Sections 4.1.2.3 and 4.1.2.4, we will show various algorithms to efficiently perform statistical MAX$(\bullet)$ operation.

4.1.2.3 First-Order (Linear) MAX(•) Approximation

To perform the statistical operation MAX(x,y), the authors of [18, 113] propose to find an approximated linear model for $z \approx \text{MAX}(x,y)$ by matching the first and second order moments. Assume that x and y are approximated as the linear delay models (4.3) and (4.4), respectively. If MAX(x,y) is approximated as a linear combination of x and y, $z \approx$ MAX(x,y) can be expressed as the following linear function:

$$z = B_z^T \cdot \varepsilon + C_z = \sum_{i=1}^{N} B_{z_i} \cdot \varepsilon_i + C_z, \tag{4.9}$$

where $C_z \in R$ is the constant term and $B_z \in R^N$ contains the linear coefficients. It is easy to verify the following relations:

$$E(z) = C_z + \sum_{i=1}^{N} B_{z_i} \cdot E(\varepsilon_i) = C_z \tag{4.10}$$

$$E(z \cdot \varepsilon_i) = E(C_z \cdot \varepsilon_i) + \sum_{j=1}^{N} B_{z_i} \cdot E(\varepsilon_i \cdot \varepsilon_j) = B_{z_i}, \tag{4.11}$$

where $E(\bullet)$ denotes the expected value. The mean and covariance values in (4.10) and (4.11) can be calculated by using the formula derived in [23]. Algorithm 4.1 summarizes the major steps for the aforementioned linear MAX($\bullet$) approximation.

Algorithm 4.1 linear MAX($\bullet$) approximation for two correlated Normal random variables.

(1) Start from the linear delay models for x and y in (4.3) and (4.4).

(2) Calculate the first and second order moments for x and y:

$$\mu_x = C_x \tag{4.12}$$

$$\mu_y = C_y \tag{4.13}$$

$$\sigma_x = \sqrt{\sum_{i=1}^{N} B_{x_i}^2} \tag{4.14}$$

$$\sigma_y = \sqrt{\sum_{i=1}^{N} B_{y_i}^2} \tag{4.15}$$

$$\rho_{xy} = \frac{1}{\sigma_x \cdot \sigma_x} \cdot \sum_{i=1}^{N} B_{x_i} \cdot B_{y_i}. \tag{4.16}$$

(3) Calculate the coefficients:

$$\alpha = \sqrt{\sigma_x^2 + \sigma_y^2 - 2 \cdot \rho_{xy} \cdot \sigma_x \cdot \sigma_y} \tag{4.17}$$

$$\beta = \frac{\mu_x - \mu_y}{\alpha}. \tag{4.18}$$

(4) Calculate the constant term C_z:

$$C_z = E(z) = \mu_x \cdot \Phi(\beta) + \mu_y \cdot \Phi(-\beta) + \alpha \cdot \varphi(\beta), \quad (4.19)$$

where $\phi(\bullet)$ and $\Phi(\bullet)$ are the probability density function and the cumulative distribution function of standard Normal distribution, respectively:

$$\varphi(x) = \frac{1}{\sqrt{2\pi}} \cdot e^{\frac{-x^2}{2}} \quad (4.20)$$

$$\Phi(x) = \int_{-\infty}^{x} \varphi(x) \cdot dx. \quad (4.21)$$

(5) For each $i = \{1,2,\ldots,N\}$
(6) Calculate the linear coefficient B_{z_i}:

$$B_{z_i} = E(z \cdot \varepsilon_i) = B_{x_i} \cdot \Phi(\beta) + B_{y_i} \cdot \Phi(-\beta). \quad (4.22)$$

(7) End For.
(8) Substituting (4.19) and (4.22) into (4.9) yields the approximated linear model for $z \approx \mathrm{MAX}(x,y)$.

In addition to the aforementioned technique based on moment matching, another approach for MAX($\bullet$) approximation is based on the *tightness probability* proposed in [115]:

$$\mathrm{MAX}(x,y) \approx z = P(x \geq y) \cdot x + P(y \geq x) \cdot y + C_z, \quad (4.23)$$

where $P(\bullet)$ denotes the probability and the constant term C_z is determined by matching the mean value between MAX(x,y) and z. In (4.23), $P(x \geq y)$ and $P(y \geq x)$ are referred to as the tightness probabilities of x and y, respectively. Intuitively, the linear model in (4.23) is the weighted sum of x and y. The weight for x (or y) is large if x (or y) is likely to be greater than y (or x).

If the random variables x and y in (4.23) are Normal, it can be proven that the tightness probability formulation in (4.23) and the moment-matching formulation in (4.10), (4.11) are exactly equivalent. The concept of tightness probability is related to the first-order Taylor expansion [53]. To show this relation, the authors of [53] prove that the tightness probability is equal to the first-order statistical sensitivity:

$$P(x \geq y) = \frac{\partial\{E[\mathrm{MAX}(x,y)]\}}{\partial\{E[x]\}} \tag{4.24}$$

$$P(y \geq x) = \frac{\partial\{E[\mathrm{MAX}(x,y)]\}}{\partial\{E[y]\}}. \tag{4.25}$$

In other words, although the MAX(•) operator is not analytical (i.e., does not have continuous derivatives), it can be statistically approximated as the form of (4.23)–(4.25) that is similar to the traditional Taylor expansion. Therefore, the linear approximation in (4.23) is referred to as the *first-order statistical Taylor expansion*. In Section 4.1.2.4, we will further show that the aforementioned statistical Taylor expansion can be extended to second order to achieve better approximation accuracy.

4.1.2.4 Second-Order (Quadratic) MAX(•) Approximation

Recently, various quadratic approximations have been proposed to handle the MAX(•) operator [55, 124]. These techniques are more accurate, but also more expensive, than a simple linear approximation. In practice, quadratic MAX(•) approximations should be selectively applied, depending on the accuracy and complexity requirements of a specific application.

Assume that MAX(x,y) is approximated as a quadratic model:

$$\begin{aligned}\mathrm{MAX}(x,y) \approx z &= \varepsilon^T \cdot A_z \cdot \varepsilon + B_z^T \cdot \varepsilon + C_z \\ &= \sum_{i=1}^{N}\sum_{j=1}^{i} A_{z_{ij}} \cdot \varepsilon_i \cdot \varepsilon_j + \sum_{i=1}^{N} B_{z_i} \cdot \varepsilon_i + C_z,\end{aligned} \tag{4.26}$$

where $C_z \in R$ is the constant term, $B_z \in R^N$ contains the linear coefficients, and $A_z \in R^{N\times N}$ contains the quadratic coefficients. The random variables $\{\varepsilon_i;\ i = 1,2,\ldots,N\}$ are used to model process variations and they are independent standard Normal distributions. Given the quadratic model in (4.26), it is easy to verify the following equations [124]:

$$E(z) = \sum_{i=1}^{N} A_{z_{ii}} + C_z \tag{4.27}$$

$$E(z \cdot \varepsilon_i) = B_{z_i} \tag{4.28}$$

$$E(z \cdot \varepsilon_i \cdot \varepsilon_j) = A_{z_{ij}} \quad (i \neq j) \tag{4.29}$$

$$E(z \cdot \varepsilon_i^2) = 3 \cdot A_{z_{ii}} + \sum_{j=1, j\neq i}^{N} A_{z_{jj}} + C_z. \tag{4.30}$$

If the moment values in (4.27)–(4.30) are available, the model coefficients in (4.26) can be solved via a set of linear equations [124]. Such a quadratic MAX(•) approximation is referred to as the moment-matching technique.

Another method for quadratic MAX(•) approximation is based on the *second-order statistical Taylor expansion* proposed in [55]. It starts from converting a two-variable MAX(•) operator to a single-variable one:

$$\mathrm{MAX}(x, y) = x + \mathrm{MAX}(0, t), \tag{4.31}$$

where

$$t = y - x. \tag{4.32}$$

Next, a second-order statistical Taylor expansion will be utilized to approximate the single-variable operator $\mathrm{MAX}(0, t)$.

Extending the first-order statistical Taylor expansion in (4.23)–(4.25) to second order and expanding $\mathrm{MAX}(0, t)$ at the expansion point $E[t]$ yield:

$$\mathrm{MAX}(0, t) = 0.5 \cdot \lambda_2 \cdot \{t - E[t]\}^2 + \lambda_1 \cdot \{t - E[t]\} + \lambda_0, \tag{4.33}$$

where the linear and quadratic coefficients λ_1 and λ_2 are determined by the statistical derivatives:

$$\lambda_1 = \frac{d\{E[\mathrm{MAX}(0, t)]\}}{d\{E[t]\}} \tag{4.34}$$

$$\lambda_2 = \frac{d^2\{E[\mathrm{MAX}(0, t)]\}}{d\{E[t]\}^2} = \frac{d\lambda_1}{d\{E[t]\}} \tag{4.35}$$

and the constant term λ_0 is determined by matching the mean value:

$$\lambda_0 = E[\text{MAX}(0,t)] - 0.5 \cdot \lambda_2 \cdot E[\{t - E[t]\}^2]. \tag{4.36}$$

Next, we show how to compute the coefficients λ_0, λ_1, and λ_2 in (4.34)–(4.36) efficiently.

As summarized in Section 4.1.2.3, the first-order derivative in (4.34) is equal to the probability:

$$\lambda_1 = \frac{d\{E[\text{MAX}(0,t)]\}}{d\{E[t]\}} = P(t \geq 0) = 1 - \text{CDF}_t(0), \tag{4.37}$$

where $\text{CDF}_t(\bullet)$ stands for the cumulative distribution function of the random variable t. If both x and y in (4.32) are approximated as quadratic models of the random variables $\{\varepsilon_i;\ i = 1, 2, \ldots, N\}$, t is equal to $y - x$ and, therefore, is also a quadratic function of $\{\varepsilon_i;\ i = 1, 2, \ldots, N\}$:

$$t(\varepsilon) = \varepsilon^T \cdot A_t \cdot \varepsilon + B_t^T \cdot \varepsilon + C_t, \tag{4.38}$$

where A_t, B_t, and C_t are the model coefficients. Given (4.38), the cumulative distribution function of t can be extracted using the algorithms discussed in Chapter 3.

Substituting (4.37) into (4.35) yields:

$$\lambda_2 = \frac{d[1 - \text{CDF}_t(0)]}{d\{E[t]\}}. \tag{4.39}$$

To calculate the derivative value in (4.39), we re-write t as:

$$t = \mu + \delta, \tag{4.40}$$

where μ is the mean value of t and $\delta = t - \mu$ is a random variable with zero mean. Substituting (4.40) into (4.39) yields:

$$\begin{aligned} \lambda_2 &= \frac{d\,[1 - \text{CDF}_{\mu+\delta}(0)]}{d\mu} = \frac{d\,[1 - \text{CDF}_\delta(-\mu)]}{d\mu} \\ &= \text{PDF}_\delta(-\mu) = \text{PDF}_{\mu+\delta}(0) = \text{PDF}_t(0), \end{aligned} \tag{4.41}$$

where $\text{PDF}_t(\bullet)$ stands for the probability density function of the random variable t. Since t is represented as the quadratic function in (4.38),

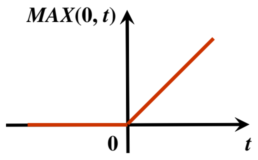


Fig. 4.4 The single variable function $\mathrm{MAX}(0,t)$.

PDF_t can be extracted by the algorithms discussed in Chapter 3 to calculate λ_2 in (4.41).

Importantly, the quadratic coefficient λ_2 in (4.41) has two interesting properties:

- $\lambda_2 = \mathrm{PDF}_t(0)$ *is non-negative.* Intuitively, as shown in Figure 4.4, the function $\mathrm{MAX}(0,t)$ is convex and, therefore, the quadratic model coefficient should be non-negative [13].
- λ_2 *indicates the nonlinearity of* $\mathrm{MAX}(0,t)$. Considering the first two cases in Figure 4.5, $\mathrm{MAX}(0,t)$ can be accurately approximated as linear models, i.e., $\mathrm{MAX}(0,t) \approx 0$ and $\mathrm{MAX}(0,t) \approx t$, respectively. This is consistent with the fact that $\mathrm{PDF}_t(0) \approx 0$ in both cases. In the third case of

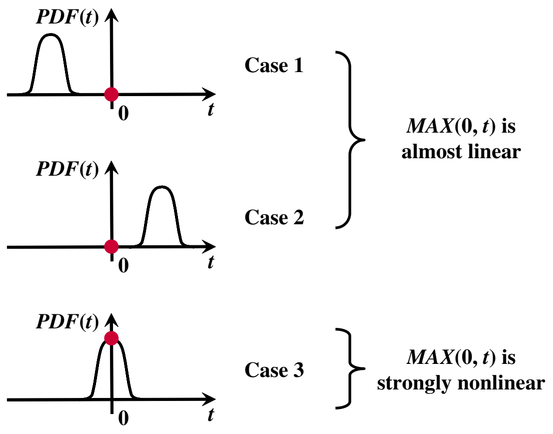


Fig. 4.5 Three different cases for the $\mathrm{MAX}(0,t)$ approximation.

Figure 4.5, however, MAX$(0,t)$ is strongly nonlinear, corresponding to a non-zero PDF$_t(0)$.

After λ_1 and λ_2 are extracted, computing the constant term λ_0 in (4.36) requires further knowing $E[\mathrm{MAX}(0,t)]$ and $E[\{t - E[t]\}^2]$. $E[\mathrm{MAX}(0,t)]$ can be calculated using the following one-dimensional numerical integration:

$$E\left[\mathrm{MAX}(0,t)\right] = \int_0^{+\infty} \tau \cdot \mathrm{PDF}_t(\tau) \cdot d\tau. \tag{4.42}$$

Since t is a quadratic function of $\{\varepsilon_i;\ i = 1,2,\ldots,N\}$ shown in (4.38), its second-order central moment $E[\{t - E[t]\}^2]$ can be determined by the following analytical equation [125]:

$$E\left[\{t - [t]\}^2\right] = B_t^T \cdot B_t + 2 \cdot \mathrm{TRACE}(A_t \cdot A_t), \tag{4.43}$$

where TRACE$(\bullet)$ represents the trace of a matrix (the sum of all diagonal elements). Substituting (4.41)–(4.43) into (4.36) yields the constant term λ_0.

After the coefficients λ_0, λ_1, and λ_2 are known, MAX$(0,t)$ in (4.33) can be approximated as a quadratic function of the random variables $\{\varepsilon_i;\ i = 1,2,\ldots,N\}$ by substituting (4.38) into (4.33) and ignoring all high-order terms.

To demonstrate the efficacy of the second-order statistical Taylor expansion, we consider a simple example to approximate MAX(x,y) where $x \sim N(0,1/9)$ and $y \sim N(0,1)$ are independent and Normal. Figure 4.6 shows the probability density functions of the random variables x and y. In this example, MAX(x,y) is strongly nonlinear, because the probability density functions of x and y are significantly overlapped. It, in turn, allows us to test the efficacy of the second-order statistical Taylor expansion and compare it with a simple linear MAX$(\bullet)$ approximation.

Three different approaches, namely, the linear approximation, the second-order statistical Taylor expansion and the Monte Carlo analysis with 10^4 random samples, are applied to estimate the probability distribution of MAX(x,y). Figure 4.7 shows the probability density functions estimated by these techniques. In this example, the distribution of MAX(x,y) is not symmetric due to the nonlinearity. The simple

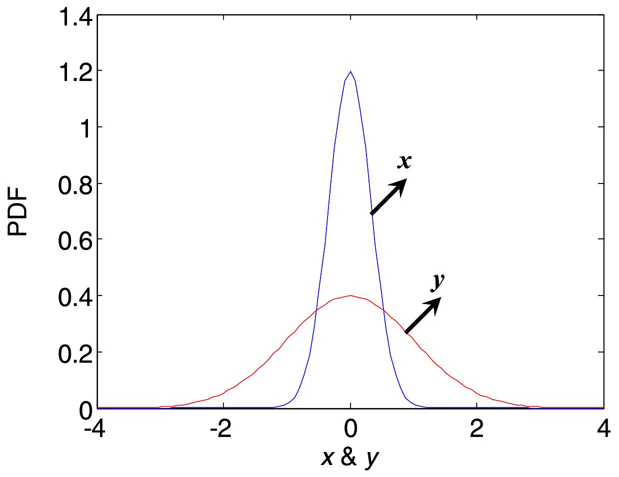


Fig. 4.6 The probability density functions of x and y.

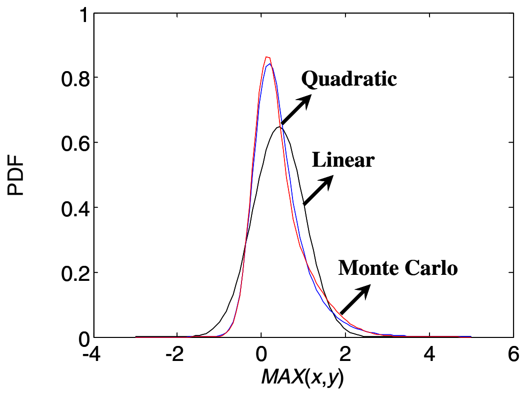


Fig. 4.7 The estimated probability density functions of MAX(x, y).

linear approximation cannot capture such a non-zero skewness and, therefore, results in large approximation error, especially at both tails of the probability density function. The quadratic MAX$(\bullet)$ approximation, however, accurately models the non-zero skewness by including second-order terms.

4.2 Statistical Timing Sensitivity Analysis

While we discussed many statistical timing analysis algorithms in the previous sub-section, a new methodology for using timing analysis results to guide timing optimization and explore the trade-off between performance, yield and cost is required in the statistical domain. In nominal timing analysis, critical path and slack are two important concepts that have been widely utilized for timing optimization, but the inclusion of large-scale process variations renders these concepts obsolete.

First, the delay of each path is a random variable, instead of a deterministic value, in statistical timing analysis. As such, every path can be critical (i.e., have the maximal delay) with certain probability. Second, the slacks at all nodes are random variables that are statistically coupled. The overall timing performance is determined by the distributions of all these slacks, as well as their *correlations*. It implies that individual slack at a single node is *not* meaningful and cannot be utilized as a criterion to guide timing optimization. Therefore, the traditional critical path and slack definitions are no longer valid, and new criteria are required to accommodate the special properties of statistical timing analysis/optimization.

In this sub-section, we describe a new concept of *statistical timing sensitivity* to guide timing optimization of logic circuits with large-scale parameter variations. We define the statistical sensitivities for both paths and arcs. The *path sensitivity* provides a theoretical framework from which we can study and analyze timing constraints under process variations. The *arc sensitivity* is an efficient metric to assess the criticality of each arc in the timing graph, which is useful for timing optimization. We prove that the path sensitivity is exactly equal to the probability that a path is critical, and the arc sensitivity is exactly equal to the probability that an arc sits on the critical path. The path sensitivity and the arc sensitivity discussed in this sub-section are theoretically equivalent to the path criticality and the edge criticality proposed in [115, 120]. More details on path criticality and edge criticality can be found in Section 4.3.

4.2.1 Statistics of Slack and Critical Path

We first give a comprehensive study on slack and critical path in statistical timing analysis. We will highlight the differences between nominal and statistical timing analyses and explain the reasons why the traditional concepts of slack and critical path become ineffective in the presence of process variations.

4.2.1.1 Slack

In nominal timing analysis, slack is utilized as a metric to measure how tightly the timing constraint is satisfied. A negative slack means that the timing constraint has not been met, while a (small) positive slack means that the timing constraint has been (marginally) satisfied. In statistical timing analysis, however, it is difficult to make such a straightforward judgment, since all slacks are random variables instead of deterministic values. For instance, Figure 4.8 shows two slack distributions computed from statistical timing analysis. The node V_1 presents a larger probability that the slack is positive than the node V_2. However, the worst-case (i.e., the smallest) slack at V_1 is more negative than that at V_2. In this case, it is hard to conclude which slack distribution is better using a simple criterion.

More importantly, the slacks in a timing graph are statistically coupled and must be considered concurrently to determine the timing performance. In nominal timing analysis, it is well-known that the timing constraint is satisfied if and only if all slacks in the timing graph are positive. In statistical timing analysis, this condition can be stated as

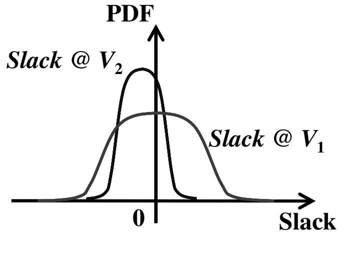


Fig. 4.8 Two slack distributions in statistical timing analysis.

follows: the probability that the timing constraint is satisfied is equal to the probability that all slacks are positive:

$$P(\text{Satisfy Timing Constraint}) = P[\text{Slack}_{V_1} \geq 0 \ \ \& \ \ \text{Slack}_{V2} \geq 0 \ \cdots]. \tag{4.44}$$

Studying (4.44), one would find that such a probability depends on all slack distributions, as well as their *correlations*. Unlike the nominal timing analysis where slacks are deterministic without correlations, knowing individual slack distributions in statistical timing analysis is insufficient to assess the timing performance. The probability in (4.44) cannot be accurately estimated if the slack correlations are ignored. The above analysis implies an important fact that *an individual slack distribution at one node may not be meaningful in statistical timing analysis.*

However, it should be noted that there exist some “important” nodes in a timing graph whose slacks have special meanings. Given a timing graph, we define a node V_{IN} as an *important node* if all paths in the timing graph pass V_{IN}. Based on this definition, the source node and the sink node are two important nodes in any timing graph, since all paths start from the source node and terminate at the sink node. In some special timing graphs, it is possible to find other important nodes. For example, the node e in Figure 4.1 is an important node by this definition. The importance of the node is that, if V_{IN} is an important node, the probability in (4.44) can be uniquely determined by the slack at V_{IN}:

$$P(\text{Satisfy Timing Constraint}) = P[\text{Slack}_{V_{\text{IN}}} \geq 0]. \tag{4.45}$$

The physical meaning of (4.45) can be intuitively explained by the concept of Monte Carlo analysis. When a timing graph is simulated by Monte Carlo analysis, a delay sample (i.e., a set of deterministic delay values for all timing arcs) is drawn from the random variable space for each Monte Carlo run. The probability $P(\text{Satisfy Timing Constraint})$ is equal to Num_1 (the number of samples for which the timing constraint is satisfied) divided by Num (the total number of Monte Carlo samples). Similarly, the probability $\text{Slack}_{V_{\text{IN}}} \geq 0$ is equal to Num_2 (the number of samples for which the slack at V_{IN} is positive) divided by Num. In each

Monte Carlo run, the timing constraint is failed if and only if there is a path P whose delay is larger than the specification. In this case, the slack at V_{IN} must be negative since all paths pass the important node V_{IN} and, therefore, V_{IN} must sit on the path P. The above analysis implies that Num_1 is equal to Num_2, yielding Equation (4.45).

Equations (4.44) and (4.45) indicate another difference between nominal and statistical timing analyses. In nominal timing analysis, the slack at any node along the critical path uniquely determines the timing performance. In statistical timing analysis, however, only the slack at an important node uniquely determines the timing performance. Compared with the critical path nodes in nominal timing analysis, important nodes belong to a much smaller subset, since they must be included in all paths in a timing graph.

Following (4.45), it is sufficient to check the slacks only for important nodes, e.g., the source node or the sink node. Therefore, using the concept of important node simplifies the timing verification procedure. This conclusion is also consistent with our intuition: the timing performance is determined by the maximal delay from the source node to the sink node. Therefore, the slacks at these two nodes are of most interest for timing verification.

4.2.1.2 Critical Path

Similar to slack, there are key differences between nominal and statistical timing analyses on critical path. First, given a timing graph, the maximal delay from the source node to the sink node can be expressed as

$$D = \text{MAX}(D_{P1}, D_{P2}, \ldots), \tag{4.46}$$

where D_{Pi} is the delay of the ith path. In nominal timing analysis, $D = D_{Pi}$ if and only if the path P_i is critical. In statistical timing analysis, however, every path can be critical with certain probability. Although it is possible to define the *most critical path* as the path P_i that has the largest probability to be critical, the maximal circuit delay in (4.46) must be determined by all paths, instead of the most critical path only.

Second, the most critical path is difficult to identify in statistical timing analysis. In nominal timing analysis, the critical path can be identified using slack since all nodes along the critical path have the same (smallest) slack. In statistical timing analysis, however, this property is no longer valid and all slacks are random variables.

Finally, but most importantly, the critical path concept is not so helpful for statistical timing optimization. In nominal case, the gates and interconnects along the critical (or non-critical) path are repeatedly selected for up (or down) sizing. This strategy is becoming ineffective under process variations. One important reason is that many paths may have similar probabilities to be critical and all these paths must be selected for timing optimization. Even in nominal case, many paths in a timing graph can be equally critical, which is so-called "slack wall." This multiple-critical-path problem is more pronounced in statistical timing analysis, since more paths can have overlapped delay distributions due to large-scale process variations. In addition to this multiple-critical-path problem, we will demonstrate in Section 4.2.2 that selecting the gates and interconnects along the most critical (or least critical) path for up (or down) sizing may not be the best choice under a statistical modeling assumption.

4.2.2 Concept of Statistical Timing Sensitivity

In this sub-section, we define the concept of statistical timing sensitivity. Two different sensitivities, i.e., path sensitivity and arc sensitivity, are discussed. *Path sensitivity* provides a theoretical framework to study and analyze timing constraints under process variations. *Arc sensitivity* provides an efficient criterion to select the most critical gates/interconnects for timing optimization.

4.2.2.1 Path Sensitivity

In nominal timing analysis, the critical path is of great interest since it uniquely determines the maximal circuit delay. If the delay of the critical path is increased (or decreased) by a small perturbation ε, the maximal circuit delay is increased (or decreased) by ε correspondingly. Therefore, given the maximal circuit delay D in (4.46), the dependence between D and the individual path delay D_{P_i} can be mathematically represented as the path sensitivity:

$$S_{P_i}^{\text{Path}} = \frac{\partial D}{\partial D_{P_i}} = \begin{cases} 1 & (\text{If } P_i \text{ is critical}) \\ 0 & (\text{Otherwise}) \end{cases}. \tag{4.47}$$

From the sensitivity point of view, a critical path is important since it has non-zero sensitivity and all other non-critical paths have zero sensitivity. The maximal circuit delay can be changed if and only if the critical path delay is changed. This is the underlying reason why critical path is important for timing optimization. It is the sensitivity (instead of the critical path itself) that provides an important criterion to guide timing optimization. A path is more (or less) important if it has a larger (or smaller) path sensitivity.

In statistical timing analysis, all path delays are random variables. Although directly computing sensitivity between two random variables seems infeasible, the path sensitivity can be defined by their expected values (i.e., moments). One simple definition for path sensitivity is to use the first order moment, i.e.,

$$S_{P_i}^{\text{Path}} = \frac{\partial E(D)}{\partial E(D_{P_i})}. \tag{4.48}$$

The path sensitivity in (4.48) models the mean value dependence between the maximal circuit delay D and the individual path delay D_{Pi}. The path sensitivity in (4.48) has several important properties. The detailed proofs of the following theorems can be found in [53].

Theorem 4.1. The path sensitivity in (4.48) satisfies:

$$\sum_i S_{P_i}^{\text{Path}} = 1. \tag{4.49}$$

Theorem 4.2. Given the maximal circuit delay $D = \text{MAX}(D_{P_1}, D_{P_2}, \ldots)$ where D_{P_i} is the delay of the ith path, if the probability $P[D_{P_i} = \text{MAX}(D_{P_j}, j \neq i)]$ is equal to 0, then *the path sensitivity in* (4.48) *is equal to the probability that the path P_i is critical*, i.e.,

$$S_{P_i}^{\text{Path}} = P(D_{P_i} \geq D_{P_1} \ \& \ D_{P_i} \geq D_{P_2} \ \& \ \cdots). \tag{4.50}$$

4.2.2.2 Arc Sensitivity

In nominal timing optimization, the gates and interconnects along the critical path are important, since the maximal circuit delay is sensitive to their delays. Following this reasoning, the importance of a given gate or interconnect can be assessed by the following arc sensitivity:

$$S_{A_i}^{\text{Arc}} = \frac{\partial D}{\partial D_{A_i}} = \sum_k S_{P_k}^{\text{Path}} \cdot \frac{\partial D_{P_k}}{\partial D_{A_i}} = \begin{cases} 1 & (A_i \text{ is on critical path}) \\ 0 & (\text{Otherwise}) \end{cases}, \tag{4.51}$$

where D is the maximal circuit delay given in (4.46), D_{Ai} denotes the gate/interconnect delay associated with the ith arc, and D_{Pk} represents the delay of the kth path. In (4.51), the path sensitivity $S_{P_k}^{\text{Path}}$ is non-zero (i.e., equal to 1) if and only if the kth path P_k is critical. In addition, the derivative $\partial D_{Pk}/\partial D_{A_i}$ is non-zero (i.e., equal to 1) if and only if the ith arc A_i sits on the kth path P_k, since the path delay D_{P_k} is equal to the sum of all arc delays D_{A_i}'s that belong to this path. These observations yield the conclusion that the arc sensitivity $S_{A_i}^{\text{Arc}}$ is non-zero if and only if A_i is on the critical path. The arc sensitivity explains why the gates and interconnects along the critical path are important for timing optimization. A gate/interconnect is more (or less) important if it has a larger (or smaller) arc sensitivity.

The aforementioned sensitivity concept can be extended to statistical timing analysis. In statistical case, we define the arc sensitivity using the first order moment:

$$S_{A_i}^{\text{Arc}} = \frac{\partial E(D)}{\partial E(D_{A_i})}. \tag{4.52}$$

The arc sensitivity in (4.52) has the following important property.

Theorem 4.3. Let D_{P_i} be the delay of the ith path. If the probability $P[D_{P_i} = \text{MAX}(D_{P_j}, j \neq i)] = 0$ for any $\{i = 1, 2, \ldots\}$, then the arc sensitivity in (4.52) is equal to:

$$S_{A_i}^{\text{Arc}} = \sum_{A_i \in P_k} S_{P_k}^{\text{Path}}. \tag{4.53}$$

The detailed proof of Theorem 4.3 can be found in [53]. Remember that $S_{P_k}^{\text{Path}}$ is equal to the probability that the kth path P_k

is critical (see Theorem 4.2). Therefore, *the arc sensitivity defined in* (4.52) *is exactly equal to the probability that the arc sits on the critical path.*

The arc sensitivity defined in (4.52) provides an effective criterion to select the most important gates and interconnects for up/down sizing. Roughly speaking, for statistical timing optimization, the gates and interconnects with large arc sensitivities are critical to the maximal circuit delay and in general should be up-sized to improve circuit speed, while the others with small arc sensitivities can be down-sized to save chip area and power consumption. Next, using the concept of arc sensitivity, we explain the reason why repeatedly selecting the gates and interconnects along the most critical (or least critical) path for up (or down) sizing can be ineffective in statistical case.

Consider a simple timing graph including three paths, as shown in Figure 4.9. Assume that the path sensitivity $S_{P_1}^{\text{Path}} = S_{P_2}^{\text{Path}} = 0.3$ and $S_{P_3}^{\text{Path}} = 0.4$. Therefore, P_3 is the most critical path since it has the largest path sensitivity and is most likely to have the maximal delay. Using the traditional concept of critical path, the arc A_2 should be selected for up-sizing in order to reduce the circuit delay. However, according to Theorem 4.3, it is easy to verify that $S_{A_1}^{\text{Arc}} =_{P_1}^{\text{Path}} + S_{P_2}^{\text{Path}} = 0.6$ and $S_{A_2}^{\text{Arc}} = S_{P_3}^{\text{Path}} = 0.4$. The arc A_1 has a more significant impact on the maximal circuit delay and should be selected for up-sizing, although it does not sit on the most critical path. In this example, using the traditional concept of critical path selects the wrong arc,

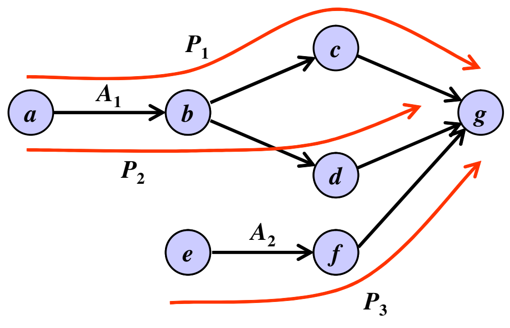


Fig. 4.9 A simple timing graph to illustrate the application of arc sensitivity.

since it does not consider the non-zero path sensitivities of other less critical paths. These non-zero sensitivities make it possible that changing an arc delay can change the maximal circuit delay through multiple paths. In Figure 4.9, the arc A_1 can change the maximal circuit delay through two paths P_1 and P_2, while the arc A_2 can change the maximal circuit delay only through one path P_3. Therefore, the arc A_1 eventually becomes more critical than A_2, although neither P_1 nor P_2 is the most critical path.

4.2.2.3 Summary of Statistical Timing Sensitivity

We have defined two statistical timing sensitivities (i.e., path sensitivity and arc sensitivity) and shown the theoretical link between probability and sensitivity. The aforementioned sensitivity-based framework has three unique properties:

- *Distribution-independent.* The theoretical results for path sensitivity and arc sensitivity do not rely on specific probability distributions for gate/interconnect delays and arrival times.
- *Correlation-aware.* The aforementioned sensitivity-based framework does not rely on any assumption of statistical independence and it can handle correlated arrival times. Theorems 4.1–4.3 are valid, even if the path/arc delays are correlated.
- *Computation-efficient.* Sensitivity values can be efficiently computed, as will be further discussed in Section 4.2.3.

4.2.3 Computation of Statistical Timing Sensitivity

In this sub-section, we describe the numerical algorithm for computing arc sensitivities. We first develop the sensitivity equations for two atomic operations: SUM(•) and MAX(•). Next, we show how to propagate sensitivity values throughout the timing graph, using a single breadth-first graph traversal.

The sensitivity analysis should be conducted after the statistical timing analysis is complete. Therefore, we assume that the timing analysis results are already available before the sensitivity analysis begins. We further assume that the gate/interconnect delays and the arrival times can be approximated as Normal distributions.

4.2.3.1 Atomic Operations

A key function in statistical timing analysis is to propagate arrival times throughout a timing graph. In order to do that, two atomic operations are required, i.e., SUM(•) and MAX(•), as shown in Figure 4.2. Since multi-variable operations can be easily broken down into multiple two-variable cases, the remainder of this sub-section focuses on the sensitivity computation for SUM(•) and MAX(•) of two random variables, i.e., $z = x + y$ and $z = \text{MAX}(x,y)$ where x, y, and z are approximated as the linear delay models in (4.3), (4.4), and (4.9), respectively. The random variables $\{\varepsilon_i;\ i = 1,2,\ldots,N\}$ in these equations are used to model process variations and they are independent standard Normal distributions.

Given the operation $z = x + y$ or $z = \text{MAX}(x,y)$ where x, y, and z are in the form of (4.3), (4.4), and (4.9), we define the *sensitivity matrix* $Q_{z \leftarrow x}$ as,

$$Q_{z \leftarrow x} = \begin{bmatrix} \dfrac{\partial C_z}{\partial C_x} & \dfrac{\partial C_z}{\partial B_{x_1}} & \cdots & \dfrac{\partial C_z}{\partial B_{x_N}} \\ \dfrac{\partial B_{z_1}}{\partial C_x} & \dfrac{\partial B_{z_1}}{\partial B_{x_1}} & \cdots & \dfrac{\partial B_{z_1}}{\partial B_{x_N}} \\ \vdots & \vdots & \vdots & \vdots \\ \dfrac{\partial B_{z_N}}{\partial C_x} & \dfrac{\partial B_{z_N}}{\partial B_{x_1}} & \cdots & \dfrac{\partial B_{z_N}}{\partial B_{x_N}} \end{bmatrix}. \tag{4.54}$$

The sensitivity matrix $Q_{z \leftarrow y}$ can be similarly defined.

The sensitivity matrix in (4.54) provides the quantitative information that how much the coefficients C_z or $\{B_{z_i};\ i = 1,2,\ldots,N\}$ will be changed if there is a small perturbation on C_x or $\{B_{x_i};\ i = 1,2,\ldots,N\}$. Next, we derive the mathematical formulas of the sensitivity matrices for both SUM(•) and MAX(•) operations.

For the SUM(•) operation $z = x + y$, it is easy to verify that:

$$C_z = C_x + C_y \tag{4.55}$$

$$B_{z_i} = B_{x_i} + B_{y_i} \quad (i = 1,2,\ldots,N). \tag{4.56}$$

Therefore, the sensitivity matrix $Q_{z\leftarrow x}$ is an identity matrix.

For the MAX(•) operation $z = \text{MAX}(x,y)$, it can be proven that [53]:

$$\frac{\partial C_z}{\partial C_x} = \Phi(\beta) \tag{4.57}$$

$$\frac{\partial C_z}{\partial B_{x_i}} = \frac{\partial B_{z_i}}{\partial C_x} = \frac{\varphi(\beta)\cdot(B_{x_i} - B_{y_i})}{\alpha} \quad (i = 1,2,\ldots,N) \tag{4.58}$$

$$\frac{\partial B_{z_i}}{\partial B_{x_i}} = \Phi(\beta) - \frac{\beta\cdot\varphi(\beta)\cdot(B_{x_i} - B_{y_i})^2}{\alpha^2} \quad (i = 1,2,\ldots,N) \tag{4.59}$$

$$\frac{\partial B_{z_i}}{\partial B_{x_j}} = -\frac{\beta\cdot\varphi(\beta)\cdot(B_{x_i} - B_{y_i})\cdot(B_{x_j} - B_{y_j})}{\alpha^2} \quad (i,j = 1,2,\ldots,N;\ i \neq j), \tag{4.60}$$

where $\phi(\bullet)$ and $\Phi(\bullet)$ are the probability density function and the cumulative distribution function of standard Normal distribution defined in (4.20), (4.21) respectively, and the coefficients α and β are defined in (4.17), (4.18) respectively. Equations (4.57)–(4.60) can be derived by directly following the mathematic formulations in [23]. The sensitivity matrix $Q_{z\leftarrow y}$ can be similarly calculated, since both SUM(•) and MAX(•) are symmetric.

4.2.3.2 Sensitivity Propagation

Once the atomic operations are available, they can be applied to propagate the sensitivity matrices throughout the timing graph. Next, we use the simple timing graph in Figure 4.1 as an example to illustrate the key idea of sensitivity propagating. In this example, propagating the sensitivity matrices can be achieved through the following steps.

(1) Start from the MAX(•) operation at the sink node, i.e., $D = \text{MAX}[\text{AT}(f) + D(f,\text{sink}),\ \text{AT}(g) + D(g,\text{sink})]$, where D denotes the arrival time at the sink node (i.e., the maximal circuit delay), $\text{AT}(i)$ represents the arrival time at the

node i, and $D(i,j)$ stands for the delay of the arc $\langle i,j \rangle$. Compute the sensitivity matrices $Q_{D \leftarrow [\mathrm{AT}(f)+D(f,\mathrm{sink})]}$ and $Q_{D \leftarrow [\mathrm{AT}(g)+D(g,\mathrm{sink})]}$ using (4.57)–(4.60).

(2) Propagate $Q_{D \leftarrow [\mathrm{AT}(f)+D(f,\mathrm{sink})]}$ to the node f through the arc $\langle f, \mathrm{sink} \rangle$. Based on the chain rule of the derivatives, $Q_{D \leftarrow \mathrm{AT}(f)} = Q_{D \leftarrow [\mathrm{AT}(f)+D(f,\mathrm{sink})]} \cdot Q_{[\mathrm{AT}(f)+D(f,\mathrm{sink})] \leftarrow \mathrm{AT}(f)}$ and

$$\begin{aligned} Q_{D \leftarrow D(f,\mathrm{sink})} = {} & Q_{D \leftarrow [\mathrm{AT}(f)+D(f,\mathrm{sink})]} \\ & \cdot Q_{[\mathrm{AT}(f)+D(f,\mathrm{sink})] \leftarrow D(f,\mathrm{sink})} \end{aligned}$$

$Q_{[\mathrm{AT}(f)+D(f,\mathrm{sink})] \leftarrow \mathrm{AT}(f)}$ and $Q_{[\mathrm{AT}(f)+D(f,\mathrm{sink})] \leftarrow D(f,\mathrm{sink})}$ are identity matrices due to the SUM(•) operation.

(3) Similarly propagate $Q_{D \leftarrow [\mathrm{AT}(g)+D(g,\mathrm{sink})]}$ to the node g through the arc $\langle g, \mathrm{sink} \rangle$. Determine $Q_{D \leftarrow \mathrm{AT}(g)}$ and $Q_{D \leftarrow D(g,\mathrm{sink})}$.

(4) Propagate $Q_{D \leftarrow \mathrm{AT}(f)}$ and $Q_{D \leftarrow \mathrm{AT}(g)}$ to the node e, yielding $Q_{D \leftarrow D(e,f)} = Q_{D \leftarrow \mathrm{AT}(f)}$, $Q_{D \leftarrow D(e,g)} = Q_{D \leftarrow \mathrm{AT}(g)}$ and $Q_{D \leftarrow \mathrm{AT}(e)} = Q_{D \leftarrow \mathrm{AT}(f)} + Q_{D \leftarrow \mathrm{AT}(g)}$. Note that the out-degree of the node e is equal to two. Therefore, the sensitivity matrices $Q_{D \leftarrow \mathrm{AT}(f)}$ and $Q_{D \leftarrow \mathrm{AT}(g)}$ should be added together at the node e to compute $Q_{D \leftarrow \mathrm{AT}(e)}$. Its physical meaning is that a small perturbation on AT(e) can change the maximal circuit delay D through two different paths $\{e \rightarrow f \rightarrow \mathrm{sink}\}$ and $\{e \rightarrow g \rightarrow \mathrm{sink}\}$.

(5) Continue propagating the sensitivity matrices until the source node is reached.

In general, the sensitivity propagation involves a single breadth-first graph traversal from the sink node to the source node with successive matrix multiplications. The computational complexity of such a sensitivity propagation is *linear* in circuit size. After the sensitivity propagation, the sensitivity matrix $Q_{D \leftarrow D(i,j)}$ between the maximal circuit delay D and each arc delay $D(i,j)$ is determined. Based on these sensitivity matrices, the arc sensitivity can be easily computed by a quick post-processing. For example, the arc sensitivity defined in (4.52) is the (1, 1)th element in $Q_{D \leftarrow D(i,j)}$ (see the sensitivity matrix definition in

(4.54)), i.e.,

$$S^{\text{Arc}}_{<i,j>} = [1 \ 0 \ \cdots] \cdot Q_{D \leftarrow D(i,j)} \cdot [1 \ 0 \ \cdots]^T. \tag{4.61}$$

4.3 Statistical Timing Criticality

The path sensitivity and the arc sensitivity discussed in Section 4.2 are theoretically equivalent to the path criticality and the edge criticality proposed in [115, 120]. In other words, the path criticality is equal to the probability that a path is critical, and the edge criticality is equal to the probability that an edge sits on the critical path. In this sub-section, we describe the numerical algorithm proposed in [120] for computing edge criticality. The computational complexity of this algorithm is also *linear* in circuit size.

To efficiently compute the edge criticality, the authors of [120] define the following terminologies:

- The *edge slack* of an edge is the maximum delay of all paths going through the edge.
- The *complement edge slack* of an edge is the maximum delay of all paths not going through the edge.

Note that these "slack" definitions are different from the traditional slack definition in Section 4.1.1. It is easy to verify that the edge criticality is the probability that the edge slack is greater than the complement edge slack [120]. For a given edge $e = \langle i, j \rangle$, if both the edge slack S_e and the complement edge slack $S_{\tilde{e}}$ are approximated as Normal distributions, $S_e - S_{\tilde{e}}$ is also Normal and the corresponding edge criticality can be calculated based on the cumulative distribution function of $S_e - S_{\tilde{e}}$.

In a timing graph, the set of paths going through an edge $e = \langle i, j \rangle$ forms an *edge flow graph* G_e that consists of three parts: the edge input cone, the edge e itself, and the edge output cone. The edge slack S_e of the edge e is equal to

$$S_e = D_{\text{Incone}} + D_e + D_{\text{Outcone}}, \tag{4.62}$$

where D_{Incone} is the delay of the edge input cone, D_e is the delay of the edge e, and D_{Outcone} is the delay of the edge output cone. Based on the definitions of arrival time and required time in Section 4.1.1, the edge slack of $e = \langle i, j \rangle$ can be expressed as [120]:

$$S_e = \mathrm{AT}_i + D_e - \mathrm{RT}_j, \tag{4.63}$$

where AT_i is the arrival time at the node i and RT_j is the required time at the node j. Therefore, after the statistical timing analysis is complete, all edge slacks can be easily computed by a simple post-processing.

The computation of the complement edge slack is more complicated and requires to borrow the *cutset* concept from network theory. A cutset between the source and the sink is defined as a set of edges whose removal from the graph disconnects the source and the sink. In particular, a cutset is referred to as the *minimal separating cutset*, if it satisfies the condition that any two paths from the source to the sink have only one common edge in the cutset. Using the algorithm in [120], a set of minimal separating cutsets can be found to cover a given timing graph.

Given a minimal separating cutset C_e containing the edge e, let $C_{\tilde{e}} = C_e - \{e\}$ be a set of all the cutset edges except e. Then the set of all paths going through the edges in $C_{\tilde{e}}$ is identical to the set of all paths not going through the edge e. For this reason, the statistical maximum of all edge slacks of the edges in $C_{\tilde{e}}$ is exactly equal to the complement slack of the edge e.

A binary partition tree can be used to efficiently compute the complement edge slacks of all edges in a minimal separating cutset [120]. The key idea is to re-use the intermediate complement slack values to reduce the computational complexity. As shown in Figure 4.10, each

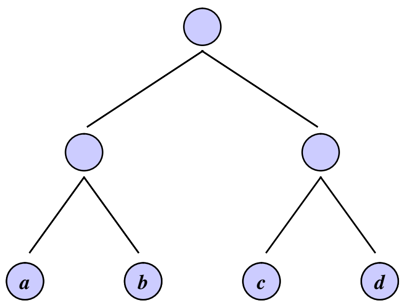


Fig. 4.10 A binary partition tree for efficiently computing the complement edge slacks for the edges a, b, c, and d.

leaf node in the binary partition tree represents one edge of the cutset. Each non-leaf node defines two sets of edges: the set of the node's children and the set of the edges that are not the node's children. With each node of the tree, we associate a node slack and a complement node slack. The *node slack* is the maximum of all edge slacks of its child edges. The *complement node slack* is the maximum of all edge slacks of the non-child edges. For a leaf node, these two slacks are exactly the edge slack and the complement edge slack. All node slacks and complement node slacks in the binary partition tree can be calculated by a bottom-up tree traversal followed by a top-down tree traversal [120], as shown in Algorithm 4.2. The computational complexity of Algorithm 4.2 is *linear* in tree size.

Algorithm 4.2 complement edge slack computation by a binary partition tree.

```
(1) Construct a binary partition tree of the cutset edges.
(2) Assign edge slacks to the leaf nodes.
(3) Traverse the tree bottom-up. For each non-leaf node, com-
    pute the node slack as the maximum of its children's node
    slacks.
(4) Set the complement node slack of the root node as negative
    infinity.
(5) Traverse the tree top-down. For each node, compute the com-
    plement node slack as the maximum of its parent's comple-
    ment node slack and its sibling node's node slack.
```

4.4 Statistical Leakage Analysis

In addition to timing variation, leakage variation is another critical issue at nano scale. The predicted leakage power is expected to reach 50% of the total chip power within the next few technology generations [4]. Therefore, accurately modeling and analyzing leakage power has been identified as one of the top priorities for today's IC design.

The most important leakage components in nanoscale CMOS technologies include *sub-threshold leakage* and *gate tunneling leakage* [93].

The sub-threshold leakage is due to the weak inversion when gate voltage is below the threshold voltage. At the same time, the reduction of gate oxide thickness facilitates tunneling of electrons through gate oxide, creating the gate leakage. Both of these leakage components are significant for sub-100 nm technologies and must be considered for leakage analysis.

Unlike many other performances (e.g., delay), leakage power varies substantially with process variations, which increases the difficulty of leakage estimation. Leakage variations can reach 20×, while delays only vary about 30%. The large-scale leakage variations occur, because leakage current exponentially depends on several process parameters such as V_{TH} and T_{OX} [93]. For this reason, leakage variation is typically analyzed in log domain. Most statistical leakage analysis techniques [17, 56, 68, 89, 106] approximate the logarithm of the leakage current $\log(I_{\mathrm{Leak}})$ as a function of random process variations. The existing statistical leakage analysis techniques can be classified into two broad categories: log-Normal approximation (linear leakage modeling for $\log(I_{\mathrm{Leak}})$ [17, 68, 89, 106]) and non-log-Normal approximation (quadratic leakage modeling for $\log(I_{\mathrm{Leak}})$ [56]). In what follows, we first describe the log-Normal approximation in Section 4.4.1 and then the non-log-Normal approximation will be discussed in detail in Section 4.4.2.

4.4.1 Log-Normal Approximation

A statistical leakage analysis flow typically contains two major steps: (1) standard cell library characterization and (2) full-chip leakage modeling. The objective of the standard cell library characterization is to approximate the leakage current of each logic cell by a response surface model. This modeling step is typically based on transistor-level simulations (or measurement models if available).

$$\log(I_{\mathrm{Cell}i}) = B_{\mathrm{Cell}i}^T \cdot \varepsilon + C_{\mathrm{Cell}i}, \tag{4.64}$$

where $B_{\mathrm{Cell}i} \in R^N$ contains the linear model coefficients, $C_{\mathrm{Cell}i} \in R$ is the constant term, $\varepsilon = [\varepsilon_1, \varepsilon_2, \ldots, \varepsilon_N]^T$ contains a set of independent standard Normal distributions to model process variations and N is the total number of these random variables. In (4.64), $\log(I_{\text{Cell}i})$ is the linear combination of multiple Normal distributions and, therefore, is Normal. It follows that $I_{\text{Cell}i}$ is log-Normal [81].

It is well-known that leakage current depends on input vector. The cell leakage in (4.64) can be the leakage current for a fixed input state or the average leakage current over all input states. For simplicity, we will not distinguish these two cases in this paper.

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

$$I_{\text{Chip}} = I_{\text{Cell}1} + I_{\text{Cell}2} + \cdots + I_{\text{Cell}M} \tag{4.65}$$

where M is the total number of logic cells in a chip. Equation (4.65) implies that the full-chip leakage current is the sum of many log-Normal distributions. Theoretically, the sum of multiple log-Normal distributions is not known to have a closed form. However, it can be approximated as a log-Normal distribution by using the Wilkinson's method [17] to match the first two moments of I_{Chip} in (4.65):

$$E(I_{\text{Chip}}) = \sum_{i=1}^{M} e^{m_{\text{Cell}i}+0.5\cdot\sigma^2_{\text{Cell}i}} \tag{4.66}$$

$$\begin{aligned} E\left(I^2_{\text{Chip}}\right) = {} & \sum_{i=1}^{M} e^{2\cdot m_{\text{Cell}i}+2\cdot\sigma^2_{\text{Cell}i}} + 2\cdot\sum_{i=1}^{M-1}\sum_{j=i+1}^{M} e^{m_{\text{Cell}i}+m_{\text{Cell}j}} \\ & \cdot e^{0.5\cdot\left(\sigma^2_{\text{Cell}i}+\sigma^2_{\text{Cell}j}+2\cdot\rho_{ij}\cdot\sigma_{\text{Cell}i}\cdot\sigma_{\text{Cell}j}\right)}. \end{aligned} \tag{4.67}$$

If $\log(I_{\text{Cell}i})$ and $\log(I_{\text{Cell}j})$ are both approximated as the linear response surface model in (4.64), the mean $m_{\text{Cell}i}$, the standard deviation $\sigma_{\text{Cell}i}$ and the correlation coefficient ρ_{ij} in (4.66) and (4.67), can be calculated by

$$m_{\text{Cell}i} = C_{\text{Cell}i} \tag{4.68}$$

$$\sigma_{\text{Cell}i} = \|B_{\text{Cell}i}\|_2 \tag{4.69}$$

$$\rho_{ij} = \frac{B^T_{\text{Cell}i}\cdot B_{\text{Cell}j}}{\sigma_{\text{Cell}i}\cdot\sigma_{\text{Cell}j}}, \tag{4.70}$$

where $||\bullet||_2$ denotes the 2-norm of a vector. Since we approximate I_{Chip} as a log-Normal distribution, $\log(I_{\text{Chip}})$ is Normal and its mean and standard deviation are determined by [17]:

$$E\left[\log(I_{\text{Chip}})\right] = 2 \cdot \log\left[E(I_{\text{Chip}})\right] - 0.5 \cdot \log\left[E(I_{\text{Chip}}^2)\right] \quad (4.71)$$

$$\sigma^2\left[\log(I_{\text{Chip}})\right] = \log\left[E(I_{\text{Chip}}^2)\right] - 2 \cdot \log\left[E(I_{\text{Chip}})\right]. \quad (4.72)$$

Substitute (4.66), (4.67) into (4.71), (4.72) yields the approximated log-Normal distribution for the full-chip leakage current.

4.4.2 Non-Log-Normal Approximation

Given the increasingly larger variations in nanoscale technologies, the aforementioned log-Normal approximation may result in inaccurate results, as it relies on the linear cell-level leakage model in (4.64) and only matches the first two moments for the chip-level leakage current in (4.65)–(4.72). To achieve higher accuracy, a quadratic approximation can be used, which, however, significantly increases the computational cost. For example, the total number of random variables can reach $10^3 \sim 10^6$ to model both inter-die and intra-die variations for a practical industrial design. In this case, a quadratic approximation will result in a $10^6 \times 10^6$ quadratic coefficient matrix containing 10^{12} coefficients!

The authors of [56] propose a projection-based algorithm to extract the optimal low-rank quadratic model for full-chip statistical leakage analysis. The algorithm proposed in [56] is facilitated by exploring the underlying *sparse* structure of the problem. Namely, any intra-die variation only impacts the leakage power in a small local region. Considering this sparse property, the statistical leakage analysis problem is formulated as a special form that can be efficiently solved by the *Arnoldi algorithm* and the *orthogonal iteration* borrowed from matrix computations. As such, an accurate low-rank quadratic model can be extracted with *linear* computational complexity in circuit size.

The statistical leakage analysis proposed in [56] starts from the standard cell library characterization where PROBE [57] is applied to fit the rank-K leakage model for each cell:

$$\log(I_{\text{Cell}i}) = \sum_{k=1}^{K} \lambda_{\text{Cell}ik} \cdot (P_{\text{Cell}ik}^T \cdot \varepsilon)^2 + B_{\text{Cell}i}^T \cdot \varepsilon + C_{\text{Cell}i}, \tag{4.73}$$

where $\lambda_{\text{Cell}ik}$, $C_{\text{Cell}i} \in R$, and $P_{\text{Cell}ik}$, $B_{\text{Cell}i} \in R^N$ are the model coefficients. In many practical applications, both $P_{\text{Cell}ik}$ and $B_{\text{Cell}i}$ are large but *sparse*, since many random variables in ε model the intra-die variations of the cells that are far away from the ith cell and they do not impact the leakage $I_{\text{Cell}i}$.

To simplify the notation, we define the following symbols to represent all cell leakage models in a matrix form:

$$\log(I_{\text{Cell}}) = [\log(I_{Cell1}) \quad \log(I_{Cell2}) \quad \cdots \quad \log(I_{\text{Cell}M})]^T \tag{4.74}$$

$$\Lambda_{\text{Cell}k} = [\lambda_{\text{Cell}1k} \quad \lambda_{\text{Cell}2k} \quad \cdots \quad \lambda_{\text{Cell}Mk}]^T \tag{4.75}$$

$$P_{\text{Cell}k} = [P_{\text{Cell}1k} \quad P_{\text{Cell}2k} \quad \cdots \quad P_{\text{Cell}Mk}] \tag{4.76}$$

$$B_{\text{Cell}} = [B_{\text{Cell}1} \quad B_{\text{Cell}2} \quad \cdots \quad B_{\text{Cell}M}] \tag{4.77}$$

$$C_{\text{Cell}} = [C_{\text{Cell}1} \quad C_{\text{Cell}2} \quad \cdots \quad C_{\text{Cell}M}]^T. \tag{4.78}$$

Comparing (4.74)–(4.78) with (4.73), it is easy to verify that:

$$\log(I_{\text{Cell}}) = \sum_{k=1}^{K} \Lambda_{\text{Cell}k} \otimes \left(P_{\text{Cell}k}^T \cdot \varepsilon\right) \otimes \left(P_{\text{Cell}k}^T \cdot \varepsilon\right) + B_{\text{Cell}}^T \cdot \varepsilon + C_{\text{Cell}}, \tag{4.79}$$

where $\otimes$ stands for the point-wise multiplication, i.e., $[a_1\ a_2\ \cdots]^T \otimes [b_1\ b_2\ \cdots]^T = [a_1b_1\ a_2b_2\ \cdots]^T$.

Using the cell leakage model in (4.79), we next describe the algorithm to efficiently extract the low-rank quadratic model of the full-chip leakage current. As shown in (4.65), the full-chip leakage current is the sum of all cell leakage currents. Applying the log transform to both sides of (4.65) yields:

$$\log(I_{\text{Chip}}) = \log[e^{\log(I_{\text{Cell}1})} + e^{\log(I_{\text{Cell}2})} + \cdots + e^{\log(I_{\text{Cell}M})}]. \tag{4.80}$$

Substituting (4.79) into (4.80) and applying a second order Taylor expansion, after some mathematical manipulations we obtain a quadratic model in the form of:

$$\log(I_{\text{Chip}}) = \varepsilon^T \cdot A_{\text{Chip}} \cdot \varepsilon + B_{\text{Chip}}^T \cdot \varepsilon + C_{\text{Chip}}, \tag{4.81}$$

where the model coefficients are given by:

$$C_{\text{Chip}} = \log\left(\frac{1}{\alpha}\right) \tag{4.82}$$

$$B_{\text{Chip}} = \alpha \cdot B_{\text{Cell}} \cdot \Phi \tag{4.83}$$

$$\begin{aligned} A_{\text{Chip}} = {} & \alpha \cdot \sum_{k=1}^{K} P_{\text{Cell}k} \cdot \text{diag}\left(\Phi \otimes \Lambda_{\text{Cell}k}\right) \cdot P_{\text{Cell}k}^T \\ & + \frac{\alpha}{2} \cdot B_{\text{Cell}} \cdot \text{diag}\left(\Phi\right) \cdot B_{\text{Cell}}^T \\ & - \frac{\alpha^2}{2} \cdot B_{\text{Cell}} \cdot \Phi\Phi^T \cdot B_{\text{Cell}}^T. \end{aligned} \tag{4.84}$$

In (4.82)–(4.84), diag($[a_1\ a_2 \cdots]^T$) stands for the diagonal matrix with the elements $\{a_1, a_2, \ldots\}$ and:

$$\alpha = \frac{1}{e^{C_{\text{Cell1}}} + e^{C_{\text{Cell2}}} + \cdots + e^{C_{\text{Cell}M}}} \tag{4.85}$$

$$\Phi = \begin{bmatrix} e^{C_{\text{Cell1}}} & e^{C_{\text{Cell2}}} & \cdots & e^{C_{\text{Cell}M}} \end{bmatrix}^T. \tag{4.86}$$

The values of α and Φ in (4.85), (4.86) can be computed with linear computational complexity. After α and Φ are known, the model coefficients C_{Chip} and B_{Chip} can be evaluated from (4.82), (4.83). Because the matrix B_{Cell} in (4.83) is sparse, computing the matrix-vector product $B_{\text{Cell}}\Phi$ has linear computational complexity. Therefore, both C_{Chip} in (4.82) and B_{Chip} in (4.83) can be extracted with linear complexity.

The major difficulty, however, stems from the *non-sparse* quadratic coefficient matrix A_{Chip} in (4.84). This non-sparse feature can be understood from the last term at the right-hand side of (4.84). The vector Φ is dense and, therefore, $\Phi\Phi^T$ is a dense matrix. It follows that $B_{\text{Cell}}\Phi\Phi^T B_{\text{Cell}}^T$ is dense, although B_{Cell} is sparse. For this reason, it would be extremely expensive to explicitly construct the quadratic coefficient matrix A_{Chip} based on (4.84).

To overcome this problem, the authors of [56] propose an iterative algorithm that consists of two steps: (1) Krylov subspace generation and (2) orthogonal iteration. Instead of finding the full matrix A_{Chip}, the proposed algorithm attempts to find an optimal low-rank approximation of A_{Chip}.

According to matrix theory [38], the optimal rank-R approximation of A_{Chip} is determined by the dominant eigenvalues $\{\lambda_{\text{Chip}1}, \lambda_{\text{Chip}2}, \ldots, \lambda_{\text{Chip}R}\}$ and eigenvectors $\{P_{\text{Chip}1}, P_{\text{Chip}2}, \ldots, P_{\text{Chip}R}\}$. The subspace generated by all linear combinations of these dominant eigenvectors is called the *dominant invariant subspace* [38] and is denoted as:

$$\text{span}\{P_{\text{Chip}1}, P_{\text{Chip}2}, \ldots, P_{\text{Chip}R}\}. \tag{4.87}$$

It is well-known that the dominant invariant subspace in (4.87) can be approximated by the following *Krylov subspace* [38]:

$$\text{span}\left\{Q_0, A_{\text{Chip}} \cdot Q_0, A_{\text{Chip}}^2 \cdot Q_0, \ldots, A_{\text{Chip}}^{R-1} \cdot Q_0\right\}, \tag{4.88}$$

where $Q_0 \in R^N$ is a non-zero vector that is not orthogonal to any dominant eigenvectors. We first show the algorithm to extract the Krylov subspace which gives a good approximation of the dominant invariant subspace. The extracted Krylov subspace is then used as a starting point for an orthogonal iteration such that the orthogonal iteration could converge to the dominant invariant subspace quickly.

The Arnoldi algorithm from matrix computations [38] is adapted to generate the Krylov subspace. The Arnoldi algorithm has been applied to various large-scale numerical problems and its numerical stability has been well-demonstrated for many applications, most notably, IC interconnect model order reduction [15]. Algorithm 4.3 summarizes a simplified implementation of the Arnoldi algorithm.

Algorithm 4.3 simplified Arnoldi algorithm.

```
(1) Randomly select an initial vector $Q_0 \in R^N$.
(2) $Q_1 = Q_0/||Q_0||_F$.
(3) For each $r = \{2, 3, \ldots, R\}$
(4) Compute $Q_r$ as:
```

$$\begin{aligned} Q_r &= A_{\text{Chip}} \cdot Q_{r-1} \\ &= \alpha \cdot \sum_{k=1}^{K} P_{\text{Cell}k} \cdot \text{diag}\left(\Phi \otimes \Lambda_{\text{Cell}k}\right) \cdot P_{\text{Cell}k}^T \cdot Q_{r-1} \end{aligned}$$

$$+\frac{\alpha}{2}\cdot B_{\text{Cell}}\cdot \text{diag}(\Phi)\cdot B_{\text{Cell}}^T\cdot Q_{r-1}$$
$$-\frac{\alpha^2}{2}\cdot B_{\text{Cell}}\cdot \Phi\Phi^T\cdot B_{\text{Cell}}^T\cdot Q_{r-1} \tag{4.89}$$

(5) Orthogonalize Q_r to all Q_i $(i = 1, 2, \ldots, r-1)$.
(6) $Q_r = Q_r/||Q_r||_F$.
(7) End For.
(8) Construct the Krylov subspace:

$$Q = [Q_R \quad \cdots \quad Q_2 \quad Q_1]. \tag{4.90}$$

Step (4) in Algorithm 4.3 is the key step of the Arnoldi algorithm. It computes the matrix-vector product $Q_r = A_{\text{Chip}}Q_{r-1}$. Since the matrix A_{Chip} is large and dense, Equation (4.89) does *not* construct the matrix A_{Chip} explicitly. Instead, it computes $A_{\text{Chip}}Q_{r-1}$ *implicitly*, i.e., multiplying all terms in (4.84) by Q_{r-1} separately and then adding them together. It is easy to verify that A_{Chip} in (4.84) is the sum of the products of many sparse or low-rank matrices. Therefore, the implicit matrix-vector product in (4.89) can be computed with linear computational complexity. Taking the last term in (4.89) as an example, there are four steps to compute $B_{\text{Cell}}\Phi\Phi^T B_{\text{Cell}}^T Q_{r-1}$, including: (1) $S_1 = B_{\text{Cell}}^T Q_{r-1}$ (sparse matrix multiplied by a vector); (2) $S_2 = \Phi^T S_1$ (dot product of two vectors); (3) $S_3 = \Phi S_2$ (vector multiplied by a scalar); and (4) $S_4 = B_{\text{Cell}} S_3$ (sparse matrix multiplied by a vector). All these four steps have linear computational complexity.

The Krylov subspace computed from Algorithm 4.3 is not exactly equal to the dominant invariant subspace. Starting from the matrix Q in (4.90), the next step is to apply an orthogonal iteration [38] which exactly converges to the dominant invariant subspace. Theoretically, the orthogonal iteration can start from any matrix. However, since the Krylov subspace Q gives a good approximation of the dominant invariant subspace, using Q as the starting point helps the orthogonal iteration to reach convergence quickly.

Algorithm 4.4 shows a simplified implementation of the orthogonal iteration algorithm. In (4.91), $Q^{(i-1)} \in R^{N\times R}$ is a matrix containing only a few columns, because R is typically small (e.g., around 10) in most applications. Therefore, similar to (4.89), $Z^{(i)}$ in (4.91) can be computed with linear complexity. For the same reason, the QR factorization in Step (5) of Algorithm 4.4 also has linear computational complexity, since $Z^{(i)} \in R^{N\times R}$ contains only a few columns.

Algorithm 4.4 simplified orthogonal iteration algorithm.

```
(1) Start from the matrix $Q \in R^{N\times R}$ in (4.90).
(2) $Q^{(1)} = Q$, where the superscript stands for the iteration index.
(3) For each $i = \{2, 3, \ldots\}$
(4) Compute $Z_i$ as:
```

$$\begin{aligned} Z^{(i)} &= A_{\text{Chip}} \cdot Q^{(i-1)} \\ &= \alpha \cdot \sum_{k=1}^{K} P_{\text{Cell}k} \cdot \text{diag}\left(\Phi \otimes \Lambda_{\text{Cell}k}\right) \cdot P_{\text{Cell}k}^{T} \cdot Q^{(i-1)} \\ &\quad + \frac{\alpha}{2} \cdot B_{\text{Cell}} \cdot \text{diag}\left(\Phi\right) \cdot B_{\text{Cell}}^{T} \cdot Q^{(i-1)} \cdot \\ &\quad - \frac{\alpha^2}{2} \cdot B_{\text{Cell}} \cdot \Phi\Phi^{T} \cdot B_{\text{Cell}}^{T} \cdot Q^{(i-1)} \end{aligned} \tag{4.91}$$

```
(5) $Q^{(i)}U^{(i)} = Z^{(i)}$ (QR factorization).
(6) End For.
(7) Construct the matrices:
```

$$Q_{\text{Chip}} = Q^{(i)} \tag{4.92}$$

$$U_{\text{Chip}} = U^{(i)}. \tag{4.93}$$

The orthogonal iteration in Algorithm 4.4 is provably convergent if the columns in the initial matrix Q are not orthogonal to the dominant invariant subspace [38]. After the orthogonal iteration converges, the optimal rank-R approximation of A_{Chip} is determined by the matrices Q_{Chip} and U_{Chip} in (4.92) and (4.93) [38]:

$$\tilde{A}_{\text{Chip}} = Q_{\text{Chip}} \cdot U_{\text{Chip}} \cdot Q_{\text{Chip}}^{T}. \tag{4.94}$$

Combining (4.94) with (4.81) yields:

$$\log(I_{\text{Chip}}) = \varepsilon^T \cdot \left(Q_{\text{Chip}} \cdot U_{\text{Chip}} \cdot Q_{\text{Chip}}^T\right) \cdot \varepsilon + B_{\text{Chip}}^T \varepsilon + C_{\text{Chip}}, \quad (4.95)$$

where C_{Chip} and B_{Chip} are given in (4.82) and (4.83).

Algorithms 4.3 and 4.4 assume a given approximation rank R. In practice, the value of R can be iteratively determined based on the approximation error. For example, starting from a low-rank approximation, R should be iteratively increased if the modeling error remains large. In most cases, selecting R in the range of $5 \sim 15$ provides sufficient accuracy. The aforementioned algorithm only involves simple vector operations and sparse matrix-vector multiplications; therefore, its computational complexity is *linear* in circuit size.

The quadratic function in (4.95) is N-dimensional, where N is typically large. It is not easy to estimate the leakage distribution directly from (4.95). Algorithm 4.5 describes a quadratic model compaction algorithm that converts the high-dimensional model to a low-dimensional one, while keeping the leakage distribution unchanged. It can be proven that the quadratic models in (4.95) and (4.97) are equivalent and the random variables $\{\delta_i;\ i = 1, 2, \ldots, R+1\}$ defined in (4.96) are independent standard Normal distributions [56].

Algorithm 4.5 quadratic model compaction algorithm.

(1) Start from the quadratic model in (4.95).
(2) $Q_{\text{Comp}} \cdot [U_{\text{Comp}} B_{\text{Comp}}] = [Q_{\text{Chip}} B_{\text{Chip}}]$ (QR factorization).
(3) Define a set of new random variables:

$$\delta = Q_{\text{Comp}}^T \cdot \varepsilon. \quad (4.96)$$

(4) Construct the low-dimensional quadratic model:

$$\log\left(I_{\text{Chip}}\right) = \delta^T \cdot \left(U_{\text{Comp}} U_{\text{Chip}} U_{\text{Comp}}^T\right) \cdot \delta + B_{\text{Comp}}^T \cdot \delta + C_{\text{Chip}}. \quad (4.97)$$

The quadratic function in (4.97) has a dimension of $R + 1$ which is much smaller than N. Based on (4.97), the PDF/CDF of $\log(I_{\text{Chip}})$ can be efficiently extracted using the algorithms described in Chapter 3.

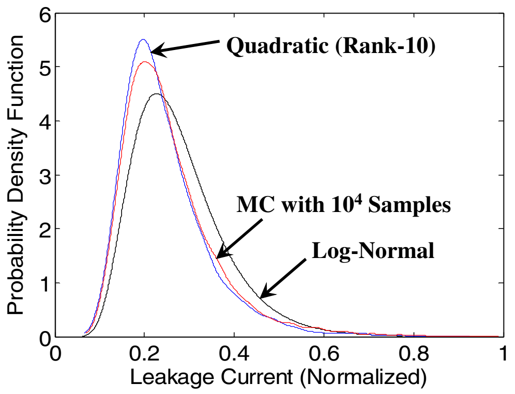


Fig. 4.11 Statistical leakage distribution for C432.

After that, the PDF/CDF of I_{Chip} can be easily computed by a simple nonlinear transform [81].

To demonstrate the efficacy of the non-log-Normal approximation, C432 (one of the ISCAS'85 benchmark circuits) is synthesized using a commercial 90 nm CMOS process. Figure 4.11 shows the leakage distributions extracted by three different approaches: the log-Normal approximation, the rank-10 quadratic approximation and the Monte Carlo analysis with 10^4 samples. As shown in Figure 4.11, the log-

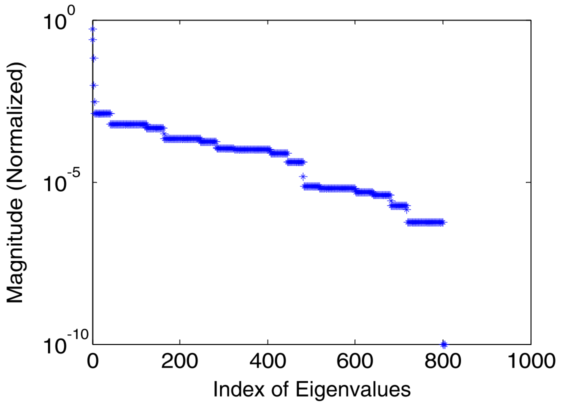


Fig. 4.12 Eigenvalue distribution for C432.

Normal approximation yields large errors, especially at both tails of the PDF which are often the points of great concern. In this example, the log-Normal approximation yields 14.93% error for worst-case leakage estimation, where the worst-case leakage is measured at the 99% point on the corresponding cumulative distribution function. The rank-10 quadratic approximation reduces the estimation error to 4.18%.

For testing and comparison, the full-rank quadratic leakage model is extracted for this example. Figure 4.12 shows the magnitude of the eigenvalues of the quadratic coefficient matrix. Note that there are only a few dominant eigenvalues. Figure 4.12 explains the reason why the low-rank quadratic approximation is efficient in this example.

5 Robust Design of Future ICs

Most existing robust IC design methodologies attempt to accurately predict performance distributions and then leave sufficient performance margins to accommodate process variations. As variations become more significant in the future, the continuously increasing performance margin can make it quickly infeasible to achieve high-performance IC design. For this reason, a paradigm shift in today's IC design methodologies is required to facilitate high yield, high performance electronic products that are based on less reliable nano-scale devices.

Toward this goal, the idea of adaptive post-silicon tuning has been proposed and successfully applied to various applications [16, 22, 63, 90, 112]. Instead of over-designing a fixed circuit to cover all process variations, adaptive post-silicon tuning dynamically configures and tunes the design based on the additional information that becomes available after manufacturing is complete, as shown in Figure 5.1.

For digital circuit applications, adaptive supply voltage and adaptive body bias are two widely-used techniques to reduce delay and leakage variations [22, 63, 112]. For analog circuit applications, device mismatch is one of the major design challenges and dynamic element

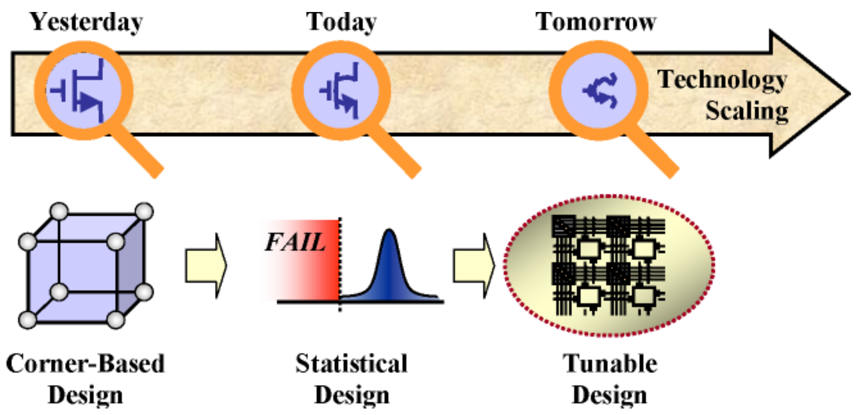


Fig. 5.1 Adaptive post-silicon tuning is one of the promising solutions to facilitate the future scaling of IC technologies.

matching has been applied to reduce random mismatches for analog devices (not only for transistors but also for resistors, capacitors, etc.) [16, 90].

The concept of tunable design poses a number of new challenges and opportunities. Extensive researches are required to solve the following problems.

- *Tunable circuit architecture.* Circuit configurability can be achieved in various ways, including: (1) continuous configuration on circuit parameters (e.g., power supply voltage, bias current, etc.); and (2) discrete configuration to change circuit topology and/or device size using switches (e.g., metal interconnects during manufacturing, CMOS switches post manufacturing, etc.). Advanced materials and devices such as phase-change switch and FinFET could possibly be applied to improve circuit performance. Exploring various tunable circuit architectures and analyzing their performance trade-offs would be an interesting research topic in the future.
- *Analysis and design of tunable circuit.* Traditional IC analysis and design methodologies were primarily developed for deterministic (i.e., fixed) circuit architectures. These tech-

niques, therefore, are ill-equipped to handle tunable circuits, due to their dynamic and configurable nature. To address this problem, a new CAD infrastructure is required to statistically analyze and optimize tunable circuits for adaptive post-silicon tuning.

Acknowledgments

We wish to thank all anonymous reviewers for their excellent comments, resulting in a significant improvement in the quality of this paper. We also would like to thank Soner Yaldiz at Carnegie Mellon University for his outstanding technical review of the manuscript. Jiayong Le would like to thank Extreme DA for all the support provided to this work.

References

[1] H. Abdel-Malek and A. Hassan, "The ellipsoidal technique for design centering and region approximation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 10, no. 8, pp. 1006–1014, August 1991.

[2] A. Agarwal, D. Blaauw, and V. Zolotov, "Statistical timing analysis for intra-die process variations with spatial correlations," in *IEEE International Conference on Computer Aided Design*, pp. 900–907, 2003.

[3] A. Agarwal, V. Zolotov, and D. Blaauw, "Statistical timing analysis using bounds and selective enumeration," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 9, pp. 1243–1260, September 2003.

[4] N. Akhiezer, *The Classical Moment Problem and Some Related Questions in Analysis.* Oliver and Boyd, 1965.

[5] C. Amin, N. Menezes, K. Killpack, F. Dartu, U. Choudhury, N. Hakim, and Y. Ismail, "Statistical static timing analysis: how simple can we get?," in *IEEE Design Automation Conference*, pp. 652–657, 2005.

[6] K. Antreich, H. Graeb, and C. Wieser, "Circuit analysis and optimization driven by worst-case distances," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 1, pp. 57–71, January 1994.

[7] A. Asenov, S. Kaya, and A. Brown, "Intrinsic parameter fluctuations in decananometer MOSFETs introduced by gate line edge roughness," *IEEE Trans. Electron Devices*, vol. 50, no. 5, pp. 1254–1260, May 2003.

[8] P. Avouris, J. Appenzeller, R. Martel, and S. Wind, "Carbon nanotube electronics," *Proceedings of The IEEE*, vol. 91, no. 11, pp. 1772–1784, November 2003.

[9] D. Bertseksa, *Nonlinear Programming.* Athena Scientific, 1999.

[10] S. Bhardwaj, S. Vrudhula, P. Ghanta, and Y. Cao, "Modeling of intra-die process variations for accurate analysis and optimization of nano-scale circuits," in *IEEE Design Automation Conference*, pp. 791–796, 2006.

[11] D. Boning, J. Panganiban, K. Gonzalez-Valentin, S. Nassif, C. Mcdowell, A. Gattiker, and F. Liu, "Test structures for delay variability," *IEEE International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems*, pp. 109–109, 2002.

[12] M. Bosley and F. Lees, "A survey of simple transfer-function derivations from high-order state-variable models," *Automatica*, vol. 8, pp. 765–775, 1972.

[13] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[14] J. Carballo and S. Nassif, "Impact of design-manufacturing interface on SoC design methodologies," *IEEE Design and Test of Computers*, vol. 21, no. 3, pp. 183–191, May-June 2004.

[15] M. Celik, L. Pileggi, and A. Odabasioglu, *IC Interconnect Analysis*. Kluwer Academic Publishers, 2002.

[16] K. Chan and I. Galton, "A 14b 100MS/s DAC with fully segmented dynamic element matching," in *IEEE International Solid State Circuits Conference*, pp. 2390–2399, 2006.

[17] H. Chang and S. Sapatnekar, "Full-chip analysis of leakage power under process variations, including spatial correlations," in *IEEE International Conference on Computer Aided Design*, pp. 523–528, 2005.

[18] H. Chang and S. Sapatnekar, "Statistical timing analysis under spatial correlations," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 9, pp. 1467–1482, September 2005.

[19] H. Chang, V. Zolotov, S. Narayan, and C. Visweswariah, "Parameterized block-based statistical timing analysis with non-Gaussian parameters, nonlinear delay functions," in *IEEE Design Automation Conference*, pp. 71–76, 2005.

[20] L. Chang, Y. Choi, D. Ha, P. Panade, S. Xiong, J. Bokor, C. Hu, and T. King, "Extremely scaled silicon nano-CMOS devices," *Proceedings of The IEEE*, vol. 91, no. 11, pp. 1860–1873, November 2003.

[21] C. Chen, C. Chu, and D. Wong, "Fast and exact simultaneous gate and wire sizing by Lagrangian relaxation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 18, no. 7, pp. 1014–1025, July 1999.

[22] T. Chen and S. Naffziger, "Comparison of adaptive body bias (ABB) and adaptive supply voltage (ASV) for improving delay and leakage under the presence of process variation," *IEEE Trans. Very Large Scale Integration Systems*, vol. 11, no. 5, pp. 888–899, 2003.

[23] C. Clark, "The greatest of a finite set of random variables," *Operations Research*, pp. 145–162, March-April 1961.

[24] G. Debyser and G. Gielen, "Efficient analog circuit synthesis with simultaneous yield and robustness optimization," in *IEEE International Conference of Computer Aided Design*, pp. 308–311, 1998.

[25] A. Devgan, "Transient simulation of integrated circuits in the charge-voltage plane," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 11, pp. 1379–1390, November 1996.

[26] A. Devgan and C. Kashyap, “Block-based static timing analysis with uncertainty,” in *IEEE International Conference on Computer Aided Design*, pp. 607–614, 2003.

[27] A. Devgan and R. Rohrer, “Adaptively controlled explicit simulation,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 13, no. 6, pp. 746–762, June 1994.

[28] A. Dharchoudhury and S. Kang, “Worst-case analysis and optimization of VLSI circuit performances,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, no. 4, pp. 481–492, April 1995.

[29] S. Director, P. Feldmann, and K. Krishna, “Statistical integrated circuit design,” *IEEE Journal of Solid-State Circuits*, vol. 28, no. 3, pp. 193–202, March 1993.

[30] P. Feldman and S. Director, “Integrated circuit quality optimization using surface integrals,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 12, pp. 1868–1878, December 1993.

[31] J. Fishburn and A. Dunlop, “TILOS: a posynomial programming approach to transistor sizing,” in *IEEE International Conference on Computer Aided Design*, pp. 326–328, 1985.

[32] K. Francken and G. Gielen, “A high-level simulation and synthesis environment for $\Delta\Sigma$ modulators,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 22, no. 8, pp. 1049–1061, August 2003.

[33] J. Friedman and W. Stuetzle, “Projection pursuit regression,” *Journal of the American Statistical Association*, vol. 76, no. 376, pp. 817–823, 1981.

[34] T. Fukuda, F. Arai, and L. Dong, “Assembly of nanodevices with carbon nanotubes through nanorobotic manipulations,” *Proceedings of The IEEE*, vol. 91, no. 11, pp. 1803–1818, November 2003.

[35] T. Gan, T. Tugbawa, B. Lee, D. Boning, and S. Jang, “Modeling of reverse tone etch back shallow trench isolation chemical mechanical polishing,” *Journal of Electrochemical Society*, vol. 148, no. 3, pp. G159–G165, March 2001.

[36] G. Gielen and R. Rutenbar, “Computer-aided design of analog and mixed-signal integrated circuits,” *Proceedings of the IEEE*, vol. 88, no. 12, pp. 1825–1852, December 2000.

[37] G. Gielen and W. Sansen, *Symbolic Analysis for Automated Design of Analog Integrated Circuits.* Springer, 1991.

[38] G. Golub and C. Loan, *Matrix Computations.* The Johns Hopkins Univ. Press, 1996.

[39] H. Graeb, S. Zizala, J. Eckmueller, and K. Antreich, “The sizing rules method for analog integrated circuit design,” in *IEEE International Conference of Computer Aided Design*, pp. 343–349, 2001.

[40] K. Heloue and F. Najm, “Statistical timing analysis with two-sided constraints,” in *IEEE International Conference on Computer Aided Design*, pp. 828–835, 2005.

[41] M. Hershenson, “Design of pipeline analog-to-digital converters via geometric programming,” in *IEEE International Conference of Computer Aided Design*, pp. 317–324, 2002.

[42] M. Hershenson, S. Boyd, and T. Lee, "Optimal design of a CMOS Op-Amp via geometric programming," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 1–21, January 2001.

[43] M. Hershenson, A. Hajimiri, S. Mohan, S. Boyd, and T. Lee, "Design and optimization of LC oscillators," *IEEE International Conference of Computer Aided Design*, pp. 65–69, 1999.

[44] M. Hershenson, S. Mohan, S. Boyd, and T. Lee, "Optimization of inductor circuits via geometric programming," in *IEEE Design Automation Conference*, pp. 994–998, 1999.

[45] J. Jess, K. Kalafala, S. Naidu, R. Otten, and C. Visweswariah, "Statistical timing for parametric yield prediction of digital integrated circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 11, pp. 2376–2392, November 2006.

[46] 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," in *IEEE Design Automation Conference*, pp. 69–72, 2006.

[47] K. Kasamsetty, M. Ketkar, and S. Sapatnekar, "A new class of convex functions for delay modeling and its application to the transistor sizing problem," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 7, pp. 779–788, July 2000.

[48] V. Kheterpal, T. Hersan, V. Rovner, D. Motiani, Y. Takagawa, L. Pileggi, and A. Strojwas, "Design methodology for IC manufacturability based on regular logic-bricks," in *IEEE Design Automation Conference*, pp. 353–358, 2005.

[49] M. Krasnicki, R. Phelps, J. Hellums, M. McClung, R. Rutenbar, and L. Carley, "ASP: A practical simulation-based methodology for the synthesis of custom analog circuits," in *IEEE International Conference of Computer Aided Design*, pp. 350–357, 2001.

[50] M. Krasnicki, R. Phelps, R. Rutenbar, and L. Carley, "MAELSTROM: efficient simulation-based synthesis for custom analog cells," in *IEEE Design Automation Conference*, pp. 945–950, 1999.

[51] X. Li, P. Gopalakrishnan, Y. Xu, and L. Pileggi, "Robust analog/RF circuit design with projection-based posynomial modeling," in *IEEE International Conference on Computer Aided Design*, pp. 855–862, 2004.

[52] X. Li, P. Gopalakrishnan, Y. Xu, and L. Pileggi, "Robust analog/RF circuit design with projection-based performance modeling," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 1, pp. 2–15, January 2007.

[53] X. Li, J. Le, M. Celik, and L. Pileggi, "Defining statistical sensitivity for timing optimization of logic circuits with large-scale process and environmental variations," in *IEEE International Conference on Computer Aided Design*, pp. 844–851, 2005.

[54] X. Li, J. Le, P. Gopalakrishnan, and L. Pileggi, "Asymptotic probability extraction for non-Normal distributions of circuit performance," in *IEEE International Conference on Computer Aided Design*, pp. 2–9, 2004.

[55] X. Li, J. Le, P. Gopalakrishnan, and L. Pileggi, "Asymptotic probability extraction for nonnormal performance distributions," *IEEE Trans. Computer-*

Aided Design of Integrated Circuits and Systems, vol. 26, no. 1, pp. 16–37, January 2007.

[56] X. Li, J. Le, and L. Pileggi, "Projection-based statistical analysis of full-chip leakage power with non-log-Normal distributions," in *IEEE Design Automation Conference*, pp. 103–108, 2006.

[57] X. Li, J. Le, L. Pileggi, and A. Strojwas, "Projection-based performance modeling for inter/intra-die variations," in *IEEE International Conference on Computer Aided Design*, pp. 721–727, 2005.

[58] X. Li and L. Pileggi, "Efficient parametric yield extraction for multiple correlated non-Normal performance distributions of analog/RF circuits," in *IEEE Design Automation Conference*, pp. 928–933, 2007.

[59] X. Li, J. Wang, L. Pileggi, T. Chen, and W. Chiang, "Performance-centering optimization for system-level analog design exploration," in *IEEE International Conference on Computer Aided Design*, pp. 422–429, 2005.

[60] K. Low and S. Director, "An efficient methodology for building macromodels of IC fabrication processes," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 8, no. 12, pp. 1299–1313, December 1989.

[61] W. Maly, H. Heineken, J. Khare, and P. Nag, "Design for manufacturability in submicron domain," in *IEEE International Conference on Computer Aided Design*, pp. 690–697, 1996.

[62] P. Mandal and V. Visvanathan, "CMOS Op-Amp sizing using a geometric programming formulation," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 1, pp. 22–38, January 2001.

[63] M. Mani, A. Singh, and M. Orshansky, "Joint design-time and post-silicon minimization of parametric yield loss using adjustable robust optimization," in *IEEE International Conference on Computer Aided Design*, pp. 19–26, 2006.

[64] M. Mckay, R. Beckman, and W. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, May 1979.

[65] C. Michael and M. Ismail, "Statistical modeling of device mismatch for analog MOS integrated circuits," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 2, pp. 154–166, February 1992.

[66] D. Montgomery, *Design and Analysis of Experiments.* Wiley & Sons, 2004.

[67] T. Mukherjee, L. Carley, and R. Rutenbar, "Efficient handling of operating range and manufacturing line variations in analog cell synthesis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 8, pp. 825–839, August 2000.

[68] S. Mukhopadhyay, A. Raychowdhury, and K. Roy, "Accurate estimation of total leakage in nanometer-scale bulk CMOS circuits based on device geometry and doping profile," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 3, pp. 363–381, March 2005.

[69] R. Myers and D. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments.* Wiley-Interscience, 2002.

[70] F. Najm and N. Menezes, "Statistical timing analysis based on a timing yield model," in *IEEE Design Automation Conference*, pp. 460–465, 2004.

[71] 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 submicron CMOS technologies," *IEEE Trans. Semiconductor Manufacturing*, vol. 12, no. 4, pp. 396–402, November 1999.

[72] S. Nassif, "Delay variability: sources, impacts and trends," in *IEEE International Solid-State Circuits Conference*, pp. 368–369, 2000.

[73] S. Nassif, "Modeling and analysis of manufacturing variations," in *IEEE Custom Integrated Circuits Conference*, pp. 223–228, 2001.

[74] W. Nye, D. Riley, A. Sangiovanni-Vincentelli, and A. Tits, "DELIGHT.SPICE: an optimization-based system for the design of integrated circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 7, no. 4, pp. 501–519, April 1988.

[75] S. Ohkawa, M. Aoki, and H. Masuda, "Analysis and characterization of device variations in an LSI chip using an integrated device matrix array," *IEEE Trans. Semiconductor Manufacturing*, vol. 17, no. 2, pp. 155–165, May 2004.

[76] M. Orshansky and A. Bandyopadhyay, "Fast statistical timing analyssi handling arbitrary delay correlations," in *IEEE Design Automation Conference*, pp. 337–342, 2004.

[77] M. Orshansky, J. Chen, and C. Hu, "Direct sampling methodology for statistical analysis of scaled CMOS technologies," *IEEE Trans. Semiconductor Manufacturing*, vol. 12, no. 4, pp. 403–408, November 1999.

[78] M. Orshansky and K. Keutzer, "A general probabilistic framework for worst case timing analysis," in *IEEE Design Automation Conference*, pp. 556–561, 2002.

[79] M. Orshansky, L. Milor, P. Chen, K. Keutzer, and C. Hu, "Impact of spatial intrachip gate length variability on the performance of high-speed digital circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 5, pp. 544–553, May 2002.

[80] M. Orshansky, L. Milor, and C. Hu, "Characterization of spatial intrafield gate CD variability, its impact on circuit performance, and spatial mask-level correction," *IEEE Trans. Semiconductor Manufacturing*, vol. 17, no. 1, pp. 2–11, February 2004.

[81] A. Papoulis and S. Pillai, *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, 2001.

[82] T. Park, T. Tugbawa, J. Yoon, D. Boning, J. Chung, R. Muralidhar, S. Hymes, Y. Gotkis, S. Alamgir, R. Walesa, L. Shumway, G. Wu, F. Zhang, R. Kistler, and J. Hawkins, "Pattern and process dependencies in copper damascene chemical mechanical polishing processes," in *VLSI Multilevel Interconnect Conference*, pp. 437–442, 1998.

[83] E. Pebesma and G. Heuvelink, "Latin hypercube sampling of Gaussian random fields," *Technometrics*, vol. 41, no. 4, pp. 303–312, November 1999.

[84] J. Pelgrom, C. Duinmaijer, and P. Welbers, "Matching properties of MOS transistors," *IEEE Journal of Solid-State Circuits*, vol. 25, no. 5, pp. 1433–1439, October 1989.

[85] R. Phelps, M. Krasnicki, R. Rutenbar, L. Carley, and J. Hellums, "Anaconda: simulation-based synthesis of analog circuits via stochastic pattern

search," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 19, no. 6, pp. 703–717, June 2000.

[86] L. Pileggi, H. Schmit, A. Strojwas, P. Gopalakrishnan, V. Kheterpal, A. Koorapaty, C. Patel, V. Rovner, and K. Tong, "Exploring regular fabrics to optimize the performance-cost trade-off," in *IEEE Design Automation Conference*, pp. 782–787, 2003.

[87] L. Pillage and R. Rohrer, "Asymptotic waveform evaluation for timing analysis," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 9, no. 4, pp. 352–366, April 1990.

[88] G. Plas, G. Debyser, F. Leyn, K. Lampaert, J. Vandenbussche, W. S. G. Gielen, P. Veselinovic, and D. Leenaerts, "AMGIE – a synthesis environment for CMOS analog integrated circuits," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 20, no. 9, pp. 1037–1058, September 2001.

[89] R. Rao, A. Devgan, D. Blaauw, and D. Sylvester, "Parametric yield estimation considering leakage variability," in *IEEE Design Automation Conference*, pp. 442–447, 2004.

[90] S. Ray and B. Song, "A 13b linear 40MS/s pipelined ADC with self-configured capacitor matching," in *IEEE International Solid State Circuits Conference*, pp. 852–861, 2006.

[91] B. Razavi, *Design of Analog CMOS Integrated Circuits*. McGraw, 2001.

[92] C. Robert and G. Casella, *Monte Carlo Statistical Methods*. Springer, 2005.

[93] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer CMOS circuits," *Proceedings of The IEEE*, vol. 91, no. 2, pp. 305–327, February 2003.

[94] B. Rubinstein, *Simulation and the Monte Carlo Method*. Wiley & Sons, 1981.

[95] S. Sapatnekar, *Timing*. Springer, 2004.

[96] S. Sapatnekar, V. Rao, P. Vaidya, and S. Kang, "An exact solution to the transistor sizing problem for CMOS circuits using convex optimization," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 11, pp. 1621–1634, November 1993.

[97] F. Schenkel, M. Pronath, S. Zizala, R. Schwencker, H. Graeb, and K. Antreich, "Mismatch analysis and direct yield optimization by spec-wise linearization and feasibility-guided search," in *IEEE Design Automation Conference*, pp. 858–863, 2001.

[98] G. Seber, *Multivariate Observations*. Wiley Series, 1984.

[99] A. Seifi, K. Ponnambalam, and J. Vlach, "A unified approach to statistical design centering of integrated circuits with correlated parameters," *IEEE Trans. Circuits and Systems – I*, vol. 46, no. 1, pp. 190–196, January 1999.

[100] Semiconductor Industry Associate, *International Technology Roadmap for Semiconductors*. 2005.

[101] M. Sengupta, S. Saxena, L. Daldoss, G. Kramer, S. Minehane, and J. Chen, "Application-specific worst case corners using response surfaces and statistical models," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 9, pp. 1372–1380, September 2005.

[102] B. Silverman, *Density Estimation for Statistics and Data Analysis.* Chapman & Hall/CRC, 1986.

[103] B. Simon, "The classical moment problem as a self-adjoint finite difference operator," *Advances in Mathematics*, vol. 137, no. 1, pp. 82–203, July 1998.

[104] K. Singhal and V. Visvanathan, "Statistical device models from worst case files and electrical test data," *IEEE Trans. Semiconductor Manufacturing*, vol. 12, no. 4, pp. 470–484, November 1999.

[105] A. Singhee and R. Rutenbar, "From finance to flip flops: A study of fast quasi-Monte Carlo methods from computational finance applied to statistical circuit analysis," in *IEEE International Symposium on Quality Electronic Design*, 2007.

[106] A. Srivastava, S. Shah, K. Agarwal, D. Sylvester, D. Blaauw, and S. Director, "Accurate and efficient gate-level parametric yield estimation considering correlated variations in leakage power and performance," in *IEEE Design Automation Conference*, pp. 535–540, 2005.

[107] G. Stehr, M. Pronath, F. Schenkel, H. Graeb, and K. Antreich, "Initial sizing of analog integrated circuits by centering within topology-given implicit specifications," in *IEEE International Conference of Computer Aided Design*, pp. 241–246, 2003.

[108] B. Stine, D. Boning, and J. Chung, "Inter- and intra-die polysilicon critical dimension variation," *SPIE Symposium on Microelectronic Manufacturing*, pp. 27–36, 1996.

[109] B. Stine, D. Boning, and J. Chung, "Analysis and decomposition of spatial variation in integrated circuit processes and devices," *IEEE Trans. Semiconductor Manufacturing*, vol. 10, no. 1, pp. 24–41, February 1997.

[110] B. Stine, D. Ouma, R. Divecha, D. Boning, J. Chung, D. Hetherington, I. Ali, G. Shinn, J. Clark, O. Nakagawa, and S. Oh, "A closed form expression for ILD thickness variation in CMP processes," in *CMP-MIC Conference*, pp. 266–273, 1997.

[111] P. Stolk, F. Widdershoven, and D. Klaassen, "Modeling statistical dopant fluctuations in MOS transistors," *IEEE Trans. Electron Devices*, vol. 45, no. 9, pp. 1960–1971, September 1998.

[112] J. Tschanz, J. Kao, S. Narendra, R. Nair, D. Antoniadis, A. Chandrakasan, and V. De, "Adaptive body bias for reducing impacts of die-to-die and within-die parameter variations on microprocessor frequency and leakage," *IEEE Journal of Solid-State Circuits*, vol. 27, no. 11, pp. 1396–1402, November 2002.

[113] S. Tsukiyama, M. Tanaka, and M. Fukui, "A statistical static timing analysis considering correlations between delays," in *IEEE Asia and South Pacific Design automation Conference*, pp. 353–358, 2001.

[114] J. Vanderhaegen and R. Brodersen, "Automated design of operational transconductance amplifiers using reversed geometric programming," in *IEEE Design Automation Conference*, pp. 133–138, 2004.

[115] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, S. Narayan, D. Beece, J. Piaget, N. Venkateswaran, and J. Hemmett, "First-order incremental block-based statistical timing analysis," *IEEE Trans. Computer-Aided Design*

of Integrated Circuits and Systems, vol. 25, no. 10, pp. 2170–2180, October 2006.

[116] Z. Wang and S. Director, “An efficient yield optimization method using a two step linear approximation of circuit performance,” in *IEEE European Design and Test Conference*, pp. 567–571, 1994.

[117] J. Wojciechowski and J. Vlach, “Ellipsoidal method for design centering and yield estimation,” *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 12, no. 10, pp. 1570–1579, October 1993.

[118] A. Wong, R. Ferguson, and S. Mansfield, “The mask error factor in optical lithography,” *IEEE Trans. Semiconductor Manufacturing*, vol. 13, no. 2, pp. 235–242, May 2000.

[119] J. Xiong, V. Zolotov, and L. He, “Robust extraction of spatial correlation,” *IEEE International Symposium on Physical Design*, pp. 2–9, 2006.

[120] J. Xiong, V. Zolotov, N. Venkateswaran, and C. Visweswariah, “Criticality computation in parameterized statistical timing,” in *IEEE Design Automation Conference*, pp. 63–68, 2005.

[121] Y. Xu, C. Boone, and L. Pileggi, “Metal-mask configurable RF front-end circuits,” *IEEE Journal of Solid-State Circuits*, vol. 39, no. 8, pp. 1347–1351, August 2004.

[122] Y. Xu, K. Hsiung, X. Li, I. Nausieda, S. Boyd, and L. Pileggi, “OPERA: optimization with ellipsoidal uncertainty for robust analog IC design,” in *IEEE Design Automation Conference*, pp. 632–637, 2005.

[123] Y. Xu, L. Pileggi, and S. Boyd, “ORACLE: optimization with recourse of analog circuits including layout extraction,” in *IEEE Design Automation Conference*, pp. 151–154, 2004.

[124] Y. Zhan, A. Strojwas, X. Li, L. Pileggi, D. Newmark, and M. Sharma, “Correlation aware statistical timing analysis with non-Gaussian delay distributions,” in *IEEE Design Automation Conference*, pp. 77–82, 2005.

[125] L. Zhang, W. Chen, Y. Hu, J.Gubner, and C. Chen, “Correlation-preserved non-Gaussian statistical timing analysis with quadratic timing model,” in *IEEE Design Automation Conference*, pp. 83–88, 2005.